(2)

Large Vocabulary Continuous Speech Recognition using Multi-Pass Search Algorithm

Akinobu LEE

July 2000

Abstract

Large vocabulary continuous speech recognition (LVCSR), which aims for automatic recognition of natural and fluent speech without lexical restriction, is a key technology for realizing intelligent speech media information processing and friendly human interfaces. It will give rise to many applications such as speech-to-text input, automatic captioning of broadcasting media, dictation of documents in judicial and medical domains, automatic transcription of lectures and meetings.

With the assumption of large vocabulary and continuous speech, the number of possible hypotheses increases explosively to a great amount. So an efficient search algorithm that can find the most likely sentence hypothesis fast is essential. Especially, as a large-scale, detailed and computationally expensive models such as long word N-gram chains or phonetic context-dependent acoustic models are typically used in LVCSR, the algorithm must apply them efficiently.

The one-pass search algorithm is widely adopted in current LVCSR systems, but applying those detailed models once at a time needs much computational cost, which hampers total processing efficiency and sometimes results in the search failures. Thus, we study a multi-pass approach for LVCSR. The search is divided into several passes and models are applied from simple ones to detailed ones. As the preliminary passes gradually narrow the search space, large-scale models are easily applied. It is also easy to introduce other complicated linguistic knowledge for further refinement.

In this thesis, we focus on multi-pass search algorithm for large vocabulary continuous speech recognition, especially a two-pass heuristic search strategy based on tree-trellis algorithm. The first pass performs recognition process using simple models, and outputs the results in some intermediate form. The second pass performs search again with detailed models, using the preliminary results as heuristic information. Specifically, we (1) use simple but powerful constraint such as single tree lexicon and word-pair grammar to compute good heuristics, and (2) store the results by the form of word trellis index,

that is free from information loss and realizes accurate scoring in the later pass. Then, (3) the second pass realizes an A* search by stack decoding using the results of the first pass as the heuristics. We realize the two-pass search algorithm based on these ideas in two recognition frameworks, one based on deterministic hand-written grammar and one based on statistical word N-gram language model.

In Chapter 2, a two-pass search algorithm based on word N-gram language model is pursued. Word N-gram is a statistical model that estimates probabilities of N word tuples from a large corpus, and is used mainly for non-specific task such as dictation systems. As all words can connect to any hypothesis, search becomes more difficult than grammar-based recognition.

Word graph method is an intermediate form popular among multi-pass recognition systems. As it definitely aligns words to a certain segment of speech input, it cannot represent the variance of matching length depending on the word context. This is eased by generating different hypotheses for every preceding word, but it remarkably increases computational cost. Therefore we introduce another intermediate form called "word trellis index", which keeps track of all survived word ends in successive frames with their scores. As the word boundaries are represented non-deterministically, they can be determined on the second search, where strict N-best scoring can be performed using a stack decoder. Approximation errors on the first pass can be also recovered on the later pass. The algorithm is implemented in our recognition engine Julius and evaluated on 5,000-word recognition of newspaper read speech, and it is shown that our engine performs accurate search almost equal to the conventional word-graph method, while the required memory amount was reduced from hundreds of megabytes to dozens of megabytes. Finally, recognition accuracy of 94.4% was achieved in that task.

In Chapter 3, we propose an efficient two-pass search algorithm for grammar-based large vocabulary continuous speech recognition. The grammar-based recognition is used in a field where the task is limited and the utterance pattern is definite. As the word hypothesis network expanded from the grammar grows explosively in large vocabulary, the popular one-pass search algorithm cannot handle them effectively. Also the grammatical rule is often not sufficient as constraint for word prediction in large vocabulary. Toward these problems, we (1) extract a compact category-pair constraint from the given grammar to represent a powerful heuristics, (2) organize a tree-structured word lexicon for every word category, and (3) keep index of the preliminary frame-wise results to predict the

probable words on the second pass. These methods are integrated together to avoid the explosion of search space and perform efficient search.

We implemented the methods into a recognition parser Julian, and compared it with a standard one-pass beam search decoder on a recognition task of a 5,000-word grammar. Our parser outperformed the one-pass beam search, causing no errors even when the beam width is one tenth. It also worked stably with any grammar regardless of its size or complexity. Finally, our parser achieved the word accuracy of 91.4% (with the real time factor of 2.2).

In Chapter 4, we propose a phonetic tied-mixture (PTM) acoustic model that realizes more efficient decoding. The dominant approach of acoustic modeling in LVCSR is large-scale tied-state triphone HMM. Although it makes detailed and accurate scoring, it needs a very large amount of Gaussian distributions that contains much redundancy. Our new model represents triphones efficiently by sharing a set of Gaussian distributions by the position of states in each phoneme, while having their own weights for different contexts. Evaluation results on a 20,000-word recognition task showed that our PTM model achieves almost the same accuracy of the best tied-state triphone, word error rate of 7.0%, with one fourth of the parameters defined.

Furthermore, we introduce a technique called Gaussian pruning, which aborts evaluation of Gaussian distributions of low probability for efficient acoustic computation. We found that computing only 3% of the Gaussian sets achieves the same accuracy in our PTM model. Several pruning methods are compared, and the beam pruning, where the pruning threshold is set independently for every vector dimension, realizes the best performance. Finally, acoustic computation cost is reduced to one fifth.

In Chapter 5, a Japanese dictation system integrating these techniques is described.

Our speech recognition engine is designed to have a common interface to various kinds of models. This portability realizes plug-and-play framework, where various language models and acoustic models developed at other sites are put and tested for evaluation. The best system achieves word accuracy of approximately 95% in a 20,000-word dictation task, and an efficient version achieves over 90% in real time recognition. It also works in a 60,000-word task, where the word accuracy is 94%.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Spoken language is one of the most essential ways of communication for us. It has been used in the human history for thousands of years before characters were invented, and babies acquire speaking naturally before reading and writing. Even in the modern society, telephone and television are more popular and easier way than postal mail, electric mail and books. But the main man-machine interface is still text-based or character-based ones such as keyboard or writing pen. As these methods are not natural and intuitive, people have to do training or get accustomed to its usage. Therefore, the technology of automatic speech recognition is essential to realize a truly human-friendly interface easily available for everyone. This technique incorporates the spoken language as a medium for man-machine interface, and also gives rise to applications such as speech-to-text input, automatic captioning of broadcasting media, dictation of documents in judicial and medical domains, automatic transcription of lectures and meetings.

Continuous speech recognition of spoken utterance has almost been realized for small or middle vocabulary (dozens or hundreds of words) in the past works. As the number of possible hypotheses is limited to a small amount, it is easy to compute all possible sentence hypotheses by full search and find the optimal hypothesis in a short time.

But in a large vocabulary continuous speech task, the search becomes substantially difficult since the number of possible sentence hypotheses grows enormously as the vocabulary size gets large. Furthermore, the language model and acoustic model for such task usually become large-scaled and detailed for high resolution, so their computational cost also increases remarkably. Even if the models have high potential accuracy, they are useless if they cannot be applied within practical computation. Therefore, to realize a continuous speech recognition of large vocabulary task, an algorithm that efficiently

applies large scale models and finds optimal solutions is essentially needed.

In this thesis, an efficient search algorithm for large vocabulary continuous speech recognition is studied. Our goal is to realize large vocabulary continuous speech recognition without search errors and perform it in real time with least accuracy degradation.

1.1 Problems

1.1.1 Framework of Continuous Speech Recognition with Probabilistic Models

The basic framework of continuous speech recognition based on information theory is described in Figure 1.1. Given an observation (i.e. speech input) X, the output is the most likely word sequence W.

$$W = \underset{w}{\operatorname{argmax}}\, P(w|X) \tag{1.1}$$

According to Bayes' rule, the probability is decomposed as follows.

$$P(w|X) = \frac{P(X|w)P(w)}{P(X)} \tag{1.2}$$

Since $P(X)$ does not affect the choice of W, recognition is formulated as follows.

$$\underset{w}{\operatorname{argmax}}\, P(w|X) = \underset{w}{\operatorname{argmax}}\, P(X|w)P(w) \tag{1.3}$$

The probability $P(X|w)$ means the output probability of the given speech for the word sequence which is given by an acoustic model, and the probability $P(w)$ is provided by a language model that evaluates how plausible the word sequence is in the context. The result will be the one that has the largest value of the product of the two probabilities. Actually, the linguistic constraint or vocabulary gives constraint of word sequence and acoustic probabilities are computed only for allowed words. Thus, the continuous speech recognition can be formulated as a search problem, to find the best word sequence under given linguistic and acoustic constraints.

The most popular acoustic model is a Hidden Markov Model (HMM), where speech signal is modeled as time-sequential automata that compute output probabilities for given speech segment (frame) and also have probabilistic transition. Typically an acoustic model is defined for a phoneme, and a sentence hypothesis is represented by the concatenation of the HMMs.

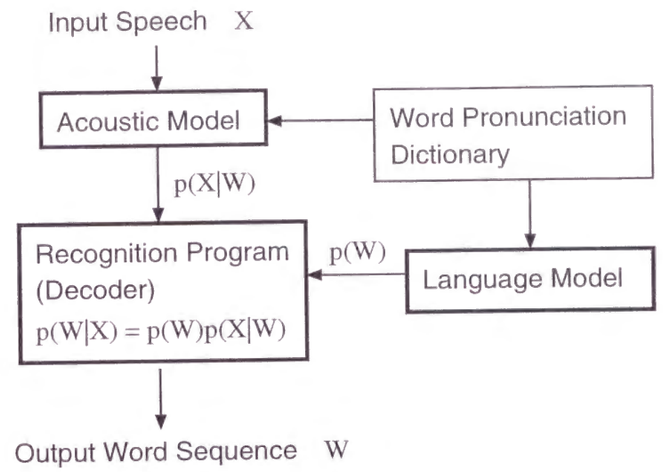


Figure 1.1: Continuous speech recognition

The acoustic probability of a hypothesis is computed by finding the best transition path on HMM trellis. The HMM trellis is the state space of the hypothesis HMM expanded to time axis of the input speech according to the allowed transitions (Figure 1.2). To find the best path, Viterbi algorithm is applied from the starting frame to the end frame, and the probability for every transition and for output of every frame are accumulated as the total acoustic probability.

1.1.2 Difficulty in Large Vocabulary CSR

The study of large vocabulary continuous speech recognition (LVCSR) includes problems in "continuous" speech recognition of "large vocabulary" literally. Each problem brings its own difficulty, but treating both at a same time makes problem much more difficult to solve. In general, "large vocabulary" means more than thousands of words in quantity. The number of hypotheses to appear grows enormously in proportion to the vocabulary size.

Recognizing continuous speech has another difficulty that the number of contained words and their lengths are totally unknown, and thus any combination of words must be examined. This time-dimensional ambiguity causes search space to grow enormously and makes the search hard to terminate. It is possible to do full search when the vocabulary is small, but it becomes a serious problem in large vocabulary task.

Thus, LVCSR is a difficult problem involving the generality of task and huge search

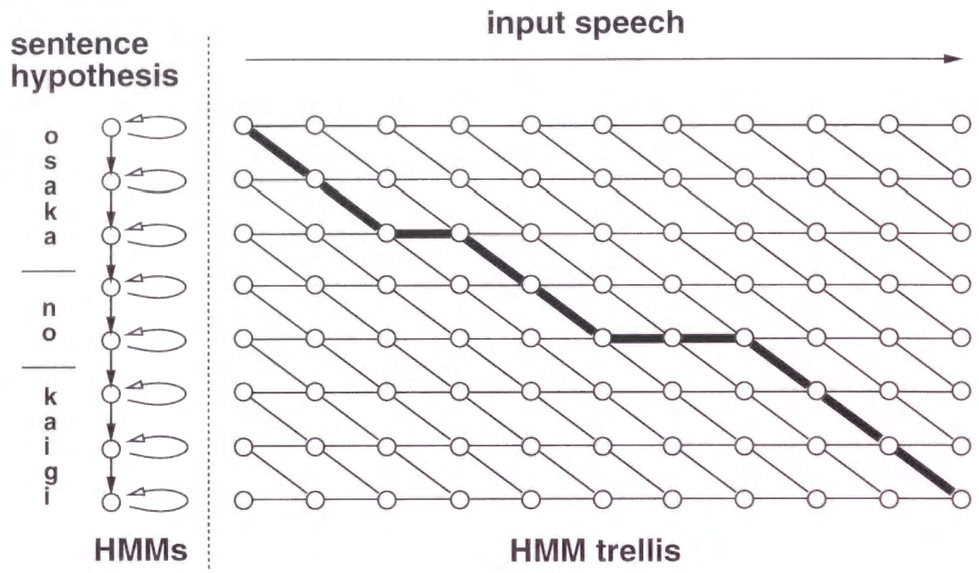


Figure 1.2: Viterbi algorithm on HMM trellis

space expanding the word axis by large vocabulary and the time dimension by handling continuous speech simultaneously. To realize this, besides accurate acoustic and language models, an effective search algorithm and total system integrating these factors in high level are needed.

1.1.3 Models for LVCSR

In general, LVCSR uses simple but large-scale statistical models such as context-dependent phoneme HMMs and word N-gram models. These models can fairly express the underlying knowledge of training sets with a statistical method. Recently these approaches have been proved to be successful in LVCSR, since its structure and training procedure are very simple enough to take advantage of the scalability of large training set, free from human errors.

But a large amount of training data is needed to reliably estimate these models. Although more complex models with many parameters can result in more accurate recognition, it is hard to collect so large training corpus for every task domain in many cases. And the growing computational cost is also a critical issue when realizing a recognition system. So the conventional method of giving the task grammar by hand is still useful.

Table 1.1: CI / CD models for phoneme /a/

CI	monophone	a
CD	biphone	a+k, a+d, a+i, ...
	triphone	s-a+k, s-a+d, s-a+i, ... t-a+k, t-a+d, t-a+i, ...

The following describes acoustic model and language model used in current LVCSR systems.

Written Grammar and Word N-gram Language Model

The language model is often given as either a set of grammatical rules or a statistical model like word N-gram to provide linguistic preference. A deterministic model, like context-free or regular grammar, also constrains the hypothesis space by itself, as it specifies the allowed word sequences explicitly and definitely. With an indeterministic and probabilistic model like probabilistic grammar or word N-gram, many words can follow other words given any probabilities, and thus search space often becomes large and makes it hard to search for the optimum hypothesis.

Context-Dependent Phoneme HMM

Baseline acoustic models are based on *monophone* HMMs, that is defined for a phoneme. This is also called a context-independent (CI) model. As the actual phonemes appeared in continuous utterance are known to vary by co-articulation effects with the surrounding environment, context-dependent (CD) models are most popular in current systems. A model that defines different phonemes depending on either the preceding or following phonemes is called *biphone* model, and a model considering dependency on both sides is called *triphone*. Table 1.1 shows example models for a phoneme `/a/`.

The model is typically defined as a mixture density HMM, where output probability for each HMM state is represented by weighed linear combination of several continuous Gaussian densities.

The number of parameters increases enormously when considering long dependencies. From a monophone model of k phonemes, k^2 biphones and k^3 triphones are made. For example, there are about 40 phonemes in Japanese, so there are $40^3 = 64000$ triphones. Thus, a sort of parameter sharing and tying in various model layers (models, states, mixtures, etc.) is essential for the context-dependent models.

The inter-word context dependency is not easy to handle. As the connections of words dynamically determined while in a search, a word must have multiple triphone variants for different connecting words on the end. Expanding all triphones on the edge for all possible hypotheses results in large computational cost.

But treating the cross-word dependency is essential for achieving high accuracy. Especially, as many short words consisting of only one or two phonemes appear very frequently in Japanese, such as /o/, /g a/, the effect will be larger. This is because Japanese texts are not space-separated and needs morpheme analysis, thus the unit of recognition is usually a morpheme rather than a word.

1.2 Approach

The most popular search method is one-pass beam decoding, in which input speech is scanned for only once and all the models and constraints are applied at a single pass. In large vocabulary task, applying full detailed models to all the hypotheses often costs too much to be handled.

In this thesis, we adopt a multi-pass search strategy. Models are separated and gradually applied in several passes. First, the whole input is scanned with a small and inexpensive models to get the preliminary result. Then the search is performed with detailed and expensive models. The advantage over the conventional one-pass decoding is that, by using the information of the preliminary pass as a heuristics, the search space of the second search becomes substantially narrower and computation cost will be much less. This advantage of multi-pass approach will become more remarkable when vocabulary grows larger and models become more complex.

As the actual search method should differ depending on the class of linguistic constraint, we pursue efficient search algorithms for both deterministic grammar and probabilistic word N-gram model.

The expensive acoustic matching cost is also one of the main problems in large vocabulary continuous speech recognition. We investigate an efficient parameter sharing method in acoustic model to suppress the total amount of acoustic matching.

1.3 Outline of the Thesis

In chapter 2, we make an overview of word N-gram based large vocabulary continuous speech recognition, and present a new heuristic search algorithm using word trellis index. It is an extension of trellis form to be efficient and accurate for large vocabulary task. Our developed portable recognition engine Julius is then explained with its implementation techniques. The algorithm is evaluated in 5,000-word dictation task.

In chapter 3, we present an efficient A* search method for grammar-based large vocabulary continuous speech recognition. After search problem with large vocabulary grammar is reviewed, the A* search using category-pair constraint as heuristics is presented. It is then evaluated in comparison with popular one-pass beam search. The algorithm is also implemented as a portable recognition parser called Julian.

In chapter 4, a new acoustic HMM for efficient decoding is investigated. A new scheme of parameter sharing between triphones is proposed against the popular shared-state triphones or tied-mixture models. It achieves both reliable parameter estimation and efficient decoding of large-scale acoustic models. The final system performance on 20,000-word dictation is reported.

In chapter 5, the Japanese Dictation System based on our recognition engine is introduced and its performance is shown. Our engine has been a part of the Japanese Dictation Toolkit Project, in which many researchers on speech recognition in Japan are engaged to develop free, standard and portable recognition modules as a common sharable platform for LVCSR. Its framework and the current system performance are reported.

Chapter 6 concludes the thesis.

Chapter 2

LVCSR Algorithm using Word N-gram Language Model

2.1 Introduction

Automatic speech recognition with unconstrained vocabulary becomes feasible as large-scale and well-organized text and speech corpus such as newspaper articles has been developed in recent years[1][2][3][4]. In this chapter, we address an efficient two-pass search algorithm for large vocabulary continuous speech recognition based on statistical language model.

We focus on multi-pass search that narrows the search space gradually and allows much detailed and expensive models to be applied with much less computational cost. This approach, however, has a potential problem that the incorrect preliminary matching results can cause serious errors that cannot be recovered on the later rescoring process. Especially, representing the preliminary results in popular word-graph form causes unrecoverable errors, because it definitely aligns words to a certain speech segment and does not allow re-alignment on the later pass. Using word-pair approximation eases this error, but generating different hypotheses for every preceding word increases computational cost.

Toward this problem, we propose a search algorithm with a trellis-based interface called "word trellis index". It keeps every hypothesis boundary indefinitely and allows re-aligning on the later pass. We also develop a portable LVCSR engine called Julius and evaluated it in recognition of read speech of newspaper articles.

In the next section, search algorithms for N-gram based large vocabulary continuous speech recognition are reviewed. Then various multi-pass search algorithms are compared

with respect to the intermediate form between passes and a new interface is proposed in the third section. The forth section gives the specification of our recognition engine and explains various search techniques adopted.

2.2 Word N-gram Language Model

An ideal language model for LVCSR should be able to represent the underlying rules independent on the text domain. As many researchers engaged in natural language processing have tried to describe a task-independent and universal grammar by their own hands in many years, they are not shown to be successful so far.

On the other hand, a statistical approach has become popular in recent years. It simply counts the frequency of a word or word sequence from a large text corpus such as newspaper articles, and models occurrences of text using the probability based on counts, without using any linguistic apriori. Given enough text corpus, it is considered to be able to reflect the knowledge underlying on the corpus.

The most popular statistical language model used in current LVCSR systems is word N-gram model. It is a Markov model where word probability is determined on the preceding $(N-1)$ words. In general, the probability for a sentence of n words $W = w_1, w_2, ..., w_n$ is explained in the following equation,

$$P(W) = P(w_1)P(w_2|w_1)P(w_3|w_1^2)P(w_4|w_1^3)...P(w_n|w_1^{n-1}) \tag{2.1}$$

where w_1^n is an abbreviation of word sequence $w_1, w_2, ..., w_n$. The word N-gram assumes each word probabilities to be dependent on only $(N-1)$ words.

$$P(w_n|w_1^{n-1}) = P(w_n|w_{n-N+1}^{n-1}) \tag{2.2}$$

Thus, this model represents the relationship of nearby words.

As all possible N-word tuples have to be assigned with some probability, a very large text database is needed to train the word N-gram. For example, when learning a word 3-gram for a vocabulary of 5,000 words, probability for 125×10^9 tuples should be estimated. As it is essentially impossible to covers all the tuples, the probability of an uncovered N-tuple is normally extrapolated by "backing-off", which assigns the discounted probability of "unseen" N-tuple by $N-1$-gram probability. Also, cutting off tuples of low frequency is effective to make training more reliably and reduce the model size. These methods are commonly adopted in current recognition models.

Currently, word 3-gram is most commonly used in the LVCSR systems. Some study concludes that using longer distance ($N = 4, 5$) has a little effect in accuracy, but the computational cost grows inevitably high. So the recent works focuses on other variants such as class N-gram or variable-length N-gram, or their hybrids.

2.3 Review of Search Algorithms for LVCSR

The purpose of the search algorithm in LVCSR is to find the best word sequence which most explains the input speech under the given language and acoustic model. As the search space becomes extremely large in LVCSR, "pruning" is the principal and basic idea. The unpromising hypotheses should be discarded or excluded from evaluation while searching. However, if the path to the optimum result is cut off in the middle of the search, it will never appear again and the search will fail. An efficient search is to find the optimum result in least steps without expanding unnecessary hypotheses. Actually, a dynamic and precise pruning is essential in LVCSR, since word N-gram allows the connection of any words with some probability and thus does not provide linguistic constraint onto the search space. Thus, the way the hypotheses are scored and thresholds are set is a crucial issue that directly affects the performance of pruning.

As pruning by only local information can results in such pruning errors, a sort of heuristic information or looking-ahead is effective to constrain the search space. The pruning becomes more stable as the look-ahead reaches more distant, but looking ahead for a long distance causes a delay of main speech processing and hampers real time processing. It can, however, sufficiently narrow the search space and total search time is faster. Thus, heuristics contributes to both speed and accuracy. This tendency will be remarkable on larger vocabulary.

In the following, the search algorithms used in current LVCSR systems are classified and compared in various views.

2.3.1 Unit of Scanning

First, search algorithms are classified by the type of scanning unit.

Frame Synchronous Search The search proceeds synchronously with the input time frame. All hypotheses are scanned simultaneously for every incoming frame, and word

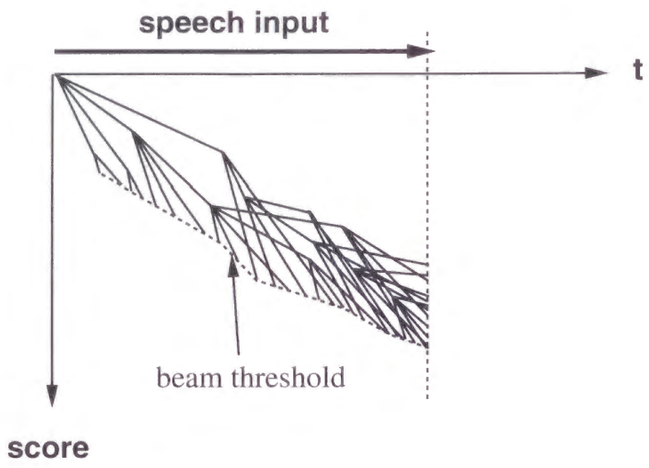


Figure 2.1: Frame synchronous search

hypotheses are expanded on demand. Figure 2.1 illustrates how the search proceeds. This is equivalent to the Viterbi operation of searching the best path on an HMM trellis. As the applied time length is identical for all hypotheses, their likelihood scores are also of the same scale and can be compared without any normalization. But since the words are expanded asynchronously with the search procedure, handling inter-word context dependency is complicated.

Beam pruning is usually used with the frame-synchronous search. For every incoming frame, all existing hypotheses are evaluated and only those of high likelihood can survive in the next frame. The threshold is determined on the node (HMM state) level, but word-level threshold can be introduced by setting another threshold only for the word ends.

Frame Asynchronous Search The search proceeds according to the unit of the models, i.e., words or phonemes. In each step, new hypotheses are expanded and matched to the input (Figure 2.2). This is equivalent to the tree search on word space. As the word expansion is simultaneous with the search step, inter-word context dependency can be easily introduced to the search. Other various word-level constraints such as semantic rules can be also easily integrated to the search.

As the matching length to the input differs between hypotheses, some normalization such as averaging or adding filler scores is needed to compare the hypotheses.

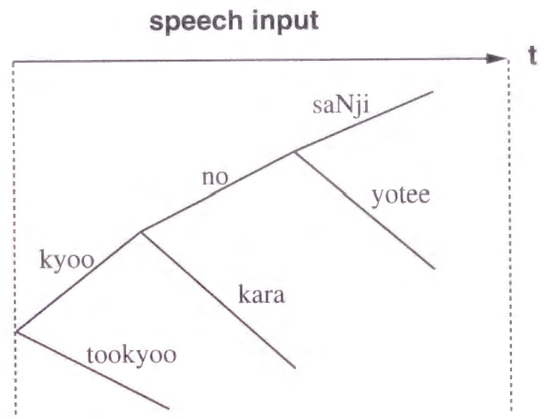


Figure 2.2: Frame asynchronous search

The beam pruning is also applicable. All existing hypotheses are expanded and those of high scores survive, keeping the same depth. Best-first stack decoding is also possible with this form.

Envelope Search[5] This is an intermediate method of the above two methods. The decoding is performed per word, while acoustic matching and hypothesis pruning are executed for each frame. The thresholds are kept for each frame and updated incrementally, activating or de-activating each hypothesis in turn.

2.3.2 Number of Scanning Passes

Next, various search algorithms for LVCSR are compared by the number of passes. From a view of search problem, this issue is closely related to the distance of the heuristics (or look-aheads) to be considered.

One Pass[6] The input is scanned only once, applying all the models together at a time without any look-ahead (Figure 2.3). Instead of using looking-ahead or heuristics, it uses available constraints as early as possible to get a precise score even in partial hypotheses from the start. As search is performed only once, it is possible that recognition finishes as soon as the input is ended. But when applying large and expensive models such as word 3-gram or triphone HMM at once, treating all the context dependencies for every partial hypothesis at the same time in parallel is very difficult to implement and can make the

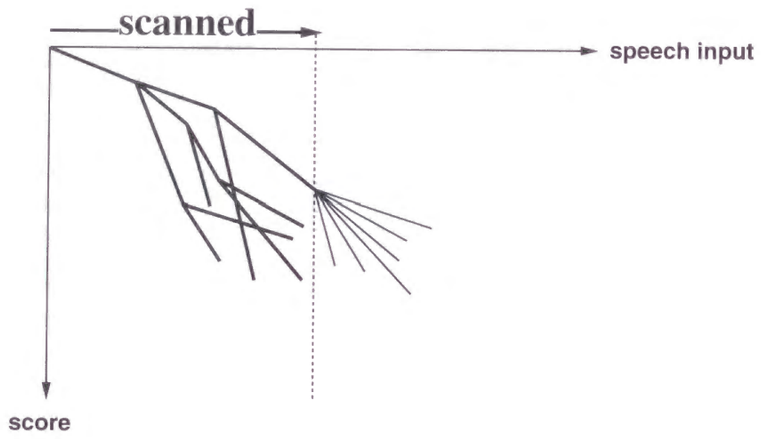


Figure 2.3: One-pass search

total recognition speed much slower.

Fast Match[7][2] In addition to the one-pass method, rough matching to several frames ahead with a simple models is performed in order to narrow the next upcoming word hypotheses (Figure 2.4). Only the words whose beginning phoneme has survived on the look-ahead matching are expanded. This method is called "fast match", and uses only rough and short-range heuristics for high-speed recognition.

Multi-Pass[8][9] This scans the whole input several times in turn, gradually narrowing the search space. The preliminary results serve as a long-distance look-ahead information that works for both pruning and heuristic information on the later pass (Figure 2.5). First the input is roughly scanned with simple models such as context-independent HMM, and some results are generated in a certain form to be used in the later pass. After the scan reaches the end, then the next search is started with more detailed models.

As the search space is gradually narrowed between the passes, complex and expensive models can be easily introduced. However, as the later pass cannot be started until the previous pass finished, the turn-around or response is slower than the one-pass method. The BYBLOS system[8] is the one that takes this approach.

Each pass can use different method. Typically, the first pass does simple frame-synchronous search for efficiency, and the later pass performs word unit-based search for accuracy.

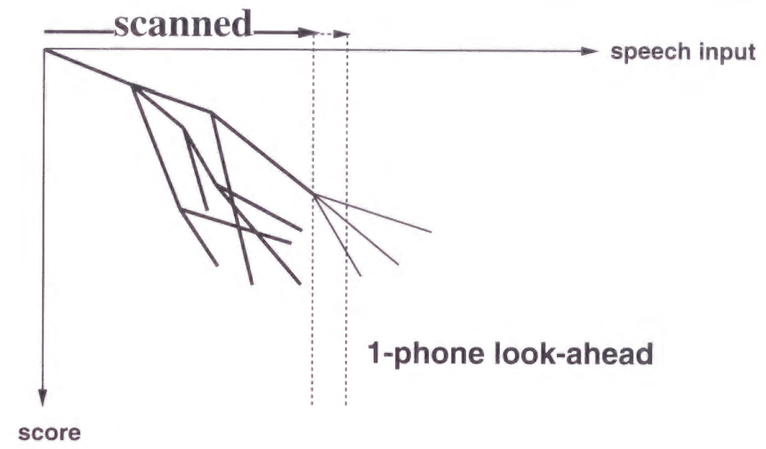


Figure 2.4: Fast match

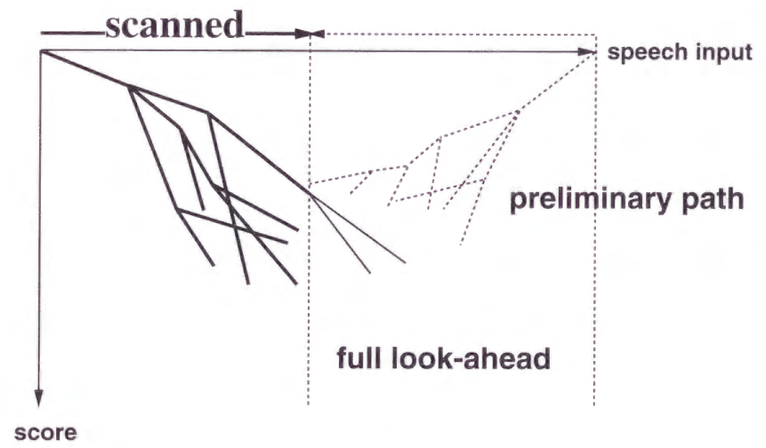


Figure 2.5: Multi-pass search

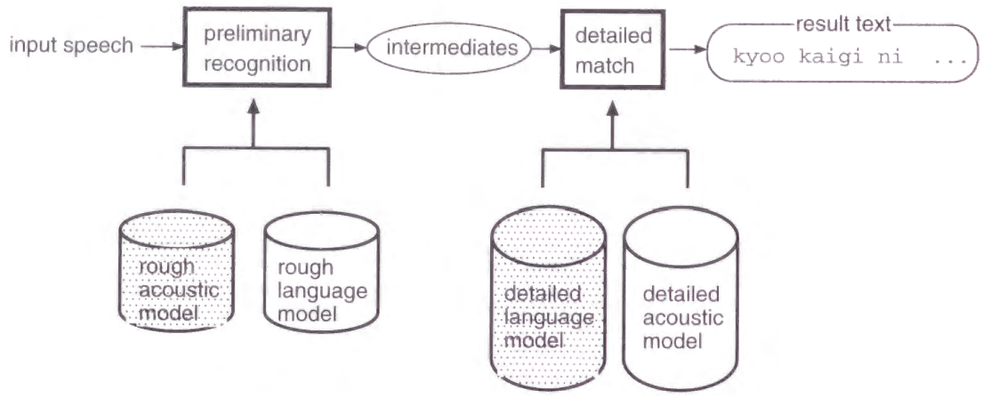


Figure 2.6: An LVCSR system using multi-pass search

2.3.3 Advantage of Multi-Pass Search

In statistical models, long distance models such as word 3-gram or 4-gram, triphone or quint-phone HMM are effective to improve recognition accuracy, as more precise context dependency is modeled. But the cost of its reference and application in search grows enormously as the models become more detailed. For example, to apply 3-gram, we must keep the history of last two words for every search nodes. Triphones must be applied considering the preceding and following phonemes, which dynamically change everywhere while search. Thus applying them all in a single pass is a very hard task and requires a highly sophisticated and complex implementation, which may results in a slower recognition.

A multi-pass search, on the other hand, can handle even such complex models easier, as the preliminary pass narrows the search space effectively. Especially, using the preliminary acoustic matching results is very effective in LVCSR, since word N-gram cannot, unlike grammatical rules, give deterministic constraints on the word connection. Figure 2.6 shows a data flow of typical LVCSR system using two-pass search algorithm.

2.4 Issues on Multi-Pass Search Algorithm

Based on the reviews in the previous section, we adopt a two-pass search algorithms described below. The first pass oriented for efficiency, uses left-to-right frame-synchronous beam search with word 2-gram and triphone HMM, where inter-word context dependency is discarded or approximately handled. The intermediate results are stored in a certain

form and used in the next pass to give heuristic information and to narrow the search space. The second pass performs word unit-based stack decoding. The direction is reversed from the first pass, right-to-left, to use the preliminary results as full look-ahead information. As search space is narrowed enough, detailed models such as word 3-gram and triphone with cross-word dependency can be used.

This approach, however, has a potential problem that incorrect results of the preliminary pass can cause serious errors in the later pass. As the later pass depends largely on the preliminary pass, the errors by the rough matching or approximations on the preliminary pass may lead to recognition errors on the final result. Thus, whether the errors on the first pass are recoverable or not is a critical issue to achieve effective recognition in multi-pass search,

In this section, several issues relating with multi-pass search algorithm are investigated and compared. The intermediate forms and their relationship with the N-best approximation are discussed. Although we focus on only two-pass algorithm for convenience, the discussion below can be applied to other multi-pass algorithms.

2.4.1 N-best Approximation

In order to re-score them with farther constraints on the later pass, the first pass is expected to output N-best candidates. However, as the matching length of a word varies depending on the context by co-articulation effect, multiple hypotheses must be handled for different contexts. As it is difficult to deal with separate hypotheses for all possible contexts in entire sentence (sentence-dependent), approximation that limits the range of dependency is normally adopted in LVCSR. Here, two methods are compared.

Word-Pair Approximation This assumes dependency on only a preceding word[10]. Multiple hypotheses are handled for each preceding word as illustrated in Figure 2.7. In Figure 2.7, the two lines show the best Viterbi path for hypotheses `(A,C)` and `(B,C)`. As the length of word `C` varies depending on the previous words `A` or `B`, multiple paths are kept in the overlapping area. This method has been proved to be a good approximation for the N-best scores, and adopted in many current LVCSR systems.

The disadvantage is that, although it enables approximate N-best handling, the multiplication of hypotheses still costs much in LVCSR. As all word hypotheses should be multiplied, the total search space will grow very large. In a frame-synchronous search,

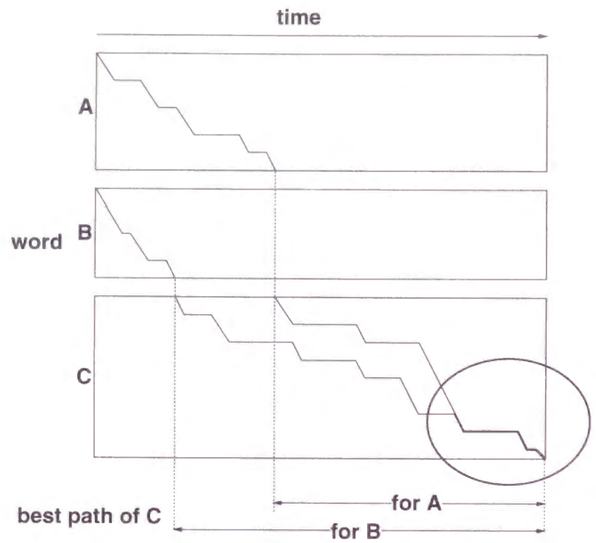


Figure 2.7: Word-pair approximation

word lexicons are multiplied in parallel corresponding to the preceding word.

Moreover, much approximation error occurs around a short word, because the matching length of such word is too short to absorb the difference of the best paths caused by the context. Especially in Japanese, the recognition unit is often a morpheme that is shorter than ordinary words and this approximation will result in more errors than other language.

One-Best Approximation Assuming no dependency, it keeps only the best hypothesis for each path discarding the context dependency. Figure 2.8 illustrates how it works. The best Viterbi path of word `C` for the preceding word `A` will be overridden by that of `B` at the circled point in the figure. As a result, the boundary of word `C` is deterministically fixed according only to the context word `B`, and the score for the sequence `(A,C)` becomes not accurate.

Although the scores of the hypotheses other than the best sequence contain boundary errors, the computational cost is much smaller than the word-pair approximation, since it uses only a single lexicon.

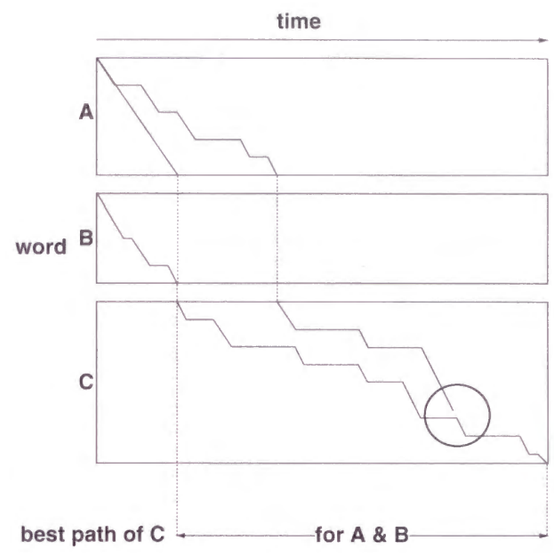


Figure 2.8: One-best approximation

2.4.2 Interface Between Passes

Next, the intermediate forms between passes in the multi-pass search algorithm are investigated in details. As the later pass depends totally on the results of the preliminary pass, how the results are stored and what kind of information is kept are essential issues.

N-best Candidates The preliminary results are output as N-best sentence hypotheses[8]. At the later pass, they are re-sorted using more accurate models. They can also be represented as a simple lattice as described in Figure 2.9, where each arc denotes a word hypothesis and a node denotes a boundary between them. This interface is simple and trivial, thus makes it easy to integrate the ordinary one-pass decoder with other linguistic knowledge.

But a precise N-best search is required on the first pass. As noted in the previous section, precise N-best search costs much and makes the implementation difficult. Moreover, this format is not efficient because it generates many similar candidates that differs by only one word and hundreds of candidates are needed to get enough variation. This becomes remarkably annoying when input is much longer.

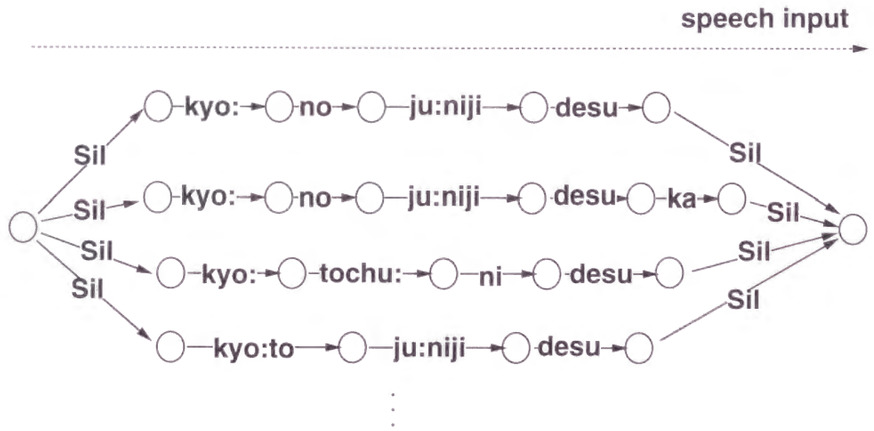


Figure 2.9: N-best candidates

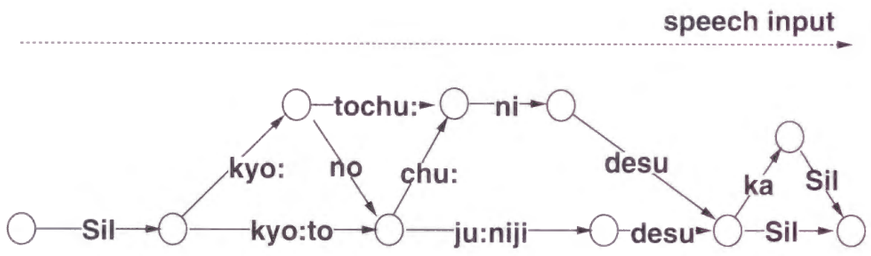


Figure 2.10: Word-graph form

Word-Graph Form Word graph represents N-best candidates in a compact graph by merging the same partial word sequence[11] as shown in Figure 2.10. Each word hypothesis is stored with its beginning and ending time and the acoustic and linguistic scores. As it represents many candidates efficiently, this form has been adopted in many multi-pass LVCSR systems.

As it binds each word hypothesis to a certain segment of speech input, re-alignment on the later pass is essentially not allowed. In other words, it lacks information on non-determinacy of word boundaries. Even if the time constraint can be ignored on the later pass, their possible sequences are still bound by the results of the first pass.

Thus, the context dependency of word boundaries must be handled on the first pass. By the same reason, precise and expensive acoustic models should be also applied on the first pass. These cause much computational cost as vocabulary gets large. Introducing the

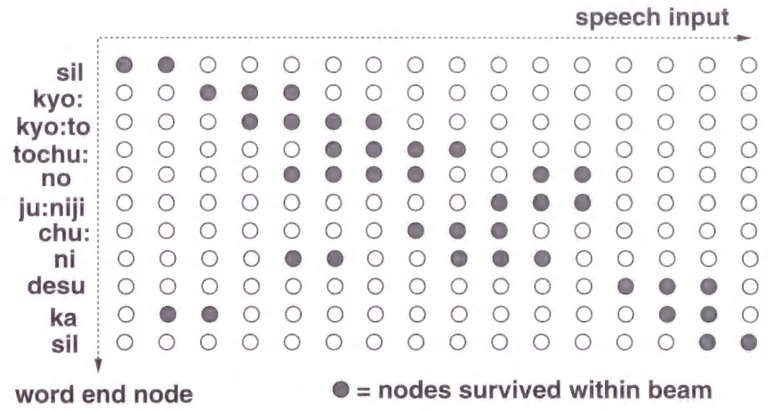


Figure 2.11: Trellis form

word-pair approximation on the first pass and generating multiple hypotheses dependning on the preceding word can ease this cost, but as the boundary cannot be moved on the later pass, the approximation errors cannot be recovered on the later pass.

Trellis Form Instead of keeping only the fully survived word hypotheses, trellis form keeps all word ends on every time-frame[12]. This redundancy of keeping all possible word boundaries allows the later pass to re-estimate the boundaries. As the search space is narrowed enough on the later pass, the precise sentence-dependent score can be obtained more easily than a word graph method. On the other hand, the later pass needs re-expansion of trellis to determine the boundary and this may result in high cost on the later pass.

This form generalizes the word-graph by loosing the time-constraint of word boundaries. Compared with the word graph that keeps only the best boundary of the path, all the possible boundaries (word ends) are kept and the best one isq selected on the later pass by re-scoring. Thus the errors caused by one-best or word-pair approximation on the preliminary pass become recoverable on the later pass, and more accurate recognition can be achieved.

There are two advantages in re-alignment on the later pass. First, the space-narrowed second pass realizes fully sentence-dependent scoring and recovers approximation errors. Second, expensive word-pair approximation is not needed. The different word-ends dependent on the previous word context will hopefully be included as trellis nodes in different

time-frames. So even one-best approximation will be enough for those ends to survive. Both acoustic and linguistic errors are recovered on the second pass that performs re-alignment and re-scoring.

However, the trellis itself is a set of nodes and scores, and does not have explicit predictive information for the second pass. Vocabulary-level constraint alone will suffice on single word recognition[13], or grammar-level constraint will do in small vocabulary CSR[12]. In LVCSR, where search space is huge, space narrowing based on preliminary acoustic matching is essential.

As the trellis form merely provides likelihood for all possible hypotheses, it only serves as heuristic information and does not explicitly narrow the search space of the later search pass. Even if the heuristics have admissibility to perform a fast search with much less steps, all possible word hypotheses should be considered each time a hypothesis is expanded, because the preliminary results cannot explicitly predict the upcoming words. Applying high-cost models without narrowing search space needs much computational costs, and makes it hard to realize the search.

2.5 Multi-Pass Search using Word Trellis Index

Comparison of the interface forms for multi-pass search is summarized as follows.

N-best candidates: so inefficient that hundreds of candidates must be computed on the first pass.

Word-graph form: compact, but boundary errors are not recoverable on the later pass.

Trellis form: Enables accurate re-alignment of boundaries, but too expensive to apply on large vocabulary task.

Based on this comparison, we propose a multi-pass search algorithm that extends the trellis form to be applicable to LVCSR. To predict next words explicitly from the preliminary results of the acoustic matching, we store the indexes of words for every frame whose end nodes survived within beam. On the second pass, only words whose end nodes appeared in the index at the starting frame of the last word hypothesis are expanded. To estimate the end frame efficiently on the second pass, the word beginning frame corresponding to each end node on the first pass is also stored. We call this interface as "word trellis index".

2.5.1 Word Trellis Index

Figure 2.12 illustrates the word trellis index and shows how it works on the second pass. It consists of the following components:

- Word end scores of partial hypotheses survived within beam for every frame
- Corresponding word beginning frames
- Indexes of survived word ends for each time-frame

The upper half of the figure is the word trellis index derived from the first pass, and the lower half shows how the second pass proceeds.

It keeps all word ends appeared successively on the sequential frames instead of only on the best one, and word boundaries are kept indeterministically to the later pass. The re-alignment of the boundaries on the second pass can be done efficiently using a stack decoding, and finally accurate scores can be obtained with much less computational cost.

Furthermore, even one-best approximation is supposed to work sufficiently with the word trellis index. The word boundaries depending on the context by the word-pair approximation can also be kept on the trellis with one-best approximation. Thus, even with the simple one-best approximation, the optimum boundaries are not lost on the first pass and precise result can be obtained on the second pass.

As the stored begginning frame for each word ends on the first pass may include some error, the word expansion on the second pass may not be accurate. However, their boundary errors are absorbed by keeping all word-ends appeared in successive time-frames. Furthermore, we allow expanding words in several frames around the estimated frame to ensure the optimum hypothesis to be appeared on the second pass.

2.5.2 Search Algorithm

On the first pass, conventional left-to-right time-synchronous beam search is performed on the whole input, and a word trellis index is generated by keeping all word ends survived within beam width. The second pass performs a best-first stack decoding search in backward (right-to-left) direction, using the word trellis index as both heuristics and word prediction.

The detailed procedure of the second pass is described below. Here, $W_n = w_n, w_{n-1}, ..., w_1$ is a partial hypothesis (w_1 is the end of the utterance), and T is input frame length.

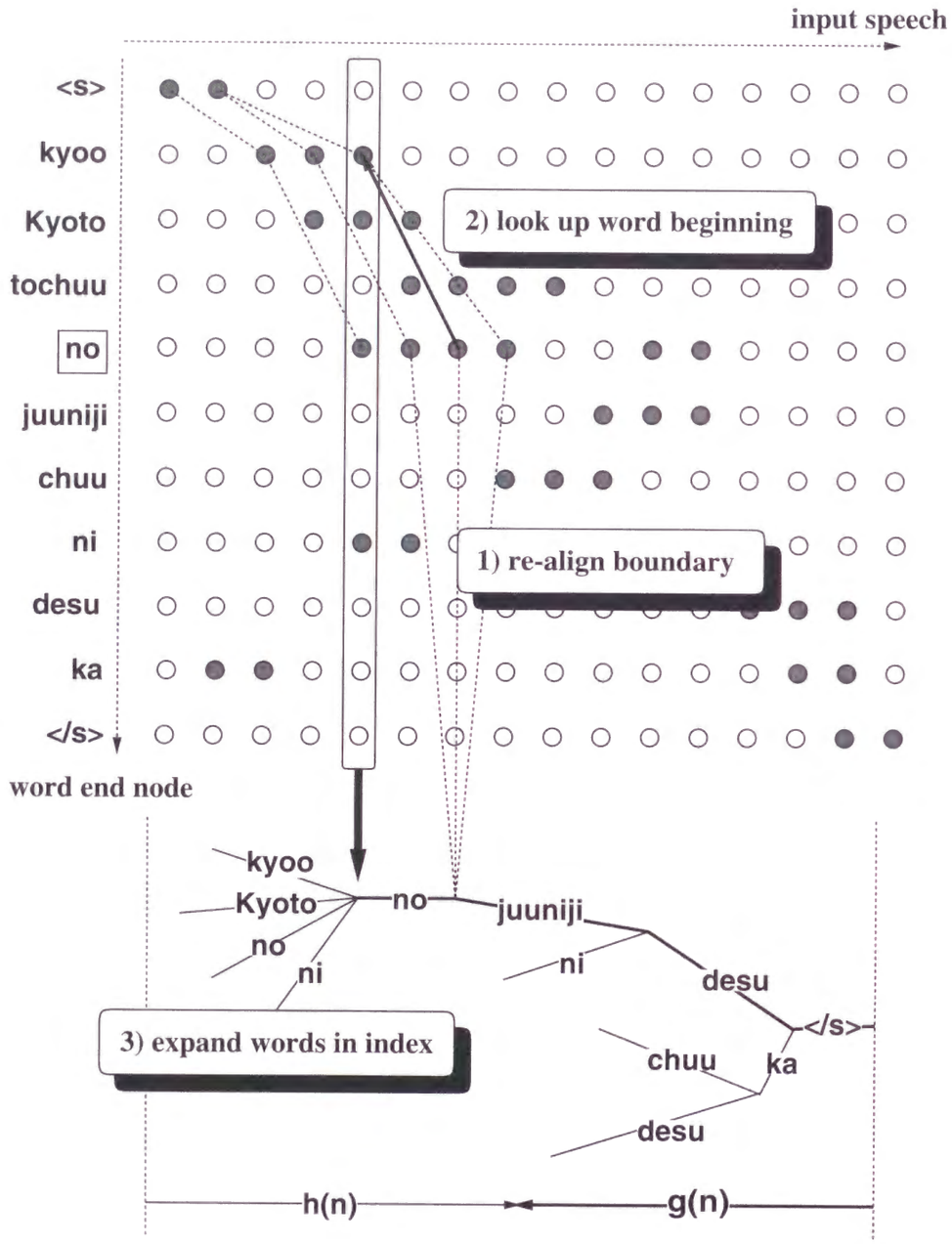


Figure 2.12: Word trellis index and its use on the second pass

$g(W_n, t)$ is the backward (right-to-left) score of W_n from time t to T, and $h(w, t)$ is the forward (left-to-right) heuristic score on end frame t of word w derived from the first pass. The total score of hypothesis W_n is $f(W_n)$, and the actual word end frame of word w_n is t_n.

1. For every word w_1 that can appear at the end of utterance, generate an initial sentence hypothesis W_1 and store to the hypothesis stack.

$$f(W_1) = h(w_1, T) \tag{2.3}$$
$$t_1 = T \tag{2.4}$$

2. Pop the best hypothesis (W_n) from the hypothesis stack. If its search end flag is set, it means that the hypothesis already reached the end, so output it as the final result and exit.

3. Estimate the next boundary time $\hat{t}_{n+1}$ according to the word trellis index. Extract the beginning time of last word w_n of W_n by looking up the index of the frame t_n (procedure 2 of Figure 2.12)

4. Compute the backward trellis of the w_n.

$$g(W_n, t) = \begin{cases} \max_{t'}\{\beta(w_n, t, t') + g(W_{n-1}, t')\} & \text{if } n > 1 \\ \beta(w_n, t, T) & \text{if } n = 1 \end{cases} \tag{2.5}$$

where $\beta(w_n, t_1, t_2)$ is backward score of word w from t_2 to t_1.

5. If $\hat{t}_{n+1} = 0$, set the search end flag of the current W_n and push back to the stack with the following score, and go to 2.

$$f(W_n) = g(W_n, 0) \tag{2.6}$$

6. Generate new hypotheses by expanding each word in the index of time $\hat{t}_{n+1}$ from W_n (procedure 3 of Figure 2.12). The best word end of each new word w_n is selected around the frame $\hat{t}_{n+1}$ (procedure 1 of Figure 2.12).

$$f(W_{n+1}) = \max_t\{h(w_{n+1}, t) + g(W_n, t)\} \tag{2.7}$$
$$t_{n+1} = \arg\max_t\{h(w_{n+1}, t) + g(W_n, t)\} \tag{2.8}$$

7. Push all the new hypotheses to the stack. Go to 2.

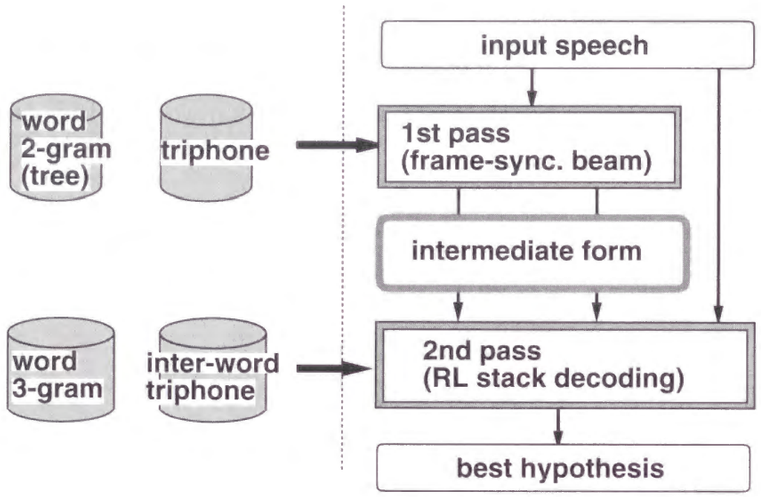


Figure 2.13: Overview of Julius

2.6 Search Techniques for LVCSR

Based on the discussions above, we have developed a 2-pass recognition engine Julius that can be applicable for various LVCSR tasks. To achieve a large vocabulary dictation in less computation time, the engine incorporates many up-to-date search techniques. In this section, the overview of Julius and adopted search techniques are explained in detail.

2.6.1 Overview of Julius

Julius uses two pass search algorithm proposed in previous section 2.5.2. It can perform both one-best approximation and word-pair approximation selectively, and word trellis index and word graph as inter-pass interface.

Figure 2.13 shows an overview of Julius. The input is a continuous speech segmented with a pause. The first pass processes the entire input, and then generates preliminary results as a word trellis index. A word 2-gram is used for recognition efficiency, and cross-word context dependency is either appoximately handled or ignored by using intra-word context independent model. The second pass is performed in the reverse direction using the word trellis index for word prediction and heuristic information. The language model is (reversed) word 3-gram and full cross-word dependency is handled. Finally, the best sentence hypothesis is output as a result. It can also output N-best candidates.

Various types of acoustic models can be used in Julius so that developers can choose

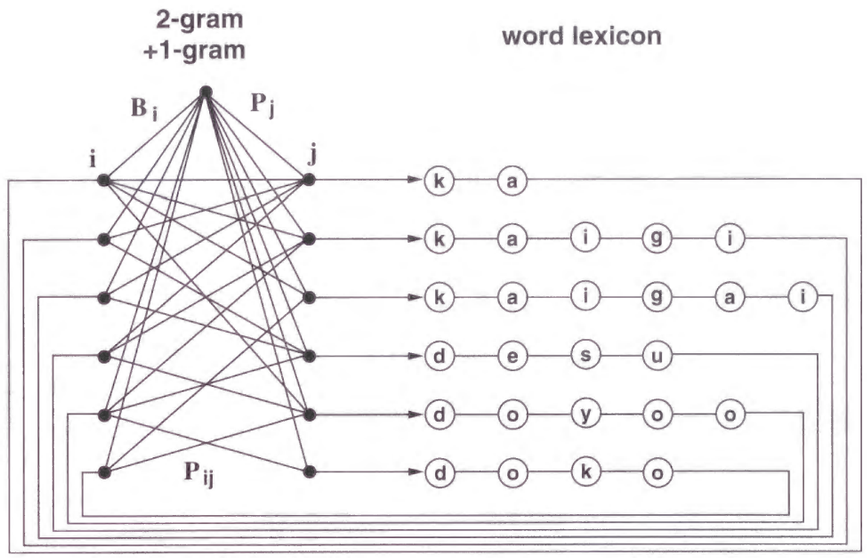


Figure 2.14: Word lexicon with back-off 2-gram

them by their computational resources for their applications. The supported HMM type is a continuous Gaussian mixture model, and any number of models, states and mixtures are handled.

2.6.2 Tree Structured Lexicon

The basic implementation of applying word N-gram in a frame-synchronous beam search is to regard the words and their connections as a probabilistic finite-state automaton of phones and build a lexicon network containing all possible words to represent the search space. Bi-gram probabilities are assigned to the transition arcs from word ends to beginning of words as in Figure 2.14. Apparently, the network size grows proportional to the vocabulary size, and arcs between words are 2 powers of the vocabulary size, which increases enormously in large vocabulary.

Sharing the same prefixes among words is effective to decrease the number of nodes. Figure 2.15 shows the tree lexicon which shares prefix phones. Also, merging of the same phones will allow smaller beam width, as it suppresses the most resemble nodes. This node sharing has remarkable effect in large vocabulary and has been an essential technique in LVCSR systems. But as one node is shared by many words, the N-gram probability cannot be determined at the beginning of words and embedding the 2-gram probability

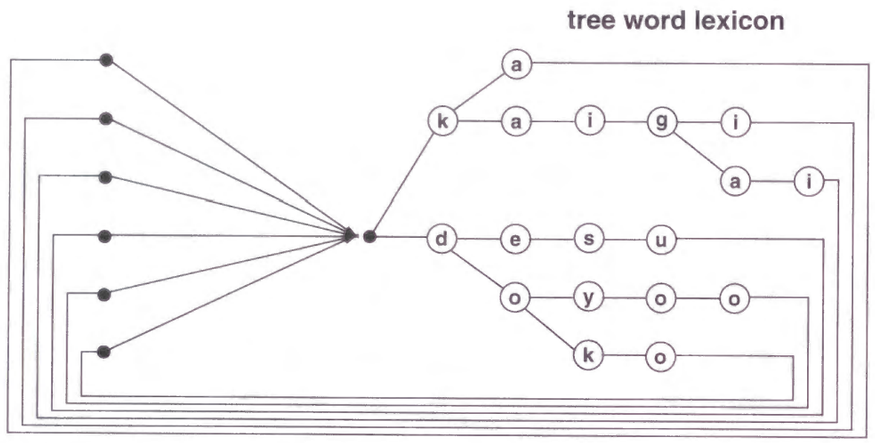


Figure 2.15: Word tree lexicon

into the lexicon network statically is impossible. Thus 2-gram values should be assigned directly while search. The actual value is determined when the search reaches leaf nodes (typically word ends).

There are some different implementations of tree lexicon. One is to build up a single static tree of all words beforehand, and the other is to expand dynamically for only necessary words while search[6]. To focus on the simple implementation, the static method is adopted in Julius.

2.6.3 Bi-gram Factoring

When using N-gram models together with a tree lexicon, the linguistic probabilities cannot be determined until the path reaches the end of words. This delay leads to the pruning errors, because the linguistic constraint cannot be incorporated to the score while in the middle of words, and the linguistically promising hypothesis can be falsely pruned. It is preferable to apply the constraint as early as possible.

N-gram factoring applies some approximate values of N-gram scores additionally in the middle of the words. We assign each shared node with the maximum N-gram probability of the possible words that shares the nodes and clear up the accumulated values as search goes to the word end[6][14]. Figure 2.16 shows how this N-gram factoring works.

In Julius, 2-gram factoring is mainly applied on the first pass. As the second pass performs stack decoding search and does not use the tree lexicon, the factoring is not

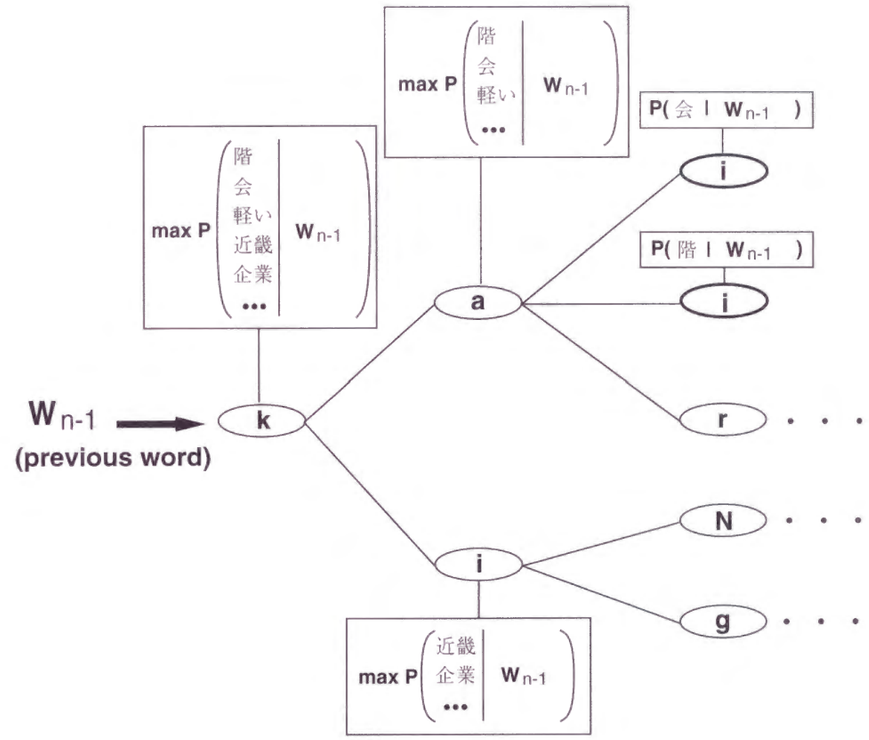


Figure 2.16: N-gram factoring

necessary.

2.6.4 Uni-gram Factoring

Although the 2-gram factoring improves the accuracy, the computational cost becomes large. The maximum likelihood of the sub tree should be computed on every branch of the tree lexicon. Actually, full 2-gram for all the words have to be accessed every time a new preceding word appears.

Using 1-gram probability for factoring instead of 2-gram can be considered. The 1-gram scores are attached at every node, and when the hypothesis reached the word end, the score is substituted by the 2-gram score. As the 1-gram score is independent on the preceding context, all factoring values can be computed statically beforehand.

Although this method remarkably reduces the computational cost of the language score, the optimality of language scoring on the first pass is violated and may cause accuracy loss in the final recognition result.

2.6.5 Language Model Score

As noted in the section 1.1.1, the hypothesis score in theory is the product (addition in log scale) of the linguistic probability and the acoustic probability of the word sequence. However, the actual dynamic range of linguistic probability is much smaller than that of acoustic model in most cases. The total recognition result tends to be dominated by the acoustic model and the effect of linguistic constraint is smaller than expected.

To balance both effects, language score is usually weighed by multiplying by a certain value. As it is well known that this language model score weighing is effective to the recognition accuracy, and Julius also adopts it.

2.6.6 Insertion Penalty

It often happens that a sequence of many short words falsely appear by local matching in N-gram based LVCSR. To suppress these insertion errors, adding some penalty score in every word transition is known to be effective and widely adopted.

Combined with the language score weight, the log score $f(h)$ for a partial hypothesis h consisting of n words $h = w_1, w_2, ..., w_n$ in Julius is defined as follows.

$$f(h) = \mathrm{AC}(h) + \mathrm{LM}(h) * \mathrm{LM\_WEIGHT} + n * \mathrm{PENALTY} \tag{2.9}$$

Where,

	$\mathrm{AC}(h)$	:	log likelihood of acoustic model for h
	$\mathrm{LM}(h)$	:	log likelihood of language model for h
	LM_WEIGHT	:	language model weight
	PENALTY	:	insertion penalty

2.6.7 Efficient Data Structure of Word N-gram

The number of parameters in word N-gram of large vocabulary is enormously large and consume extremely large amount of machine memory. For example, word 3-gram for 20k words should have probabilities for $20k^3 = 8 \times 10^{12}$ tuples. Actually the unseen N-tuples are extrapolated from $(N-1)$-tuples, but even with this back-off mechanism, we must hold millions of tuple data. Especially, our engine Julius uses two sets of N-gram, 2-gram and reverse 3-gram at the same time. Thus a compact data representation of N-gram models is required to realize LVCSR.

We adopt a data structure as described in Figure 2.17. The entries having the same context are stored in a linear list, and each has a link to upper structure. The 2-gram and reverse 3-gram are merged together to share the same 2-gram context. Apparently this data structure assumes that the 3-gram should have exactly the same (reverse) 2-gram tuple entries as the forward 2-gram. This condition is always satisfied if the two models are trained from the same text corpus with the same parameters such as cut-off and discount method.

2.6.8 Enveloped Best-First Search

In the best-first stack decoding on the second pass, there are some cases when the search fails. The best-first search is efficient if the heuristics is A*-admissible, but this cannot be always true in word N-gram based recognition because there is no guarantee that 2-gram score is always higher than the 3-gram score. When the 2-gram score in the first pass is much lower than 3-gram score for a certain word in the best sequence, the search becomes breadth-first on the point and does not proceed. Although this problem can be eased to some extent by applying larger weight on 3-gram than on 2-gram, failure of search is inevitable.

Frame-asynchronous beam search is one of the solutions to prevent such search failure and obtain some result at least, but it is not efficient compared with the best-first search in most cases.

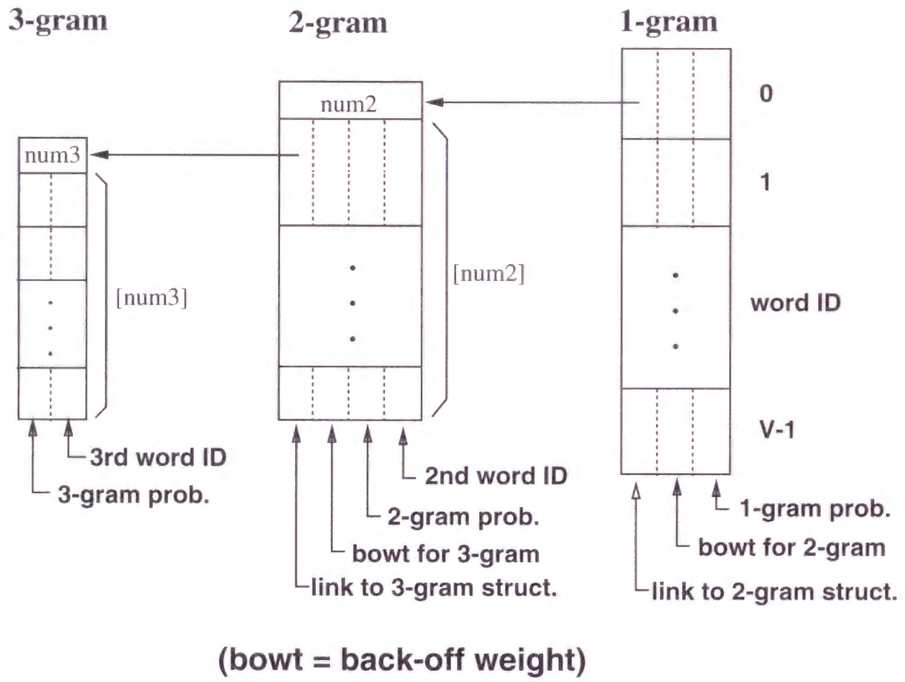


Figure 2.17: N-gram data structure

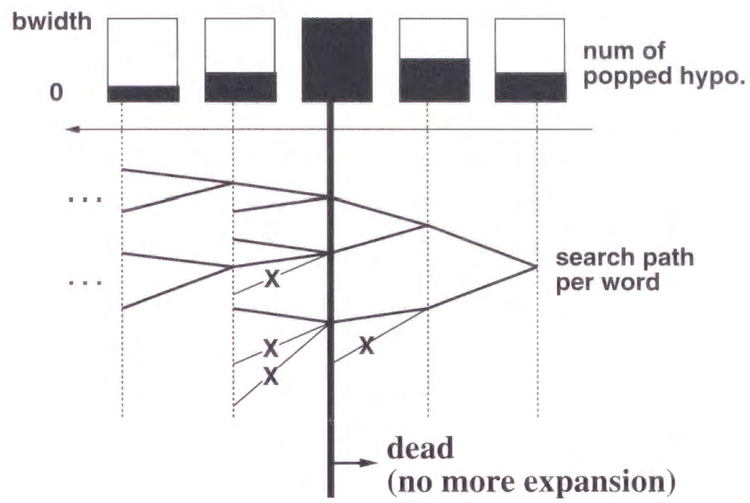


Figure 2.18: Enveloped best-first search

Toward this problem, we introduce a kind of "envelope", i.e. upper limit of the number of word expansion by the hypothesis length on the second pass (Figure 2.18). The total number of popped hypothesis are counted for its length, and when a counter of certain length reached the limit (bwidth in the figure), all the shorter hypotheses are treated as "dead" and dropped off from the hypothesis stack. This mechanism prevent best-first search from stopping at a certain point and make the recognition faster. It does not affect the original efficiency of best-first decoding, because it only activates when search is going to fail and does not take effect when a hypothesis is well guided by heuristics.

2.6.9 Improvements of Handling Inter-Word Context Dependency

Considering inter-word context dependency on the first pass is much expensive, since the models on the root and leaf node of the lexicon tree must be changed according to the connecting words. As the next word is unknown, the leaf node should have all the possible right-context variants. This results in significant computation cost.

In our multi-pass search strategy, we can handle only the word-internal part on the first pass and evaluate inter-word dependency on the later pass. Using different acoustic models on respective passes, however, result in mismatch in the scores and search failures.

We introduce an approximation method to handle the inter-word context dependency

Table 2.1: Search algorithms evaluated

first pass	intermediate form	
	word trellis index	word graph
word-pair	word-pair+trellis	word-pair+graph
1-best	1-best+trellis	

on the first pass. Instead of applying the exact triphone for different context, we use a single node that is assigned with the maximum score of all contexts. Specifically, acoustic scores of all the possible triphone variants are computed and only the maximum one is taken. Although the maximum score is not the exact score to be applied, this approximation using the maximum value conforms A* admissibility.

2.7 Experimental Evaluation

The proposed search algorithm is evaluated at 5,000-word dictation task. Note that, in this evaluation, only the basic search techniques are implemented and the following techniques are not used: unigram factoring, enveloped best-first search and approximation of inter-word context dependency on the first pass. Also, the inter-word context dependency is not handled even on the second pass for comparison.

Table 2.1 lists combination of methods evaluated. The baseline is a typical word graph method (word-pair+graph). All the methods in Table 2.1 are implemented in Julius for comparison.

2.7.1 Task and Database

The test condition is shown in the following paragraphs. All the models for evaluation are the results of the joint project with many researchers outside our laboratory. The training corpora and testsets are also common ones publicly provided for open evaluation.

Task The target is 5,000-word Japanese dictation of newspaper articles. As a large amount of electric corpus is available and the sentences have less ill-formedness, the target is suitable for a milestone of LVCSR study[1][15].

Table 2.2: Specification of language model

training set	45 months of newspaper article (from '91/01 to '94/09)
vocabulary size	5005
cutoff	1 for 2-gram, 2 for 3-gram
num of 2-gram tuple	578653
num of 3-gram tuple	1978931
smoothing	back-off (Witten-Bell discounting)
perplexity	70 (3-gram), 107 (2-gram)

Table 2.3: Specification of acoustic model

training set	read speech of 130 male speakers
num of sentences	about 25,000
total time	about 50 hours
parameter	MFCC (12 dim.) +ΔMFCC+ΔPow
CMS	applied by a sentence
num of phones	43
num of triphones	7921
states per triphone	3 (allow skip transition)
num of total HMM states	2110
output distribution	16 mixture density

Language Model Word 2-gram and reverse 3-gram are trained by the Mainichi newspaper articles of 45 months. The morpheme analysis results are provided by RWCP[1]. The specification of the models is shown in Table 2.2.

In the newspaper texts, unreadable characters (i.e. symbols, braces, etc.) or articles that are not sentence (i.e. weather forecast, financial tables, etc.) are deleted beforehand to avoid improper training.

Acoustic Model A speaker-independent, gender-dependent shared-state triphone HMM is trained by Phoneme-Balanced Continuous Speech corpus and Japanese Newspaper Article Sentences (JNAS) corpus collected by Acoustical Society of Japan[16]. Table 2.3 shows its specification.

[1]Real World Computing Partnership, Japan

Word Lexicon By considering variety of word pronunciation, we set up 7,451 baseform entries for 5,005 words.

Testset We used test set of 100 samples by 10 male speakers. The text is extracted randomly from the JNAS corpus, which is open to the LM training set. There is no unknown word in the testset. Examples are shown below.

Sample text in Japanese

十年前には，日本の産業社会には勢いがあった。
ヨルダンがシリアをこれほど厳しく批判するのは初めて。
一方，首相が新党構想を協議する両院議員総会の開催を提案。
ダイエーを選んだ理由は？
長野五輪に，日本は開催国として出場することができる。
受賞者十五人は次の通り。
これはどう見てもおかしい。
お隣の車のエンジンの音がする。
３０日２時半，３１日７時。東京芸術。

2.7.2 Evaluation Measure

Generally, the accuracy of recognition is measured by word accuracy, word %correct, and word %error. For each sentence, the recognition result and the correct transcription are DP-matched to determine which words are correct, and measures are computed by the following formula. Here N is the number of words in correct transcription, S is the number of substituted words, D is the number of deleted words and I is the number of falsely inserted words.

$$\text{word accuracy} = \frac{N-S-D-I}{N} \times 100 \tag{2.10}$$

$$\text{word \%correct} = \frac{N-S-D}{N} \times 100 \tag{2.11}$$

$$\text{word \%error} = 100 - \text{word accuracy} \tag{2.12}$$

$$\text{word \%improve} = \frac{\text{word \%error (after)}}{\text{word \%error (before)}} \tag{2.13}$$

The sentence tags and punctuations are ignored from evaluation.

2.7.3 Trellis vs. Word Graph

First, intermediate forms are compared. Both use word-pair approximation on the first pass. Word %error of trellis and word graph for various beam widths is plotted in Figure 2.19. Result on the first pass is also plotted here. The word trellis index achieves better accuracy (10.5% error) than word graph (12.0%) given enough beam width. This improvement is remarkable especially around short words, where the word-pair approximation may not be assumed. It suggests that the word-pair approximation includes some errors, but they are recoverable on the second pass by re-alignment with word trellis index, while the conventional word-graph cannot.

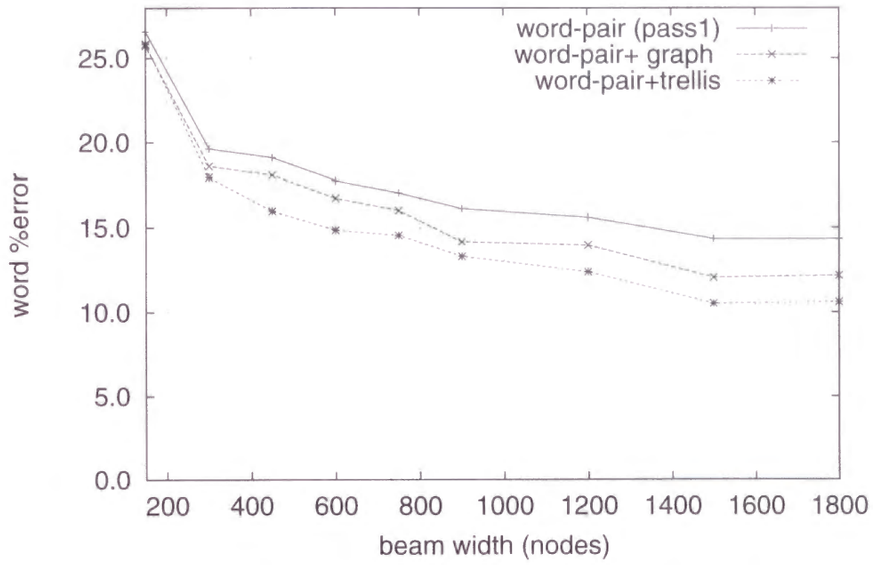


Figure 2.19: Comparison of trellis and graph (Word %Error)

2.7.4 Word-Pair vs. One-Best Approximation

Next, two N-best approximation methods are compared to examine how the errors influence on the final result. The results with word trellis index are shown in Figure 2.20. With one-best approximation together with tree lexicon, both approximation and factoring errors increase recognition failure on the first pass by 5.0%. However, they are recovered to 1.7% on the second pass, which is comparable to the baseline method. Thus, it is confirmed that even with simple one-best approximation, trellis index does not lose word boundaries that will make up the best sequence and finally gives as same accuracy as the word-pair approximation does.

In addition, when the beam width is relatively small, one-best approximation gets

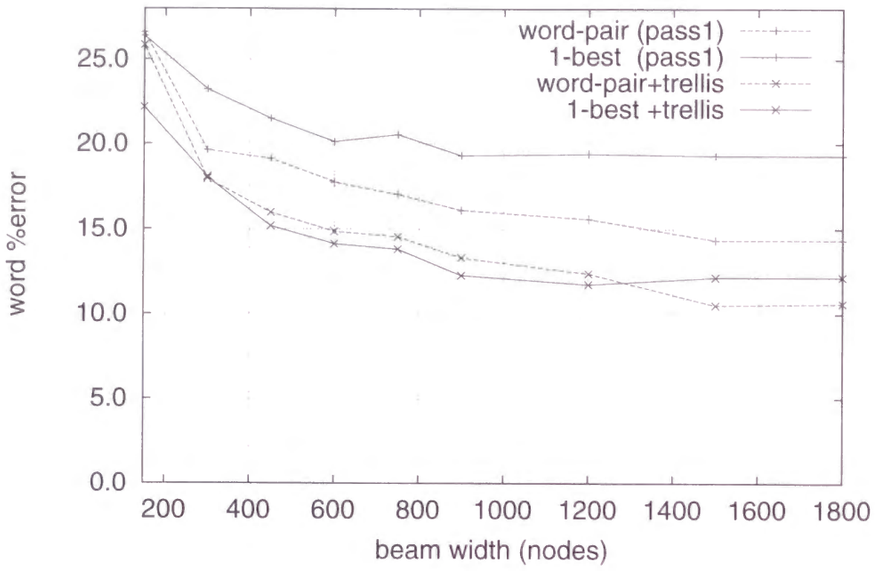


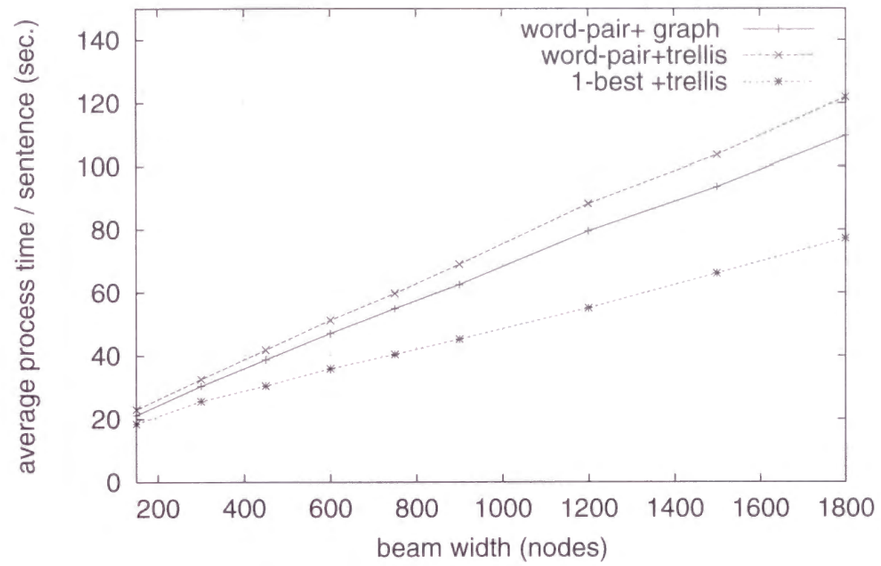
Figure 2.20: Comparison of word-pair and one-best (Word %Error)

better accuracy by 1%. This suggests that with word-pair approximation, same word hypotheses with different contexts occupy the beam, which are merged in word trellis index.

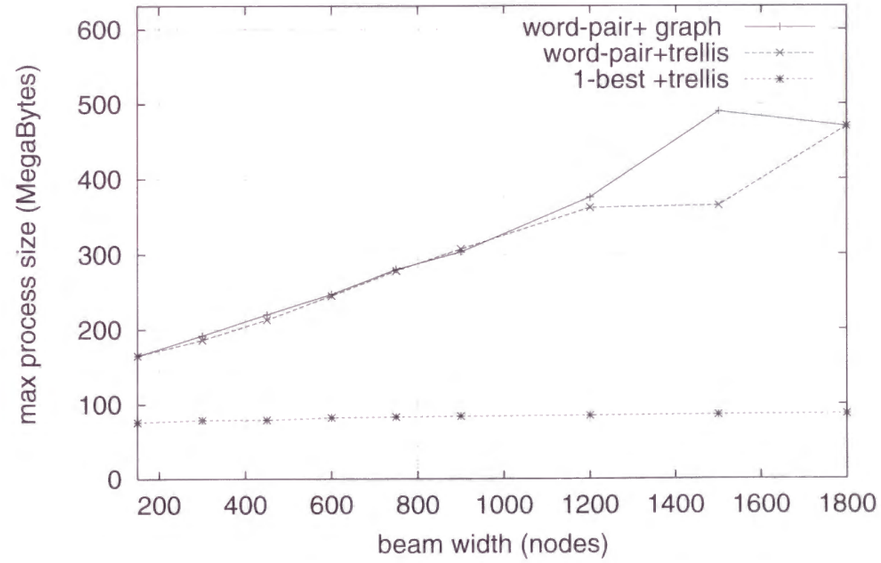
2.7.5 Efficiency Comparison

Next, we compare the algorithms with respect to efficiency. Figure 2.21 shows the average process time per sample and total memory size needed. Compared with the baseline (word-pair+graph), word trellis index (word-pair+trellis) needs trellis re-connection procedure for re-alignment on the second pass, but it costs a little. And the introduction of one-best approximation (one-best+trellis) significantly reduces computation cost. The average CPU time becomes almost 2/3, and the maximum workspace size needed for the search is reduced from over 400MB to nearly 30MB (plus 56MB for models). This difference comes from sharing a single lexicon on the first pass and will grow as vocabulary becomes large.

Thus, the proposed search method (word trellis index + one-best approximation) proved to be superior in that it performs far more efficiently while keeping high accuracy.



average process time per sample (sec.)



total memory size

CPU: 200MHz Ultra/SPARC

Figure 2.21: Computational cost

Table 2.4: Evaluation of search techniques

search method	word %error total (pass1)	avg. time (sec.)
(a) base	8.8 (21.1)	52.8
(b) +enveloped	8.0 (21.1)	49.9
(c) +IWCD1	6.5 (14.9)	89.5
(d) +1factor	8.8 (26.1)	35.7

CPU: UltraSPARC 300MHz

2.7.6 Improvement of Search Techniques

Next, several search techniques for further efficiency and accuracy are evaluated. Now the task is extended to 20k words. The language model is trained from 75 months of newspaper articles. Testset is extended to 100 samples by 23 male speakers. The baseline setting of search algorithm is the one proposed in the previous section (trellis + one-best), but some improvement and optimization are carried out on implementation.

Table 2.4 shows the breakdown. By setting envelope for each word length to limit the search range on the second pass as described in section 2.6.8, search failures were reduced from seven samples to only one, while not affecting the rest. The total search errors were reduced by 0.8% (b).

Introducing approximate handling of inter-word context dependency on the first pass (section 2.6.9) resulted in further significant error reduction (c). The accuracy of the first pass is greatly improved by 6.2% and the total result is also improved by 1.5%. On the other hand, computing triphones of maximum probability on every word end resulted in the large growth of computational cost.

By using 1-gram factoring instead of true 2-gram (section 2.6.4), the computational cost of language model and the total recognition speed are remarkably decreased (d). The errors were increased to 5.0% on the first pass, but the final degradation was only 0.8%. This fact also shows the robustness of word trellis index to the approximation errors on the preliminary pass.

2.8 Conclusion

A two-pass search algorithm with trellis interface for word N-gram language model is presented. The trellis form is extended to word trellis index that has the frame-indexed active word list and corresponding time-frame information to be applicable for LVCSR.

Compared with the conventional word graph method, word trellis index can recover the errors caused by the approximations. Thus, simple one-best approximation instead of word-pair approximation is sufficient to achieve almost the same accuracy while computational cost is remarkably reduced.

Many search techniques for speech recognition are incorporated to the recognition engine Julius, and several approximations and pruning methods are evaluated. All the adopted techniques successfully contributed to more accurate and faster recognition. The final system showed over 91% of word accuracy in 20k-word dictation task.

Chapter 3

LVCSR Algorithm using Finite State Grammar

3.1 Introduction

In chapter 2, we have studied large vocabulary continuous speech recognition based on word N-gram for general dictation system. The statistical language model is now currently pursued by many researchers[2][3][4], and is automatically trained by large corpus. On the other hand, a recognition system designed for a specific task needs an adequate language model that precisely represents its task domain, such as simple queries, transactions, and directory guidance system. Training such a specific statistical language model for every task is not easy, because collecting and transcribing thousands of utterances costs too much. Furthermore, there are cases in which statistical information has little meaning, such as names of place or product names. In such cases, using grammatical rules and vocabulary explicitly defined by hand is advantageous to the statistical approach, in that it can easily represent task-specific constraint and even switch the vocabulary for sub-tasks.

In this chapter, we focus on an efficient parsing algorithm for large vocabulary continuous speech recognition. Here parsing means searching for an optimal hypothesis that satisfies the given grammar. Although many parsing algorithms are developed so far, they have been evaluated mainly on tasks of hundreds or a thousand words at most[17][18][19], and not yet applied to larger vocabulary.

Simply expanding acceptable words results in the explosive increase of the hypothesis network in large vocabulary parsing. The lexicon used for the acoustic matching also grows proportional to the vocabulary size, and a large number of hypotheses should be

handled. Thus, the conventional one-pass beam search needs a large beam width and is not efficient.

To solve this problem and achieve an efficient parsing, we extend a two pass A* search method using word-pair constraint as heuristics[17] applicable to the large vocabulary. We modify the heuristic information calculated on the first pass to additionally provide information for preliminary pruning of unpromising words. The first pass uses category-pair constraint derived from the given grammar to compact the possible word hypothesis network. The second pass is the main parsing process using the preliminary results as a heuristics. Moreover, the following three methods are introduced for efficiency in large vocabulary: beam pruning on the first pass, per-category tree-structured lexicon, and preliminary candidate selection based on the first pass.

In the next section, the problems caused by the large vocabulary is described, and then the A* search method is compared with conventional one-pass beam search. Then we propose the extension of A* search to be applicable to large vocabulary, and present our recognition parser Julian. The proposed parsing algorithm is compared with the conventional method at recognition experiments in 5,000-word task.

3.2 LVCSR using Finite State Grammar

Continuous speech recognition is formulated as a search problem to find the best possible word sequence upon given linguistic and acoustic constraint. When using a written rule-based grammar, the search space is a definite word network generated according to the given grammar and vocabulary. When the grammar belongs to the regular expression class, the whole network can be statically represented by expanding all the rules beforehand. Otherwise, if the grammar is a context-free grammar, the network should be dynamically expanded while search only for necessary range.

3.2.1 Problems in Large Vocabulary Grammar

As the vocabulary gets larger, the mutual acoustic distance between words becomes close and many confusing words exist. Thus, the number of hypotheses that should be handled grows up. Especially in grammar-based recognition, there are several factors that make recognition very difficult. Their combined effect makes it even harder.


```
S:    MY [EVENT] NO PLAN GA ARU_V
S:    [EVENT] GA PP DE ARU_V
S:    TIME NI [EVENT] GA ARU_V
S:    PP DE [EVENT] GA ARU_V
PP:   [PLACE] TO PP
PP:   [PLACE]
```

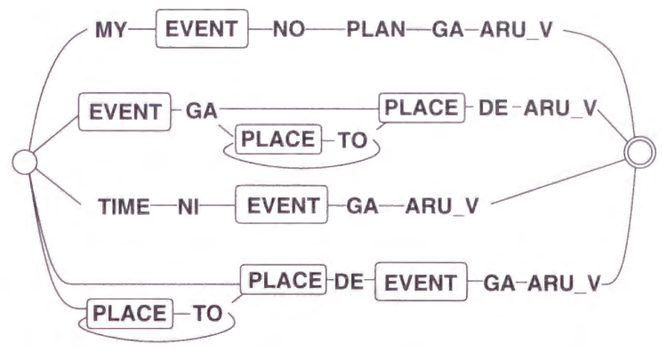


Figure 3.1: A typical recognition grammar and its expanded network

Redundancy in Network Definition Compared with word N-gram model that represents only the co-occurrence of neighborhood words, a ruled grammar expresses a longer and detailed constraint for whole sentence. As the occurrence of a word is determined by the sequence of all preceding words, the same words are likely to be generated in many locations depending on the various context, which results in the explosive expansion of word network. Figure 3.1 shows an example of a grammar and its expanded network. Note that each terminal symbol in the figure is not a word but a word category, and all words in the category will be expanded in the actual search. Especially assume that `EVENT` and `PLACE` categories contain thousands of words. They appear in different rules and are expanded according to respective rules. As a typical grammar-based large vocabulary recognition task often has a special category which contains huge number of the words, i.e. name of place or name of persons, expanding them everywhere in the network causes serious increase of network size, resulting in inefficient recognition.

Certainly, it is possible to write "optimal" rules that have no redundancy and will not generate such duplicated entries in the network. But writing such excellent rules is extremely hard in a complicated task. Moreover, in an actual development of a speech interface, the developer who writes the grammar for the desired task is not always an

expert who can optimize the rules by hand. Another way is to optimize the network automatically by a kind of automaton compiler, but the minimization of an automaton network is an NP-hard problem and the effect is still limited. Thus, the redundancy in the word network is a serious and inevitable problem in large vocabulary continuous speech recognition.

Problems in Tree Lexicon In actual recognition process, a word lexicon is used for acoustic matching in which words are listed with their baseforms. As it grows proportional to the vocabulary size, a tree-organized lexicon in which words share the same prefix phones is effective to reduce the size. This tree lexicon is currently adopted in most LVCSR systems with word N-gram language model[20].

Tree lexicon, however, makes it impossible to embed the linguistic constraint in a statistical network, since word entries cannot be identified at the beginning of words. For word N-gram based recognition, a typical solution is to introduce N-gram factoring as explained in section 2.6.3, or use partial lexicon depending on the preceding context, either statically or dynamically. But for deterministic grammar-based recognition, such probabilistic factoring does not work, and generating many lexicons depending on all preceding contexts is impractical.

Insufficient Grammatical Prediction When the vocabulary size becomes large, the word prediction by grammatical constraint becomes insufficient. A grammar can determine the next word deterministically as compared with a word N-gram, but since words in one or two categories almost occupy the whole vocabulary in many cases, such as address names or product names, the grammar's ability of word prediction becomes close to mere vocabulary constraint and relatively small for large vocabulary.

3.2.2 One-Pass Beam Search

The most popular search method for continuous speech recognition is one-pass beam search[21][18], where hypotheses are expanded simultaneously by the time frame and pruned by their temporal scores. As its performance is greatly affected by the local noise or similar hypotheses, the beam width must be wide enough to keep the optimal result.

3.2.3 A* Search

A* search is a kind of best-first search. A hypothesis is evaluated by summing the score of already generated path and a heuristic estimate of un-searched portion, and the one that has the best estimated score is always expanded. The estimated score $\hat{f}(n)$ for a partial hypothesis n is defined as followings,

$$\hat{f}(n) = g(n) + \hat{h}(n) \tag{3.1}$$

where $g(n)$ is the determined score of already expanded portion in log scale, and $\hat{h}(n)$ is the estimated score. To guarantee the search to be able to obtain the optimum result finally, $\hat{h}(n)$ must not exceed the actual score $h(n)$.

$$|\hat{h}(n)| \leq |h(n)| \tag{3.2}$$

When this condition is satisfied for all possible n, the search is called "A*-admissible". The search performance depends on the heuristic value $\hat{h}(n)$. The closer the value becomes to the true $h(n)$, the faster the optimum result is found.

One implementation of this A* search in speech recognition is a tree-trellis search[12], where $\hat{h}(n)$ is stored as a trellis form computed by acoustic matching to the input on reverse direction beforehand. The main search roughly estimates the scores of un-searched input by connecting the reverse trellis to every hypothesis. Especially, using word-pair constraint as the heuristics is effective[17]. The word-pair constraint consists of all words and their connectivity between them. It is a reduced constraint extracted from the given grammar, and satisfies the A*-admissibility. The constraint was proved to be a better heuristics than simple word-conjunction or phoneme-conjunction constraint.

This A* search was proved to be superior to one-pass search in small and middle vocabulary[17][22], but has not been realized in large vocabulary task because of its computational cost. Although the optimal result is guaranteed to be found if given a proper heuristics, all heuristic scores should be kept for any possible hypothesis to realize a true A*-admissible search. It is hardly possible in large vocabulary.

3.3 A* Search Algorithm for LVCSR

The process of heuristic calculation in A* search has the same problem of the growing computational cost as that of one-pass beam search. Unlike one-pass beam search, how-

ever, A* search is a multi-pass (two pass) method so that some errors and approximations are allowed on the first pass, which is used only for heuristic information on the later pass.

Toward these problems, we propose several methods to realize the accurate A* search in an efficient way. The main idea is to regard the first pass as not only a heuristic calculation but also a preliminary selection of word candidates by acoustic matching with simple and compact linguistic constraint. The second pass performs a search by referring to the preliminary results as both heuristic information and word prediction. By keeping not all the word hypotheses but only the promising ones, the proposed search method does not strictly satisfy the A*-admissibility, but keeping sufficient candidates will not affect practical recognition performance.

The original idea is based on the word trellis index as shown in chapter 2, although the implementation is designed to be suitable for grammar-based parsing. The concrete solutions for each problem are explained in the following.

3.3.1 Network Bundling by Category-Pair Constraint

Using word-pair constraint significantly reduces the network size. Instead of the full grammar, the word-pair constraint is represented by listing all words in parallel and connecting word ends whose transition is allowed in any grammar rules. With its simple static loop, the multiplication of grammar nodes is bundled to the compact network, and such redundancy is not involved. Since the network is statically defined and no duplication appears, the beam width can be narrower and computational cost can be remarkably reduced compared with the one-pass search method. Although this constraint is subset of the full grammar, the results are not the final result and another recognition process will be performed again with the full grammar using the results as heuristic information.

The word-pair constraint, or a category-pair constraint here equivalently, can be extracted from any grammar by only checking their connectivity. The size of category-pair network is at most proportional to the vocabulary size.

This network compaction is considered as a kind of hypothesis bundling in search. This so-called "bundle search" was proposed by Ito et al[23] in phone level, and by Watanabe et al[24] in word level. Both methods gather the same words or phones that appear near around the frame in different contexts, and share the same acoustic scores to avoid multiple acoustic matching. Compared with the proposed word-pair method, they require some special mechanism integrated to search procedure, and causes some approximation

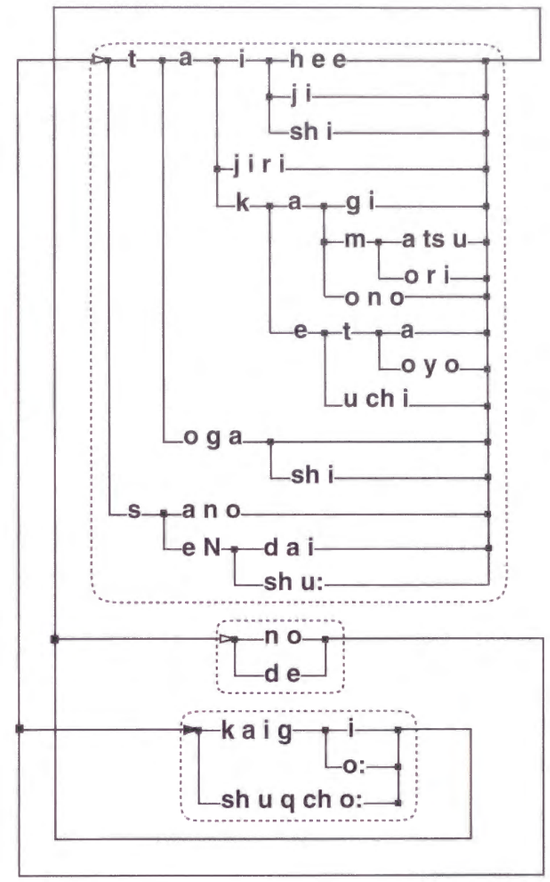


Figure 3.2: Tree-structured category-pair network

errors by re-using single acoustic scores. The proposed A* search method can get more precise results finally by performing another search.

3.3.2 Lexicon Tree Organized by Category

Instead of building a single tree lexicon, we build sub-tree for each category to represent the whole static word-pair network more compactly. Figure 3.2 shows the category-pair network with tree organized lexicon. As the grammar rules are normally represented with category levels, making independent trees for categories does not affect the grammatical constraint, while the network size can be reduced remarkably.

The same word-pair constraint can be applied with a single tree by dynamically expanding necessary part of the tree according to the preceding word[10]. But since dynamic

expansion of lexicon tree results in substantial overhead of computation, preparing static sub-trees is advantageous.

3.3.3 Word Prediction by Preliminary Acoustic Matching

As limiting the possible words by grammatical rules alone is not sufficient in large vocabulary, it is effective to utilize the pre-computed acoustic information by the first pass for pruning. We propose an intermediate form that keeps the indexes of survived words within beam for each frame in addition to the basic trellis information. When expanding words on the second search, only the words that are both listed in the index and conform the grammar are expanded. This word prediction by preliminary acoustic matching suppresses the number of hypotheses effectively, especially when the grammar constraint does not work, i.e., for a very large category that contains names of place or names of product.

3.3.4 Parsing Algorithm

The proposed A* parsing algorithm consists of two passes as illustrated in Figure 3.3. The first pass performs a left-to-right frame-synchronous beam search using category-pair constraint for the entire input, and stores the word-end trellis survived within the beam with their scores and their indexes for time frames. The second pass performs a stack decoding with the original grammar in right-to-left direction. Expansion of only the words in the index of the preliminary pass are allowed, and their word-pair scores of the preliminary pass are used as the estimated scores of un-searched area.

The concrete algorithm is described below. Here $W_n = w_n, w_{n-1}, ..., w_1$ is a partial sentence hypothesis of n words (w_1 is the end), T is the input frame length. The forward (right-to-left on the second pass) likelihood of W_n from time T to t is denoted as $g(W_n, t)$, the backward (left-to-right) likelihood of word w which ends in time t on the first pass is as $h(w, t)$, and the total score of W_n is as $f(W_n)$. And the word index stored after the first pass of time t is described as $index(t)$. Note that the search is performed in reverse (right-to-left) direction:

1. Generate initial sentence hypotheses W_1 consisting of one word w_1 which can grammatically appear at the end of sentence. The initial score is given as below, and

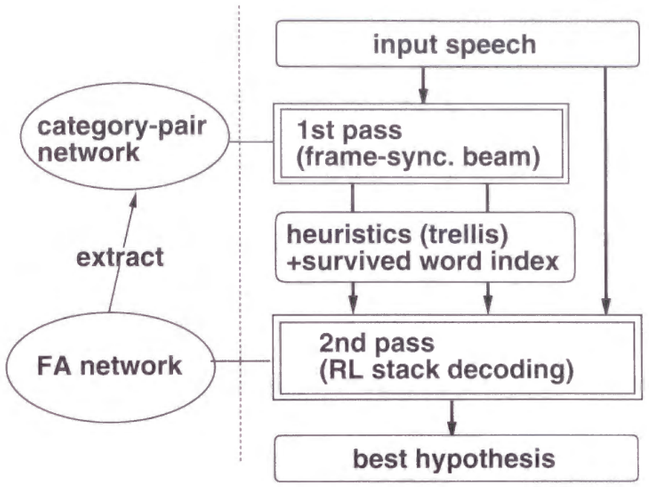


Figure 3.3: Overview of parsing algorithm

they are pushed to the stack.

$$f(W_1) = h(w_1, T) \tag{3.3}$$

2. Pop the best sentence hypothesis W_n from the stack. If it is acceptable by the grammar and reaches the beginning of input, output it as a result and finish the search.

3. Expand the forward HMM trellis for the last word w_n.

$$g(W_n, t) = \begin{cases} \max_{t'} \{\beta(w_n, t, t') + g(W_{n-1}, t')\} & \text{if } n > 1 \\ \beta(w_n, t, T) & \text{if } n = 1 \end{cases} \tag{3.4}$$

Here $\beta(w_n, t, t')$ is the forward likelihood expanded from time t' to t.

4. Among the words which can follow W_n grammatically, those that included in the trellis index are connected to generate hypotheses W_{n+1}. The new scores are calculated as follows:

$$f(W_{n+1}) = \max_{t: w_{n+1} \in index(t)} \{h(w_{n+1}, t) + g(W_n, t)\} \tag{3.5}$$

5. Push all generated W_{n+1} to the stack. Return to 2.

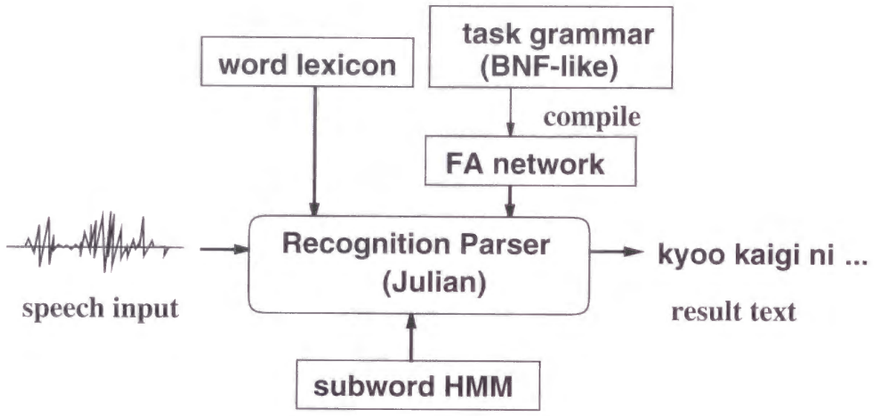


Figure 3.4: Recognition system based on CSR parser Julian

3.4 A Portable CSR Parser Julian

We have implemented the proposed search algorithm as a continuous speech recognition parser for large vocabulary task. Its name is Julian, aimed at efficient and portable recognition applicable for any applications defined by grammars. Anyone can construct his own grammar-based recognition system by preparing a grammar and acoustic HMM model. Figure 3.4 describes an overview of the recognition system based on Julian.

A grammar and a word lexicon for Julian should be prepared separately. The grammar is written in a BNF form, in which the terminal symbols are word categories. The lexicon is a list of words classified by the category block, together with their baseform entries. The grammar is then converted to a finite automaton network by a compiler designed for Julian. As the original BNF form allows writing a grammar of context-free class, whether the written grammar belongs within the finite automaton (regular expression) class or not is checked in this compilation stage. It can handle only right-recursive rules. The category-pair constraint for the first pass is extracted automatically by Julian itself at the initialization.

Julian can use various types of acoustic models, so developers can choose them by their computational resources for their applications. The format of acoustic model is an HTK[25] HMM definition format. The supported HMM type is a continuous Gaussian mixture model, and any number of models, states and mixtures are handled. It can also uses context-dependent models such as triphone HMM.

- There is a Speech Committee tomorrow from two to three on laboratory 2.
- I want to change the place of budget conference on next Friday to small seminar room.
- When will today's seminar start?
- Then, please set the beginning time to 16 o'clock.

Figure 3.5: Example sentences

3.5 Experimental Evaluation

The proposed A* search method is evaluated by comparing with standard one-pass beam search. The developed engine Julian is compared by the HTK recognition decoder "HVite". As they can use exactly the same acoustic model and equivalent grammars, their accuracy and efficiency of search algorithms can be compared fairly through the recognition experiments.

3.5.1 Task and Database

The task domain is a personal schedule management, and the recognition target is a well-formed sentences about registration, deletion, modification and query of the schedule. Test set is prepared from 50 sentences uttered by eight person, 400 utterances in total. The average length is 3.2 seconds and 6.2 words per sentence. The example sentences are shown in Figure 3.5. Note that the actual text and data are all in Japanese.

Two types of grammar are prepared for evaluation. One is a phrase grammar named PG, which allows repetition of any phrases in a sentence. The other is a sentence-level grammar named SG, in which each phrase must reside in their fixed place in a sentence.

To test the impact of vocabulary size on recognition performance, two vocabularies of different sizes are prepared for both grammars. PG1 and SG1 have a middle vocabulary gathered by hand. PG2 and SG2 have a large vocabulary that is extended semi-automatically from PG1 and SG1 by adding noun and verbs (especially place names and time nouns) extracted from ordinary dictionary.

Thus, there are four grammars in total. Table 3.1 shows their specifications. The FA nodes in the table are the number of category nodes after converted to the finite-state automaton network. Julian and HTK have their own grammar compilers that

Table 3.1: Specification of task grammars

grammar	lexicon size	word perp.	FA nodes Julian	FA nodes HTK
PG1	806	91.6	257	280
PG2	4439	257.1	257	280
SG1	833	28.7	5061	2849
SG2	5023	76.0	5061	2849

FA nodes: number of category nodes after converted to FA

convert a given grammar to a word (category) network for recognition. Although SG1 and SG2 represent stronger constraints and have lower perplexity than PG1 and PG2, their complicated network cannot be easily optimized and the network nodes get significantly larger.

The acoustic model is either a context-independent monophone model or context-dependent shared-state triphone model. Their specification is the same as described in section 2.7.1.

3.5.2 Evaluation Measure

As criteria to compare search methods, we use recognition rate for accuracy and beam width for cost efficiency. Here beam width is defined as the average number of HMM states per frame calculated in Viterbi operation in search.

For HTK decoder, the beam width is given by the number of phone models in a frame and it is easily converted to the number of states. Actually, the all acoustic models used here consist of 3 states. Thus three times the given beam width is the compared value.

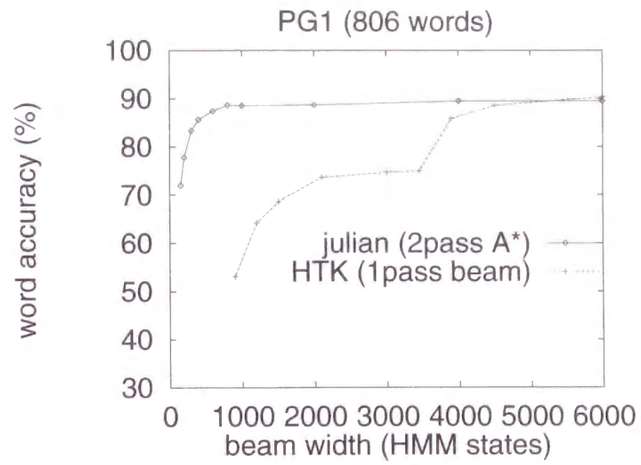
For Julian, where search is performed in two passes, we define the beam width $\hat{b}$ to be calculated in the following equation:

$$\hat{b} = b_1 + \hat{b_2} \tag{3.6}$$

Here, b_1 is the normal beam width on the first pass of heuristic calculation, and $\hat{b_2}$ is the virtual beam width, the total computed HMM trellis averaged per frame on the second pass.

$$\hat{b_2} = \frac{npop}{nword} \times avg\_state \tag{3.7}$$

Here, *npop* is the total number of popped hypotheses, and *nword* is the length of the final sentence hypothesis, and *avg_state* is the average number of states per word.



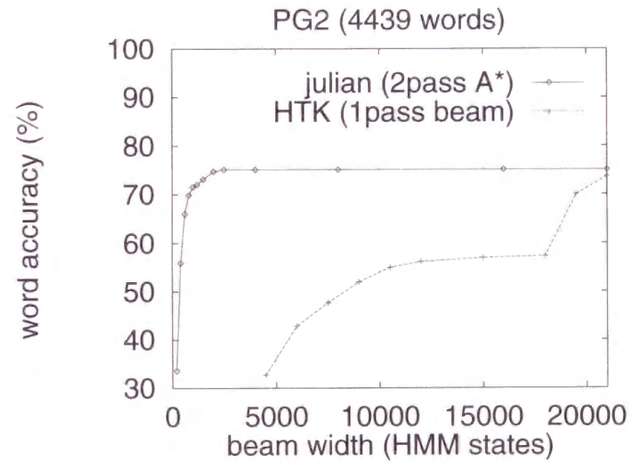


Figure 3.6: Comparison of search algorithms

3.5.3 A* Search vs. One-Pass Beam Search

The word recognition accuracy of PG1 and PG2 for the two algorithms in various beam widths are shown in Figure 3.6. Monophone HMM is used here.

Julian achieves the equivalent accuracy of HTK decoder with a remarkably small beam width. In the typical one-pass beam search decoder like HTK, the word network grows explosively large in proportional to the perplexity, and requires a very large beam width to get optimum result. On the other hand, in the two-pass decoder Julian, the category-pair constraint on the first pass suppress the network to very small size, and allows narrower beam width. Furthermore, the computational cost of the second pass is much smaller

Table 3.2: Recognition accuracy and average process time

grammar	word acc. / time (sec.)	
	Julian	HTK
PG1	88.6 / 3.2 (1000)	88.6 / 10.5 (4500)
PG2	74.7 / 6.2 (2000)	73.7 / 44.2 (21000)
SG1	97.1 / 3.4 (600)	— (—)
SG2	91.4 / 7.0 (1500)	— (—)

CPU: UltraSPARC 300MHz
real time: 3.2 sec.
embraced number is beam width

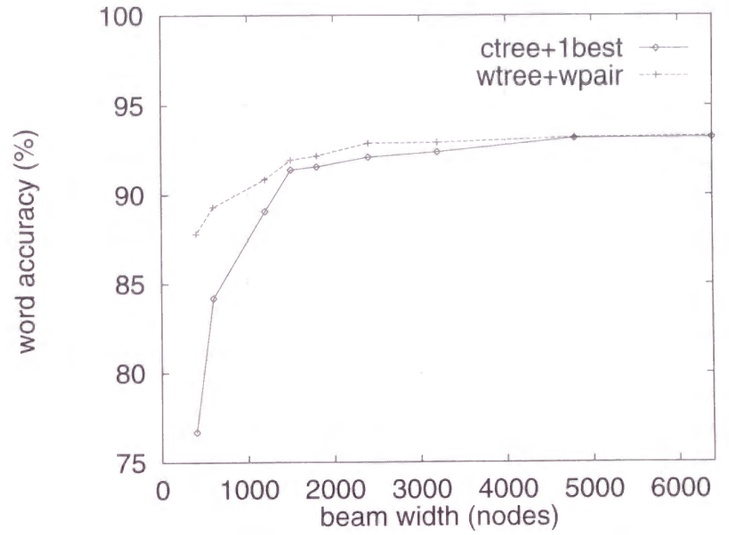
than the first pass. As the heuristics works well on the second pass, the final result was obtained best-first in most samples. The virtual beam width of the second pass $\hat{b}_2$ is only from 24 to 26, which can be neglected compared with the first pass.

When applied to SG1 and SG2, HTK decoder did not work because of the network explosion. Whereas Julian stably works and realizes word accuracy of 97.1% in SG1 and 91.4% in SG2. The accuracies and average recognition times for all grammars are summarized in Table 3.2. Although it is hard to generate a compact grammar network from BNF description in general, the word-pair constraint can be easily and feasibly extracted from any grammar independent of their complexity, which results in the stability and high performance of Julian.

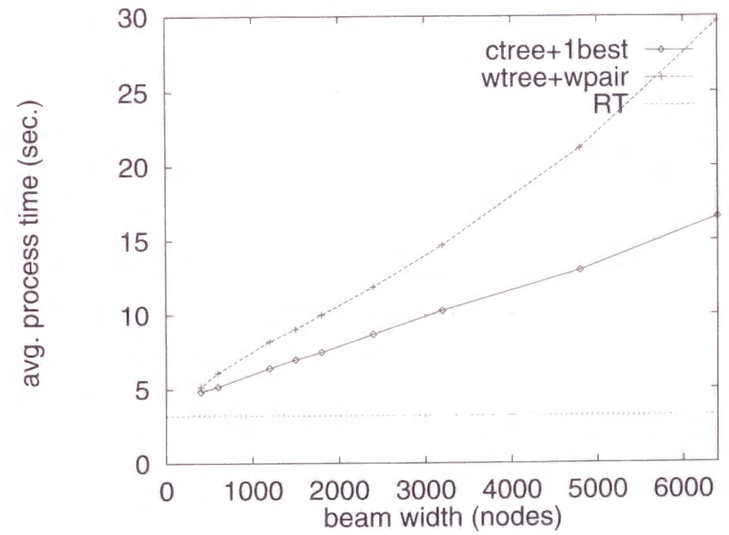
With respect to the recognition speed, Julian costs significantly less time to get the same accuracy. The results can be obtained within a real time in SG1, and 2.2 times the real time in SG2.

3.5.4 Static Category-Pair Network vs. Dynamic Expansion

Next, we compare the two methods of applying the category-pair constraint into search. One is the static representation with making sub-trees per category, and the other is the dynamic expansion with single tree lexicon and the constraint. The results are shown in Figure 3.7. the `ctree+1best` is the static one, and `wtree+wpair` is the dynamic one. Given enough beam width, their difference in accuracy is less than 1%, which is not a



Word Accuracy



Average Time

CPU: UltraSPARC 300MHz
Grammar: SG2

Figure 3.7: Comparison of static and dynamic expansion of category-pair network

Table 3.3: Performance with context-dependent model

	monophone	triphone
word acc. (%)	91.4	93.2
avg. time (sec.)	7.0	20.0
memory used (MB)	12	29

CPU: UltraSPARC 300MHz
Grammar: SG2

significant difference. In tree-organizing the word lexicon, the number of total phone nodes is reduced to 70.0% by the former category-level tree lexicon, and to 51.3% by the latter single tree. As the dynamic one has less nodes, its accuracy is less decreased even in a small beam.

The average computational time for the category-tree static method is two thirds of the single-tree dynamic method. As the dynamic expansion apparently costs more and saturated accuracy is comparable, the static network is superior in that peak accuracy can be achieved with much less processing time[4][26].

3.5.5 Effect of Context-Dependent Model

Finally, acoustic models are changed to context-dependent triphone HMM instead of monophone. The recognition results are compared in Table 3.3. Recognition errors are reduced by 21% relatively, and the final accuracy reaches 93.2%. On the other hand, a large-scale triphone model needs much memory and computational cost. The process time grows by almost three times.

3.6 Conclusion

An efficient A* search is realized and evaluated in large vocabulary continuous speech recognition based on a finite-automaton grammar. The proposed search algorithm uses category-pair constraint as heuristics and consists of the following two passes.

1. First pass: an efficient decoding with compact category-pair constraint derived from the original grammar

2. Second pass: an accurate best-first search using the preliminary results as heuristics and full parsing with the given grammar

Experimental results of the 5,000-word recognition task show that the proposed method is superior to the conventional one-pass beam search in both accuracy and computational cost. Tests on many grammars of different complexity and vocabulary sizes show that the advantage becomes greater in larger vocabulary. The category-pair constraint is stably extracted and robustly applied from any scale or complexity of given grammars. Our developed recognition parser Julian achieved the accuracy of 91.4% in 2.2 times the real time at the 5,000-word task, and 97.1% in almost real time at the 1,000-word task.

Chapter 4

Efficient LVCSR using Phonetic Tied-Mixture HMM

4.1 Introduction

Sharing HMM parameters for accurate and efficient acoustic modeling has been a major concern in large vocabulary continuous speech recognition systems. A typical triphone model has thousands of states and hundreds of thousands of Gaussian distributions. Estimation of such a large number of parameters requires huge amount of training data to obtain the desired accuracy. Thus, sharing of the model parameters in various levels are widely adopted to reduce the number of total parameters.

The current dominant approach of parameter tying is to share the states among certain clusters of triphone model according to acoustic similarity. This is called a shared-state triphone model[27]. The state clusters are defined by either top-down similarity rules of its contexts or bottom-up acoustic similarity given by their output probability distributions. Another approach is a tied-mixture (TM) system where a single set of Gaussian distributions is shared by all HMMs while each state has different mixture weights. There is also an extended form of the TM model called phonetic tied mixture (PTM) HMM[28][29], where a set of Gaussian components is defined independently for each phone and the triphone variants of the same base phone share the corresponding Gaussian set.

The TM and PTM HMM are advantageous to the state-clustered triphone in that overlapping mixture distributions on different states are properly modeled with less Gaussians and the training can be more reliable. In conventional TM and PTM HMM, however, as all Gaussian distributions in different states (within a phone in PTM) should be covered by one codebook, the size often gets very large to the extent of hundreds or thousands. It

is not easy to train such a large mixture to an optimal point. In this chapter we propose a PTM model of another parameter sharing scheme, a *phonetic state-based* tied mixture model that realizes both easy training and reliable estimation.

Another merit of TM over the shared state triphone model is that since it has a larger codebook with less redundant distributions, it is easier to introduce pruning mechanism in Gaussian mixture computation. Therefore, several pruning methods are also proposed and compared.

4.2 Parameter Sharing in Acoustic Models

First, the parameter sharing scheme of the shared-state triphone, TM, and PTM models are compared on how they represent the entire acoustic space. For clear discussions, the output probability type in the following discussion is assumed to be a mixture of continuous density distributions, where an output probability is composed by a weighed linear combination of multivariate Gaussian distributions of diagonal co-variances.

A triphone model defines separate states for each context-dependent phone variant in order to represent the cross-phone articulation effects precisely. Since many context-dependent states have their own mixtures, the number of acoustic parameters to be trained becomes large proportional to the number of states. As it is hard to prepare a sufficient amount of training set that covers all the possible triphone variants, sharing the states among acoustically similar ones is usually adopted[27][30].

But this state-level sharing of acoustic distributions is not necessarily efficient. Even if the states are determined to be separated (unshared) by having differences between their entire acoustic probabilities, those states may still have overlapping distributions among them. It means there may be many redundant Gaussians defined for the same overlapping distributions. Such redundancy on parameter definition requires training data for separate states, and even more data as a whole, thus makes parameter estimation unreliable.

In TM HMM, on the other hand, a set of a large number of mixture components is shared among all states while each states has different mixture weights. A whole acoustic space is modeled with a larger mixture unit, and the overlapping mixture components are well represented with less parameters.

However, TM models have not demonstrated as good performance as the shared-state triphone model. The total number of mixture components of TM model is usually smaller

by a magnitude, thus it cannot have enough discriminative ability. The root cause is that it is not easy to train or estimate a large-scale codebook of distributions as a whole.

This problem also arises in a conventional PTM model. In conventional PTM, a rather small codebook is defined for each phone and shared among all states in triphone variants of the same base phone. As the acoustic space to be modeled on each codebook is rather narrow, the codebook size can be smaller than TM models. However, training many codebooks that consists of hundreds of mixtures to optimal point for each phone is still not easy.

4.3 A Phonetic Tied-Mixture Model based on Monophone Model

Based on these viewpoints, we propose a new PTM model that shares Gaussian components of monophone states. A set of Gaussians are trained independently for states of each position of each base phone as codebooks, and assigned to states of the same topological position in their triphone variants. Each state has different weights for the assigned codebook.

This proposed model is an inter-mediate of the shared-state triphone and the conventional TM. It is more efficient than shared-state triphone and allows larger-scale model than conventional TM. Compared with the shared-state triphone model, the redundant Gaussians in overlapping space are merged by tying the Gaussian set of each state among triphone variants. Compared with conventional PTM model, the Gaussian distributions is modeled more accurately by having independent mixture components on different position.

Especially, the codebooks can be derived straight-forwardly from monophone mixtures. As the EM algorithm can be used to train the monophone mixture, larger codebooks can be reliably estimated by gradually increasing their numbers of components than conventional TM models.

4.3.1 Definition of Model

To make differences clear, computation of output probability is explained for the three models: shared-state triphone, TM, and proposed PTM. In basic shared-state triphone,

an output probability of state S_j for input vector O is calculated as show below.

$$b(O|S_j) = \sum_{m=1}^{M_j} w_{jm} \cdot P(O|G_{jm}) \quad (4.1)$$

Here M_j is the number of mixtures, G_{jm} is the m-th Gaussian distribution assigned to state S_j, and w_{jm} is the corresponding weights. $P(O|G_{jm})$ is the output probability of Gaussian G_{jm} for input vector O.

In a TM model, a set of Gaussian distributions is defined as a codebook, and all states refer to them with their own weights. Assuming C as the codebook and N is the total number of Gaussians, the output probability for each state S_j is calculated as:

$$C = \{G_i : 1 \leq i \leq N\} \quad (4.2)$$

$$b(O|S_j) = \sum_{i=1}^{N} w_{ji} \cdot P(O|G_i) \quad (4.3)$$

In actual, only mixture components whose weights are large enough are selected for computation, instead of all N components.

In the proposed PTM, a codebook is defined for each state position of each phone. In other words, among triphone variants of the same base phone p, the l-th states share the codebook C_{pl}. Here assume phone(j) as the base phone of the triphone in which S_j belongs, and position(j) as the position of S_j in the phone, then the output probability is calculated as:

$$p = \text{phone}(j) \quad (4.4)$$

$$l = \text{position}(j) \quad (4.5)$$

$$C_{pl} = \{G_{plm} : 0 \leq m < N_{pl}\} \quad (4.6)$$

$$b(O|S_j) = \sum_{m=1}^{N_{pl}} w_{jm} \cdot P(O|G_{plm}) \quad (4.7)$$

N_{pl} is the number of Gaussians in the codebook C_{pl}, assigned to the l-th position of triphone variants of base phone p. Note that we assume here that the triphones of the same base phone has the same number of states.

4.3.2 Synthesizing Monophone and Triphone Models

The proposed PTM HMM is synthesized from state-clustered triphones and a monophone model as illustrated in Figure 4.1. First, the whole state sets of state-clustered triphone

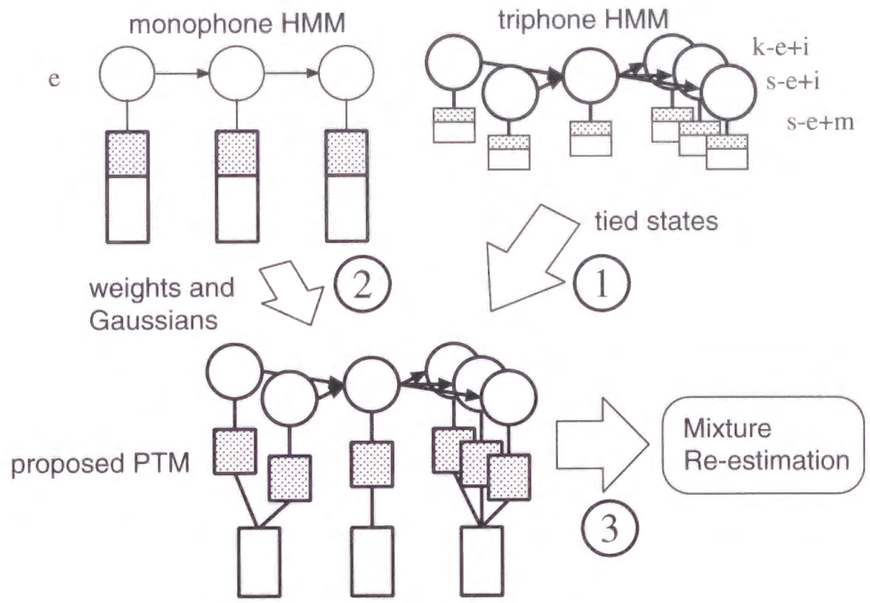


Figure 4.1: Building proposed PTM HMM

HMM are copied while stripping off the mixture definitions. Their state sharing structure is kept unchanged. Next, for each state, the Gaussian distribution set of the corresponding phone and state position from monophone HMM is assigned and shared. Tied triphone states also share both the same mixture components and weights, and non-tied states share only the mixture components and have different mixture weights. Finally, the overall mixtures and weights are re-estimated for optimization by maximum likelihood estimation.

The construction process is straightforward and easy. Mixtures are estimated by monophone models by gradually increasing their numbers of components. This makes it easy to reliably estimate a large number of mixtures. Context-dependent modeling is realized by the state-clusters of the conventional triphones. Here, corresponding mixture components are substantially assigned by different weighting among a large codebook that efficiently covers the whole acoustic space.

4.3.3 Training Procedure

Actual steps for building this model is as follows:

1. Train shared-state triphone HMMs. The mixture components of this model are used

only for determining state-sharing, so a single Gaussian is enough. Any method can be used to determine the state sharing, but the number of states in a phone must be the same as the monophone on step 2, and only states of the same topological position within a phone are allowed to be shared. In this work, top-down decision tree clustering is adopted.

2. Train monophone HMMs that have a large Gaussian mixture for each state. No parameter tying is done here.

3. Extract the state definitions (state names and transitions) from the shared-state HMMs in step 1. For each state, information of its position in the triphone and the base phone to which it belongs is kept for later synthesis.

4. Extract the mixture definitions (Gaussian sets and weights) from the monophone HMMs in step 2. For each set of Gaussians and weights, the corresponding phone and location in monophone is kept for later synthesis.

5. For each state, assign the extracted Gaussian set from the corresponding state of monophone HMM.

6. Re-estimate the mixture components and weights to discriminate triphones. The mixture itself is also re-trained.

4.4 Gaussian Pruning

Acoustic matching often occupies the largest part of processing time in current recognition systems, because a large number of Gaussian distributions must be computed. So reducing the cost is significant for fast decoding.

It is obvious that computing only the Gaussians of high probabilities is effective for reducing the acoustic computation cost. Actually, among all Gaussians within a mixture, those near the input vector have dominant effect on the output probability, and others do not affect the final result. So computing only the best Gaussians for each given input effectively reduces the computation cost. Especially, such a pruning mechanism works more effectively with PTM than tied-state triphone HMM, since state probabilities can be expressed by smaller number of larger codebooks.

Gaussian pruning is one approach to achieve the selection of Gaussians, by determining which ones are not promising dynamically in computing the mixtures. The basic concept of this pruning method has been proposed and implemented in a rather simple way[29, 31]. In this study, we investigate more efficient implementation to realize the Gaussian pruning, and evaluate them by comparison through recognition experiments.

4.4.1 Calculation of Output Probability in PTM

Given a Gaussian distribution G with mean vector μ, covariance matrix Σ and number of vector dimension d, its output probability for an input vector O is calculated by the following equation.

$$P(O|G) = \frac{1}{\sqrt{(2\pi)^d|\Sigma|}} e^{-\frac{1}{2}(O-\mu)^t\Sigma^{-1}(O-\mu)} \tag{4.8}$$

As we assume Σ to be diagonal in this study, it is equivalent to compute the following.

$$Q(O|G) = -\frac{d\log 2\pi + \log|\Sigma|}{2} - \frac{1}{2}\sum_{v=1}^{d}\frac{(o_v-\mu_v)^2}{\sigma_v} \tag{4.9}$$

Here $Q(O|G)$ is the log likelihood of $P(O|G)$, μ_v is individual vector component of μ and σ_v is the corresponding diagonal component of Σ.

Using $q(O|G,v)$ as the partial accumulation of the log likelihood of G up to dimension v, the following recurrence formula is derived.

$$\begin{aligned} q(O|G,0) &= -\frac{d\log 2\pi + \log|\Sigma|}{2} \\ q(O|G,v) &= q(O|G,v-1) - \frac{(o_v-\mu_v)^2}{2\sigma_v} \\ Q(O|G) &= q(O|G,d) \end{aligned} \tag{4.10}$$

Equation 4.10 shows that $q(O|G,v)$ decreases in monotony as calculation goes. So, if the log probability of a Gaussian become below a threshold, its final value is guaranteed to be below the threshold. Gaussian pruning uses this feature to determine which Gaussian can be neglected.

To apply Gaussian pruning in PTM models, all existing codebooks C_{pl} are pre-computed for each frame in the following steps.

1. Set a threshold value.

2. Calculate probabilities of all Gaussian G_{plm} $(1 \le m \le N_{pl})$ in the codebook by Equation 4.10. For each Gaussian, if $q(O|G_{plm},v)$ becomes below the threshold at a dimension v, calculation is terminated and it is discarded.

After the probabilities of all codebooks are computed, the acoustic probability of state S_j is calculated by Equation 4.7, using its own weights and skipping the pruned Gaussians.

4.4.2 Pruning Methods

Several methods to determine the threshold in Gaussian pruning are investigated. The purpose is to get k-best Gaussians out of a mixture while cutting off as much computations of other Gaussians as possible.

a) k-best safe pruning Use the value of the temporary k-th best Gaussian as the pruning threshold. A Gaussian is pruned if the accumulated distance reaches under the threshold during computing each distance component. If it is not pruned to the last dimension, it means that the value is within the k-best, so update the k-best threshold. The pruning condition is defined as the following (K means the set of the current k-best Gaussians):

$$q(O|G,v) < \min_{G' \in K} Q(O|G') \tag{4.11}$$

b) k-best safe pruning, previous best Same as above but the k-best Gaussians of the previous time frame are computed first. As input vectors change gradually in successive frames, we can expect that the best Gaussians in the previous frame get higher scores. This makes the initial threshold closer to the true k-best value.

c) k-best with heuristic estimation In computing a Gaussian, its expected value is estimated by adding the temporary maximum values of the yet-to-be-computed dimensions to the current accumulated distance. Pruning is performed by the estimated score. The initial maximum score on each dimension is set up by computing the previous k-best Gaussian set $prevK$ first. The estimated value from dimension v' to the last is computed as:

$$\hat{q}(O|G,v') = -\frac{1}{2}\sum_{v=v'}^{d} \max_{G' \in prevK} \left\{ \frac{(o_{tv} - \mu_{mv})^2}{\sigma_{mv}} \right\} \tag{4.12}$$

Then the pruning condition is defined using the $\hat{q}(O|G,v')$:

$$q(O|G,v) + \hat{q}(O|G,v+1) < \min_{G' \in K} Q(O|G') \tag{4.13}$$

That is,

$$q(O|G,v) < \min_{G' \in K} Q(O|G') - \hat{q}(O|G,v+1) \tag{4.14}$$

d) beam pruning Instead of Gaussian's final score, an independent threshold is set up for each dimension. The thresholds are defined by subtracting a certain offset (corresponding to a beam width) from the maximum of $q(O|G,v)$. First, compute the previous k-best Gaussians and get the maximum for each dimension. A Gaussian is pruned if its accumulated score up to the dimension is below the offset range. The pruning condition is defined as:

$$q(O|G,v) < \max_{G' \in prevK} q(O|G',v) - \text{offset} \tag{4.15}$$

The former two thresholds are *safe* in that the accurate k-best Gaussians are guaranteed to be obtained without errors. The latter two are *unsafe*, rather aggressive methods where k-best Gaussians can be lost in the computation process by mis-leading heuristics or using a too small offset.

4.5 Experimental Evaluation

4.5.1 Task and Database

Accuracy and efficiency of the proposed PTM model are evaluated through recognition experiments. We build gender-independent PTM HMM from 64 mixture monophone HMM and 3000 state shared-state triphone HMM.

The task is 20k-word dictation of Japanese newspaper articles[16] with a word 3-gram model. The acoustic model is integrated with Julius, our two-pass decoder based on best-first search[32]. All decoding techniques described in chapter 2 are adopted to get the best performance. The reference models are gender-independent tied-state triphone models. They are all of 2,000 states but different in number of mixture components per state. These modules are all available in Japanese dictation toolkit[33]. The proposed PTM is trained by exactly the same corpus as the reference models.

The specification of the models is shown in Table 4.1. The shared-state triphone has dedicated mixtures for different states, while the PTM has 129 codebooks of 64 Gaussians for 43 phones and 3 states for each. The total number of Gaussians of PTM is almost equal to the 4 mixture shared-state triphone.

Test-set contains 200 sentences spoken by 46 speakers, equal number of male and female. Recognition is run on Sun Solaris workstation with UltraSPARC 300MHz.

Table 4.1: Specification of acoustic HMM

model	states	codebooks	Gaussians per codebook	total num. of Gaussians
triphone, 2000×16	2000	←	16	32000
triphone, 2000× 8	2000	←	8	16000
triphone, 2000× 4	2000	←	4	8000
PTM	3000	129	64	8256

Table 4.2: Comparison of models

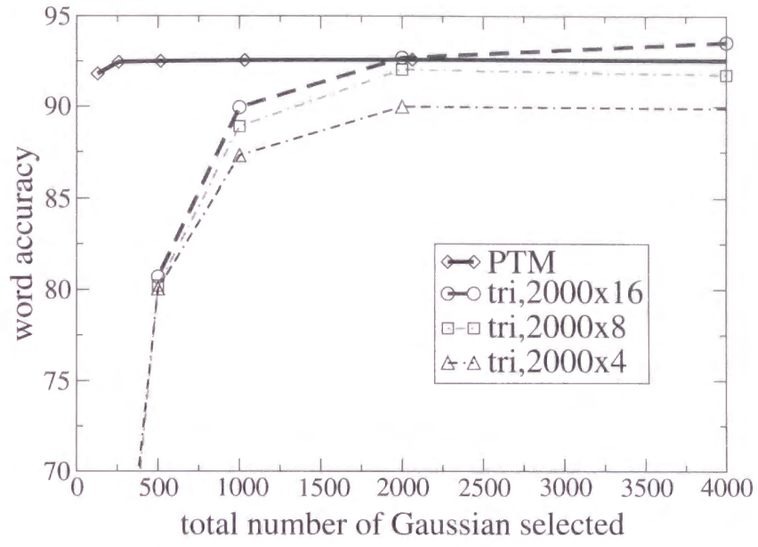
model	word accuracy
triphone, 2000×16	93.8
triphone, 2000× 8	92.7
triphone, 2000× 4	90.8
PTM, synthesized	92.3
PTM, re-trained	93.0

beam width: 1500

4.5.2 Comparison of Models

Word accuracy of the PTM model and triphone models is compared in Table 4.2. The proposed PTM achieves higher accuracy than the shared-state triphone of the same complexity (i.e. total number of Gaussian), and is comparable to the triphone of four times as many mixture components. The figure is almost best for the test-set. Thus it is proved that the proposed PTM is superior to the triphone in that the same accuracy can be achieved with less parameters. The "PTM, synthesized" in Table 4.2 is a model that re-estimates only the mixture weights and does not re-train the mixture components. On the other hand, the "PTM, re-trained" model re-estimates both. The latter achieves better accuracy, thus it is shown that re-estimating not only mixture weights but also the Gaussian distributions is effective.

Next, we examine how these models are affected by Gaussian pruning. Accuracy of the models against various numbers of selected Gaussians is plotted in Figure 4.2. For comparison, Gaussians are pruned independently for each codebook in PTM and across all Gaussians in triphone models. A smaller beam width is used in decoding for convenience. The applied pruning method is "k-best safe, previous-best", which cause no pruning error.



(beam width = 800)

Figure 4.2: Accuracy decrease by Gaussian pruning

The proposed PTM keeps high accuracy even when the number of selected Gaussians is limited to only 3% (2-best out of 64 mixture components in a mixture, 258 in total). On the other hand, triphone systems are obviously sensitive to Gaussian pruning. Over 2000 Gaussians per frame are required to get sufficient accuracy. Our state-based PTM efficiently covers the while acoustic space, and makes severe pruning possible.

Thus, our PTM model is proved to be both more accurate and efficient than the shared-state triphone models.

4.5.3 Comparison of Gaussian Pruning Methods

Next, we evaluate the performance of Gaussian pruning methods to reduce the total computation cost. In this comparison, the purpose is to select the best 2 Gaussians out of 64 mixture Gaussians with cutting computation of others as much as possible. The computational cost is measured by the total percentage of computed Gaussian distance components (100% means no pruning). The number of vector dimension is 25.

In Table 4.3, both computed amount and word accuracy for the proposed methods are listed. The first safe pruning method reduces the computed Gaussian distances to 59%. By setting an initial threshold by the previous k-best Gaussians, the ratio is improved

Table 4.3: Comparison of Gaussian pruning methods

pruning methods	Gaussian distance components computed	word accuracy
a) k-best safe	59 %	92.5
b) k-best safe, previous-best	52 %	92.5
c) k-best with heuristics	36 %	92.3
d) beam pruning	21 %	92.2

beam width: 800, 2-best selected

to 52%. As pruning error never occurs in these methods, they are simple and reasonable ways to reduce the acoustic matching cost without decreasing any accuracy.

Using the heuristic estimation reduces the cost further to 36% with a little pruning error. The beam pruning method realizes the best performance. The computed densities are remarkably reduced to 21% with only a little loss of accuracy.

Furthermore, the effect of offset value in beam pruning is examined. Figure 4.3 shows the reduction rate of computed Gaussian distance components and recognition accuracy. The computation cost is stably reduced in proportion to the offset value. The maximum reduction reaches 11% when the beam offset is set to 0, but the accuracy is greatly spoiled. By giving a sufficient offset, the accuracy is recovered to the original level. Thus, it is proved that setting a proper offset to the threshold is significant in beam pruning. Based on these results, an offset value of 2.5 was adopted in d) of Table 4.3.

4.6 Monophone Lexicon Tree on Preliminary Recognition

As the proposed PTM HMM is built using mixtures of monophone HMM, it is possible to re-define monophone models based on triphones by sharing the same mixture components. As monophone is context-independent, we can make a smaller lexical tree and omit handling context dependency. So we explore the possibility of using a monophone tree lexicon at the preliminary recognition in our multi-pass decoder for further efficiency.

The result is shown in Table 4.4. The compaction of the lexicon tree does not improve the recognition speed, and the increased errors on the preliminary pass affect the accuracy on the final result. The results confirmed that the use of better acoustic model on the first pass is significant.

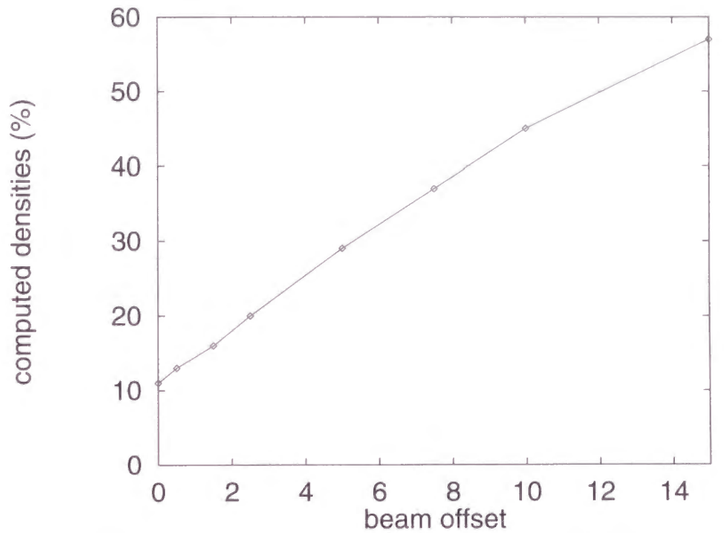
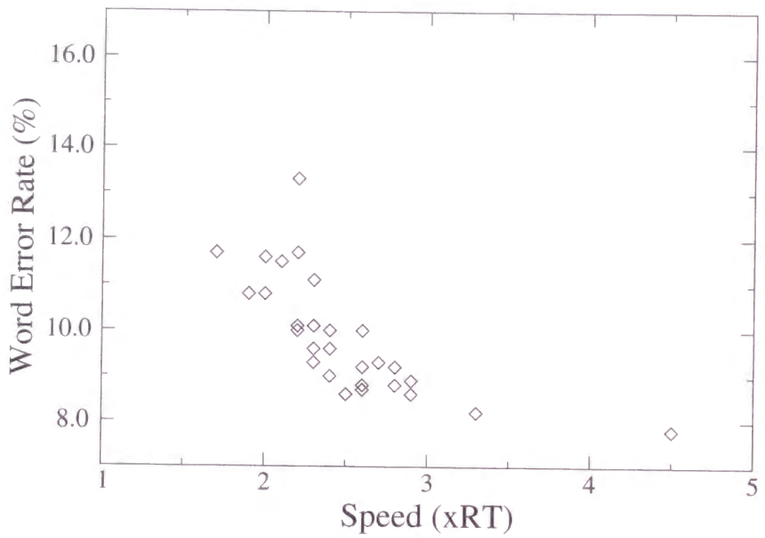


Figure 4.3: Beam pruning offset vs. performance

Table 4.4: Lexicon tree: triphone vs. monophone

lexicon	state #	accuracy (acc1)	time (×RT)
triphone	173251	92.2 (82.2)	4.5
monophone	128188	90.2 (76.8)	4.4

acc1: accuracy on the preliminary pass
CPU: UltraSPARC 300MHz



CPU: UltraSPARC 300MHz

Figure 4.4: System performance

4.7 System Performance

Finally, we pursue the maximum recognition performance with our proposed PTM. The average processing time in real time factor and the word error rate with various search parameter settings (beam width and so on) is plotted in Figure 4.4. By tuning search parameters and using a smaller beam width, accuracy of 91.4% is achieved with a speed of 2.5 times the real time.

4.8 Conclusion

A new PTM model with state-based mixture-tying scheme has been introduced. The model is synthesized from mixtures of a monophone model and state-clusters of triphone models. As construction of the model is straightforward, training of the parameters can be more reliable.

This model achieves a word error rate of 7.0%, which is comparable to the best figure by the triphone of much higher complexity. With Gaussian pruning, computing only 2 out of 64 mixtures per state does not cause any loss of accuracy. The acoustic computational cost is reduced to about 20% by the beam pruning method.

Chapter 5

Japanese Dictation System

5.1 Introduction

Large Vocabulary Continuous Speech Recognition (LVCSR) that we have studied in this thesis is a basis of speech technology applications in the next generation including a voice-input word processor and dictation of broadcast programs or personal audio tapes. Its component technologies can also be used in various applications such as spoken dialogue interfaces.

In order to build an LVCSR system, high-accuracy acoustic models, large-scale language models and an efficient recognition program (decoder) are essential[34][35][5] [7]. Integration of these components and adaptation techniques for real-world environment are also needed. In order to promote both research of various component technologies and development of such complex systems, we have recognized the necessity of a common platform.

We have adopted Mainichi Newspaper, one of the nation-wide general newspapers in Japan, for the sharable corpus of both text and speech[16], and organized a project to develop a standard software repository that includes acoustic and language models and recognition programs[33]. The three-year project (1997-2000), funded by the IPA (Information-technology Promotion Agency), Japan, is a collaboration of researchers of different academic institutes. The software repository as the product of the project is available to the public. The overview of the corpus and software mentioned here is depicted in Figure 5.1.

The specifications of acoustic models, language models and recognition engine are described in this chapter. We also report evaluation of each module under 20K-word and 60K-word Japanese dictation task.

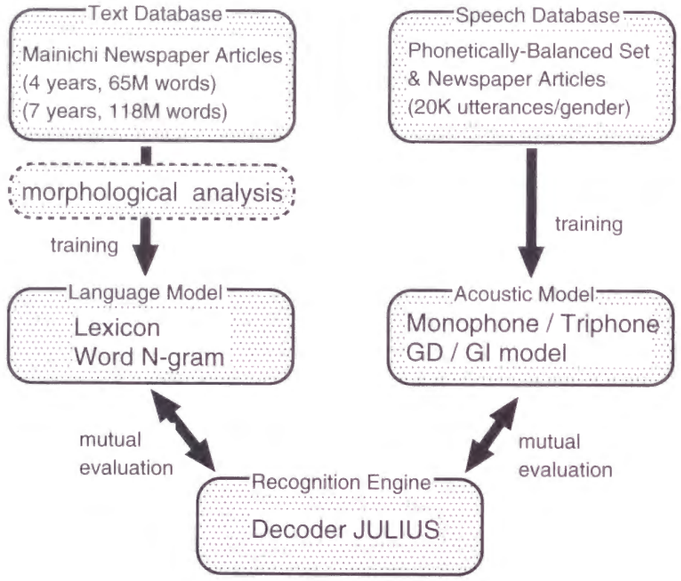


Figure 5.1: Platform of LVCSR

5.2 Specification of Models and Programs

5.2.1 Acoustic Model

Acoustic models are based on continuous density HMM. They are available in the HTK format[25].

We have trained several kinds of Japanese acoustic models from a context-independent phone model to triphone models, as listed in Table 5.1. Gender-dependent models are prepared for each model type, and the typical ones also has gender-independent models. The phonetic tied-mixture models as described in chapter 4 is also included for efficient decoding. Users can choose an adequate model according to the purpose. A simpler model realizes faster recognition at the expense of accuracy degradation.

The set of 43 Japanese phones are listed in Table 5.2. The phone notation is defined by Acoustical Society of Japan (ASJ) committee on speech database. Here, the symbols `a:`∼`o:` stand for long vowels and the symbol `q` for a double consonant. Three pause models, `silB`, `silE` and `sp`, are introduced for pauses at the beginning, at the end of utterances and between words, respectively.

The acoustic models are trained with ASJ speech databases of phonetically balanced sentences (ASJ-PB) and newspaper article texts (ASJ-JNAS). In total, around 20K sentences uttered by 132 speakers are available for each gender.

Table 5.1: List of acoustic models

model	#state	#mixture	gender
monophone	129	4, 8, 16	GD, GI
triphone 1000	1000	4, 8, 16	GD
triphone 2000	2000	4, 8, 16	GD, GI
triphone 3000	3000	4, 8, 16	GD
PTM triphone	3000/129	64	GD, GI

Table 5.2: List of Japanese phones

```
a i u e o a: i: u: e: o: N w y
p py t k ky b by d dy g gy ts ch
m my n ny h hy f s sh z j r ry
q sp silB silE
```

The speech data were sampled at 16kHz and 16bit. Twelfth-order mel-frequency cepstral coefficients (MFCC) are computed every 10ms. The difference of the coefficients (Δ MFCC) and power (Δ LogPow) are also incorporated. So the pattern vector at each frame consists of 25 (=12+12+1) variables. Cepstral mean normalization (CMN) is performed on whole utterances to offset the channel mis-match.

Each phone model consists of three states excluding the initial and final states that have no distributions. The state transitions are all left-to-right, and the path from the initial state and that to the final state are limited to one.

When a triphone model is applied to CSR, it has to cover all possible combinations of the phones. Thus, an extra file is used to define the mapping from the possible tuples (logical triphone) to prepared models (physical triphone). In actual, there are not sufficient data for all logical triphones. So the decision tree-based clustering is performed to build physical triphones that group similar contexts and can be trained with reasonable data. By changing the threshold of clustering, we set up a variety of models whose number of the states is 1000, 2000 and 3000, respectively.

The PTM model is built from Gaussian sets derived from monophones of 64 mixtures and state set from triphone of 3000 states. The triphone states share the Gaussian sets of the corresponding state in monophone, and only has different weights for each Gaussian components. The Gaussian sets and weights are then re-trained for optimization. This model effectively represent context-dependency of phones, while the training procedure is

Table 5.3: Lexical coverage

vocabulary size	coverage
5000	88.3%
20000	96.4%
24000	97.0%
53000	99.0%
60000	99.2%
101000	99.7%
154000	99.9%

stable and reliable.

5.2.2 Lexicon

A lexicon is a set of lexical entries specified with their notations and baseforms. It is provided in the HTK format[25].

The lexicon is consistent with both the acoustic model and the language model. The phone symbols used in the baseforms are covered with the acoustic model. For each lexical entry, its probability is given by the language model (at least 1-gram).

The vocabulary consists of the most frequent words (=morphs) in Mainichi newspaper articles from Jan. 1991 to Sep. 1994 (45 months) [16]. In Japanese, lexical entries are mainly defined by a morphological analyzer that segments undelimited texts. In this toolkit, we adopt the morphological analysis tool called "Chasen"[36]. Not a few lexical entries have multiple baseform entries because Japanese Kanji usually has multiple pronunciations. The lexicon also includes entries of comma, period and question marks that are re-written as a pause in pronunciation.

The pronunciations are based on the NHK[1] Japanese pronunciation and stress dictionary. A post-processing tool called "ChaWan" is also included to correct the ambiguity of the pronunciations. In Japanese, numerals has various baseform depending on the context and following counter suffix often varies to voiced consonant. This ambiguity is handled by ChaWan after processed by Chasen.

Generally, the morphs of different parts of speech have different tendency of possible adjacent words, even if they are same in notation. In order to improve language modeling, we distinguish lexical entries by not only their notations but also their morphological

[1]Japan Broadcasting Corporation

Table 5.4: Specification of 20K N-gram

	2-gram entries	3-gram entries
45month cutoff-1-1	1,238,929	4,733,916
45month cutoff-4-4	657,759	1,593,020
45month compress10%	1,238,929	473,176
75month cutoff-1-1	1,675,803	7,445,209
75month cutoff-4-4	901,475	2,629,605
75month compress10%	1,675,803	744,438

Table 5.5: Specification of 60K N-gram

	2-gram entries	3-gram entries
75month cutoff-1-1	2,420,231	8,368,507
75month compress10%	2,420,231	836,852

attributes (part-of-speech tags). The lexical coverage of various vocabulary sizes is listed in Table 5.3. Finally, lexicons of 5K, 20K and 60K vocabulary size is available. The 60K lexicon realizes coverage of over 99%.

5.2.3 Language Model

N-gram language models are constructed based on the lexicon. Specifically, word 2-gram and 3-gram models are trained using back-off smoothing. Witten-Bell discounting method is used to compute back-off coefficients. The models are available in the CMU-Cambridge SLM toolkit format[37].

The comma, period and question marks are also included in the statistical language models. As a result, the occurrence of short pauses between words is estimated by the probabilities of these symbols that correspond to pauses.

The cut-off thresholds for the baseline N-gram entries are 1 for 2-gram and 2 for 3-gram. More compact models are also prepared for memory efficiency by setting higher cut-off thresholds (4). Moreover, entropy-based cut-off is tried by deleting entries successively to minimize the mutual entropy between before and after EM training[38]. By applying the method, a compact model that has only 10% of 3-gram entries are also prepared.

The training corpus (Mainichi newspaper '91/01-'94/09 and '95/01-'97/06) after pre-processing has 118M words (=morphs). Specifications of the resultant model for the 20K

Table 5.6: Overview of decoder Julius

	cross-word phone model	language model	search approx.
1st pass	approximate	2-gram	1-best
2nd pass	accurate	3-gram	N-best

and 60K lexicon are shown in Table 5.4 and Table 5.5. Each entry occupies 18 bytes for 2-gram and 6 bytes for 3-gram in our decoder. For the decoder that performs forward-backward search, the backward 3-gram model is trained.

5.2.4 Decoder

The recognition engine Julius[32] has been developed to interface the acoustic model and the language model. It can deal with various types of the models, thus can be used for their evaluation.

Various kinds of models are supported. Monophone and Triphone HMM with any number of mixtures, states, phones can be used. It can also handles tied-mixture models with Gaussian pruning. These model types are automatically identified. The maximum size of vocabulary is 65535 words.

Speech input by waveform file (16bit PCM) or patter vector file (HTK format) is supported. Live microphone input is also available on Linux, Sun and SGI workstations and DAT-Link/netaudio. Note that only the speech analysis that is needed by the acoustic models as described before is implemented.

Julius performs two-pass (forward-backward) search using word 2-gram and 3-gram on the respective passes. The included version in this toolkit integrates all search techniques described in chapter 2. The various search parameters can be determined in configuration file such as beam width, language weight, insertion penalty, word envelope width and so on. Overview of the decoder is summarized in Table 5.6.

5.3 Japanese Dictation System

By integrating the modules specified in the previous section, a Japanese dictation system has been designed and implemented.

The block digram of the system is illustrated in Figure 5.2. The acoustic model and language model are integrated based on the decoder specification. In the first pass, word

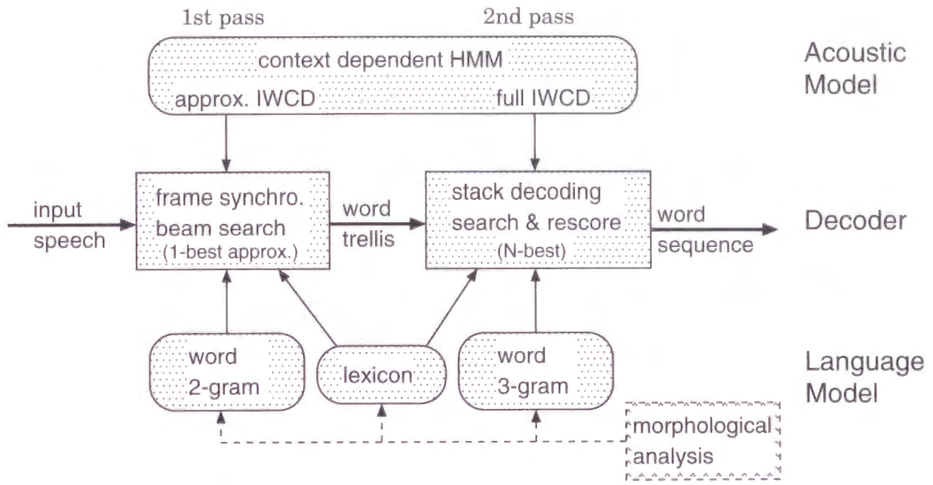


Figure 5.2: Block diagram of Japanese dictation system

2-gram is applied and inter-word phonetic context dependency (CD) is approximately handled. Word 3-gram and full inter-word context dependency, which are more precise and computationally expensive, are incorporated in the second pass to re-score and search on the reduced candidates.

Since there are several variations in both acoustic model and language model, we can design different systems accordingly. Typically, use of monophone model instead of context dependent model makes an efficiency-oriented system. Setting of decoding parameters such as the beam width may also yield variations of the system.

20K-word and 60K-word dictation system was developed. The components independently developed at different sites were successfully integrated.

5.4 Evaluation of Modules and Systems

The integrated system can be used to evaluate the component modules, in turn. By changing the modules, we can evaluate their effects with respect to the recognition accuracy and efficiency. These evaluations were executed on the 20K-word system.

For the evaluation, we have used a portion of the ASJ-JNAS speech database that were not used for training of the acoustic model (IPA-98-testset)[2]. We picked up 23 speakers and 100 utterances in total for both male and female. The uttered sentences are text-open

[2] http://www.milab.is.tsukuba.ac.jp/jnas/test-set/male/male1_LARGE.txt, http://www.milab.is.tsukuba.ac.jp/jnas/test-set/female/female1_LARGE.txt

Table 5.7: Evaluation of language models

	accuracy	LM size
20K 75month cutoff-1-1	94.3	79MB
20K 75month compress10%	94.3	38MB
60K 75month cutoff-1-1	93.7	100MB
60K 75month compress10%	93.5	55MB

to the language model training. The total number of words in the test-set samples is 1575, and 0.44% of them are unknown words which uncovered by the 20K lexicon.

Word accuracy is used as the evaluation criterion. Definition of the word accuracy in Japanese involves several issues. First, the length of word unit is not explicitly determined and may vary when using different morpheme analysis. In this experiment, word unit is fixed to the morpheme for simple discussion. Second, even if the same morpheme analysis is used, there is some ambiguity that the same sequence may be interpreted in several ways. For example, Japanese morpheme "tomoni" (=together) can be segmented as "tomo" + "ni". Toward this problem, concatenation of compound words in recognition result is performed before matched to the answer transcription. Furthermore, Japanese has several transcription for a word, in Kanji (Chinese character) and in Kana (pronunciation-based representation), which makes answer matching difficult. Converting them all to Kana, however, confuses homonyms. Here we use the Kanji transcription only. These ambiguity cause the automatic scoring tool to mis-judge the accuracy. The error increases by about 0.5% compared with hand decision.

5.4.1 Evaluation of Language Models

At first, we present evaluation of language models. The triphone 2000x16 and standard decoding is used. The model compressed by entropy-based cut-off (compress10%) is compared with the baseline model (cut-off 1-1) for both 20K-word and 60K-word task.

Their word accuracy and size in memory is shown in Table 5.7. Increasing vocabulary size only resulted in a slight degradation of recognition accuracy by less than 1%, while the processing time increased by almost 30%. It is also confirmed that compressing 3-gram entries to one tenth does not affect the accuracy.

Table 5.8: Evaluation of acoustic models (male)

	mix.4	mix.8	mix.16
GD monophone	75.3	79.6	83.9
GI monophone	68.3	78.0	81.7
GD triphone 2000	92.0	92.6	94.3
GI triphone 2000	89.3	91.8	92.5
GD PTM 129x64 (3000)		92.4	
GI PTM 129x64 (3000)		89.5	

Table 5.9: Evaluation of acoustic models (female)

	mix.4	mix.8	mix.16
GD monophone	75.5	80.7	88.9
GI monophone	76.0	80.8	84.7
GD triphone 2000	92.0	94.4	95.2
GI triphone 2000	92.3	93.4	94.8
GD PTM 129x64 (3000)		94.6	
GI PTM 129x64 (3000)		94.3	

5.4.2 Evaluation of Acoustic Models

Next, we present evaluation of a variety of acoustic models. Here, the baseline language model (75 month, cut-off 1-1) and the decoder are used with standard setting.

The word accuracy is listed in Table 5.8 for male and Table 5.9 for female speakers, respectively.

It is observed that the monophone model needs many mixture components to achieve high accuracy, while increase of model complexity of the triphone does not improve so much. It suggests that much more data is needed to train the triphone model to the full extent. The PTM model shows high accuracy almost equal to the triphone models with much less mixtures, while the recognition time is shorter than a half of the triphone HMM. The gender-independent models generally lose accuracy by a several percentage as compared with gender-dependent models. There is not much performance difference between male and female results.

5.4.3 System Assessment

Finally, performance of the total system is summarized in Table 5.10 for 20K-word system and Table 5.11 for 60K-word system. As performance measure, word accuracy (Acc.), word %correct (Corr.), and processing speed in real time factor is shown in the tables.

The fast decoding in the tables means that the precise inter-word context dependency is not handled on the second pass, and aggressive beam pruning is used for Gaussian pruning.

Three typical system configurations are listed, "accuracy-oriented system", "efficiency-oriented system" and "simple system". In the accuracy-oriented system with precise acoustic model (2000x16) and standard decoding with sufficient resources, word accuracy of about 95% is achieved. The efficiency-oriented system uses an efficient PTM HMM and the search parameters are also tuned, to achieve as fast decoding as possible while keeping word accuracy of 90%. The result shows that recognition accuracy of almost 90% can be performed in almost real time in a current generic PCs. Using compressed language model also suppress the work memory. The simple system uses inexpensive monophone model, intended to be used in environment of low machine resources. At a cost of degradation in word accuracy to about 84%, much faster decoding can be achieved. These methods also work as well with a 60K-word task.

5.5 System Refinements

The overall improvements of the Japanese dictation system since the project is founded is summarized. Table 5.12 is for systems oriented for obtaining high accuracy, and Table 5.13 is for systems considering recognition speed. These figures are almost the best ever reported on the same test and training set.

Substantial improvements have been achieved by continuous refinement of models and search algorithm. No training corpus has been largely strengthened for the models (only text corpus was extended from 45 months to 75 months on the same domain), and no hardware improvements are taken place for recognition speed, that is, the same workstation is used throughout to measure the search efficiency strictly.

5.6 Conclusion

The key property of the software toolkit is generality and portability. As the formats and interfaces of the modules are widely acceptable, any modules can be easily replaced. Thus, the toolkit is suitable for research on individual component techniques as well as development of specific systems. Moreover, it is possible to replace or integrate modules that are developed at different sites and evaluate them.

Table 5.10: Specification of typical systems (20K)

	simple	efficiency	accuracy
acoustic model	monophone 16 (0.5MB)	PTM 129x64 (3.0MB)	triphone 2000x16 (8.6MB)
language model	75month compress10% (38.0MB)		75month cutoff-1-1 (78.5MB)
decoding	fast	fast	standard
CPU time	1.1x RT	2.3x RT	12.8x RT
Acc,Corr(male)	82.6, 83.5c	89.1, 91.1c	94.3, 95.4c
Acc,Corr(female)	85.7, 87.1c	91.8, 93.1c	95.2, 96.2c
Acc,Corr(GD ave)	84.2, 85.3c	90.5, 92.1c	94.8, 95.8c
Acc,Corr(GI)	81.5, 84.0c	89.7, 91.1c	93.7, 94.7c

RT (Real Time): 5.8sec./sample, CPU: Ultra SPARC 300MHz

Table 5.11: Specification of typical systems (60K)

	efficiency	accuracy
acoustic model	PTM 129x64 (3.0MB)	triphone 2000x16 (8.6MB)
language model	75 compress10% (54.5MB)	75 cutoff-1-1 (99.7MB)
decoding	fast	standard
CPU time	2.9x RT	16.9x RT
Acc,Corr(male)	89.1, 90.9c	93.7, 94.6c
Acc,Corr(female)	91.6, 92.7c	93.4, 94.9c
Acc,Corr(GD ave)	90.4, 91.8c	93.6, 94.8c
Acc,Corr(GI)	88.9, 90.5c	93.2, 94.2c

RT (Real Time): 5.8sec./sample, CPU: Ultra SPARC 300MHz

Table 5.12: Performance improvements in accurate system

year	5k system	20k system	60k system
'97	93.0%	N/A	N/A
'98	93.5%	92.6%	N/A
'99	–	94.8%	93.6%

CPU: Ultra SPARC 300MHz

Table 5.13: Performance improvements in fast system

year	5k system	20k system	60k system
'97	87.2% (2.8 xRT)	N/A	N/A
'98	87.7% (1.1 xRT)	84.2% (1.1xRT)	N/A
'99	–	90.5% (2.3xRT)	90.4% (2.9xRT)

CPU: Ultra SPARC 300MHz

It is proven that our platform demonstrates reasonable performance when adequately integrated. The current version of the software (decoder) works under standard Unix platform. It needs about 64MB memory including space for the language model.

Finally, our system achieves word accuracy of approximately 95% in 20,000-word dictation, and above 90% in a real time speed. Also it works with 60,000-word task, where word accuracy reaches to 94%.

Chapter 6

Conclusions

We have studied various issues for multi-pass search algorithm in large vocabulary continuous speech recognition (LVCSR). The search is separated into several passes, and the search space is gradually narrowed by the preliminary pass. The advantage over the conventional one-pass search is that large-scale and detailed models are easily applied. It is also easy to introduce other complicated linguistic knowledge for further refinement.

We have proposed a two-pass search algorithm using word trellis index as an intermediate form between passes. By keeping indexes of all word ends survived within beam, the search space can be efficiently constrained without loss of information. This form is superior to the conventional word graph form in both accuracy and computational cost. It realizes accurate re-scoring in the second pass, and enables simple one-best approximation to be applied without loss of accuracy.

Two implementations of the proposed search algorithm have been studied for different language models: probabilistic word N-gram and deterministic finite state grammar. The word N-gram based recognizer uses a single lexical tree with 2-gram constraint on the first pass, and performs stack decoding with word 3-gram on the second pass. On the other hand, the grammar based recognizer uses a category-wise tree lexicon together with category-pair constraint on the first pass, and applies the full grammar constraint on the second pass. The grammar-based system can be considered as a special case of the word N-gram based system, where word connections are given deterministically. So these implementations are very straight-forward and general in that it realizes the same algorithm over the different language models with only slight modification.

The superiority of this two-pass search strategy over the conventional methods is clearly significant. In word N-gram based system, the workspace needed for the search

was reduced from 400MB to 30MB without any accuracy loss. In grammar based system, the same accuracy with one-pass search was achieved with far less beam width. Other search techniques for LVCSR such as enveloped best-first search, approximate handling of inter-word context dependency, 1-gram factoring are also implemented to achieve high performance.

The multi-pass search strategy has a disadvantage that the process overhead delays the turn-around response. As the whole input must be processed beforehand, the later search cannot be started before the user finishes the input. However, our search algorithm performs efficient stack decoding which completes in a moment (less than one second) for most samples and no significant delay was found.

We have also studied a new acoustic model called phonetic tied-mixture model for efficient decoding. By sharing the Gaussian sets among states based on the monophone topology and introducing Gaussian pruning, the acoustic computation cost was remarkably reduced to almost 20% without any accuracy loss. As the output probabilities of all possible states can be computed and stored on the first pass, the second pass proceeds fast even with context-dependent models.

Based on the studies above, we have developed a portable recognition software for both word N-gram and finite automaton grammar. The former is named Julius and the latter is named Julian. They have a general interface to various language models (word N-gram and DFA grammar for each) and acoustic models (HMM), which is helpful for implementing various speech recognition systems. These recognition engines serves as a common platform to evaluate various models through recognition experiments. Julius is adopted as a platform engine of Japanese Dictation Toolkit Project. In this project, our Japanese dictation system has achieved word accuracy of 95% in the 20,000-word dictation task, and 90% in speed of almost real time. The system also works for even larger vocabulary (60,000 tested) with a little additional computational cost.

Acknowledgments

The study has been accomplished in Kyoto University since I entered graduate school. I would like to express my foremost gratitude to Professor Michihiko Minoh (Kyoto University) for his constant guidance and fruitful suggestions through the doctor course.

I would like to express my greatest appreciation to Professor Toru Ishida (Kyoto University) and Professor Masahiko Sato (Kyoto University) for their invaluable comments to the thesis.

I would like to express my sincere gratitude to Emeritus Professor Shuji Doshita (Kyoto University) for his enlightening guidance through the bachelor and master course. His earnest attitudes toward research invited me to the research activities.

I greatly appreciate Associate Professor Tatsuya Kawahara (Kyoto University). The core of this work originated with his pioneering ideas in speech recognition algorithm, which led me to a new research idea. This work could not have been accomplished without his continuous guidance and warmful support.

I would like to thank Professor Kiyohiro Shikano (Nara Institute of Science and Technology) and Associate Professor Kazuya Takeda (Nagoya University) for their support in the work. I would also like to thank Katsunobu Itou (Electrotechnical Laboratory) and all members of the Japanese Dictation Toolkit Project for their comments and suggestions.

I owe a great deal to the members of Speech Media Laboratory (formerly Professor Doshita's Laboratory). I had many suggestions from Dr. Masahiro Araki (currently at Kyoto Institute of Technology) and Mr. Kiyokazu Miki (currently at NEC). I am also indebted to all members of our laboratory for their helpful support and discussions.

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

Bibliography

[1] K.Takeda, K.Ito, T.Matsuoka, T.Takezawa, and K.Shikano. Developing text database for large vocabulary continuous speech recognition technologies (in Japanese). In *IPSJ Tech. Report*, 96-SLP-11-9, 1996.

[2] M.Nishimura and N.Itoh. A word-based japanese dictation system (in Japanese). *Trans. IEICE*, Vol. J81-D-II, No. 1, pp. 10–17, 1998.

[3] T.Matsuoka, K.Ohtsuki, T.Mori, S.Furui, and K.Shirai. Japanese Large-vocabulary Continuous-speech Recognition using a Business-newspaper Corpus. In *Proc. ICSLP*, Vol. 1, pp. 303–306, 1996.

[4] A. Lee, T. Kawahara, and S. Doshita. An efficient two-pass search algorithm using word trellis index. In *Proc. ICSLP*, pp. 1831–1834, 1998.

[5] P.S.Gopalakrishnan, L.R.Bahl, and R.L.Mercer. A tree search strategy for large-vocabulary continuous speech recognition. In *Proc. IEEE-ICASSP*, pp. 572–575, 1995.

[6] J.J.Odel, V.Valtchev, P.C.Woodland, and S.J.Young. A one pass decorder design for large vocabulary recognition. In *Proc. ARPA Human Language Technology Workshop*, pp. 405–410, 1994.

[7] L.R.Bahl, S.V.de Gennaro, P.S.Gopalakrishman, and R.L.Mercer. A fast approximate acoustic match for large vocabulary speech recognition. Vol. 1, No. 1, pp. 59–67, 1993.

[8] L.Nguyen, R.Schwartz, Y.Zhao, and G.Zavaliagkos. Is N-best dead ? In *Proc. DARPA Speech & Natural Language Workshop*, pp. 411–414, 1994.

[9] Y.Noda, S.Matsunaga, and S.Sagayama. An approximation technique in large-vocabulary continuous speech recognition using a word graph (in Japanese). In *IEICE Tech. Report*, SP96-102, 1997.

[10] R.Schwartz et al. A Comparison of Several Approximate Algorithms for Finding Multiple (N-best) Sentence Hypotheses. In *Proc. ICASSP*, Vol. 1, pp. 701–704, 1991.

[11] H.Ney and X.Aubert. A Word Graph Algorithm for Large Vocabulary Continuous Speech Recognition. In *Proc. ICSLP*, pp. 1355–1358, 1994.

[12] F. K. Soong and Eng-Fong Huang. A tree-trellis based fast search for finding the N best sentence hypotheses in continuous speech recognition. In *Proc. IEEE-ICASSP*, pp. 705–708, 1991.

[13] J.K.Chen, F.K.Soong, and L.Sh.Lee. Large vocabulary word recognition based on tree-trellis search. In *Proc. ICASSP*, Vol. 2, pp. 137–140, 1994.

[14] G.Antoniol, F.Brugnara, M.Cettolo, and M.Federico. Language model representations for beam-search decoding. In *Proc. IEEE-ICASSP*, pp. 588–591, 1995.

[15] T.Matsuoka, K.Ohtsuki, T.Mori, K.Yoshida, S.Furui, and K.Shirai. Japanese Large-Vocabulary Continuous-Speech Recognition using a Business-Newspaper Corpus. In *Proc. ICASSP*, Vol. 3, pp. 1803–1806, 1997.

[16] K.Itou, M.Yamamoto, K.Takeda, T.Takezawa, T.Matsuoka, T.Kobayashi, K.Shikano, and S.Itahashi. The design of the newspaper-based Japanese large vocabulary continuous speech recognition corpus. In *icslp*, pp. 3261–3264, 1998.

[17] 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*, Vol. J77-D-II, No. 1, pp. 1–8, 1994.

[18] K.Kita, T.Kawabata, and H.Saito. HMM continuous speech recognition using generalized LR parsing. *Trans. IPSJ*, Vol. 31, No. 3, pp. 472–480, 1990.

[19] M.Ida and S.Nakagawa. Comparison between beam search and A* search methods for speech recognition (in Japanese). In *IEICE Tech. Report*, SP96-12, 1996.

[20] J.W.Klovstad and L.F.Mondshein. The CASPERS linguistic analysis system. In *IEEE Sympo. Speech Recognition*, pp. 234–240, 1974.

[21] S.Nakagawa and A.Kai. A context-free grammar driven, one pass HMM-based continuous speech recognition method (in Japanese). *Trans. IEICE*, Vol. J76-DII, No. 7, pp. 1337–1345, 1993.

[22] D.B.Paul. An efficient A* stack decoder algorithm for continuous speech recognition with a stochastic language model. In *Proc. IEEE-ICASSP*, Vol. 1, pp. 25–28, 1992.

[23] K.Itou, S.Hayamizu, and H.Tanaka. Continuous Speech Recognition By Context-Dependent Phonetic HMM And An Efficient Algorithm For Finding N-Best Sentence Hypotheses. In *Proc. ICASSP*, Vol. 1, pp. 21–24, 1992.

[24] T.Watanabe, K.Yoshida, and K.Hatazaki. High speed continuous speech recognition using a bundle search algorithm (in Japanese). *Trans. IEICE*, Vol. J75-DII, No. 11, pp. 1761–1769, 1992.

[25] S.Young, J.Jansen, and J.Odell D.Ollason P.Woodland. *The HTK Book*, 1995.

[26] L. Nguyen and R. Schwartz. The BBN single-phonetic-tree fast-match algorithm. In *Proc. ICSLP*, pp. 1827–1830, 1998.

[27] S.J.Young. The general use of tying in phoneme-based HMM speech recognizer. In *Proc. IEEE-ICASSP*, pp. 569–572, 1992.

[28] G.Zavaliagkos, J.McDonough, D.Miller, A.El-Jaroudi, J.Billa, F.Richardson, K.Ma, M.Siu, and H.Gish. The BBN BYBLOS 1997 large vocabulary conversational speech recognition system. In *Proc. IEEE-ICASSP*, pp. 905–908, 1998.

[29] A.Sankar. A new look at HMM parameter tying for large vocabulary speech recognition. In *Proc. IEEE-ICSLP*, pp. 193–196, 1998.

[30] V.Digalakis and H.Murveit. Genones: optimizing the degree of mixture tying in a large vocabulary hidden markov model based speech recognizer. In *Proc. IEEE-ICASSP*, pp. I–537–540, 1994.

[31] J.Duchateau, K.Demuynck, and D.Van Compernolle. Fast and accurate acoustic modelling with semi-continuous HMMs. In *Speech Communication*, pp. 24(1):5–17, 1998.

[32] A.Lee, T.Kawahara, and S.Doshita. An efficient two-pass search algorithm using word trellis index. In *icslp*, pp. 1831–1834, 1998.

[33] T.Kawahara, T.Kobayashi, K.Takeda, N.Minematsu, K.Itou, M.Yamamoto, A.Tamada, T.Utsuro, and K.Shikano. Sharable software repository for Japanese large vocabulary continuous speech recognition. In *icslp*, pp. 3257–3260, 1998.

[34] S.J.Young. A review of large-vocabulary continuous-speech recognition. *IEEE Signal Processing magazine*, Vol. 13, No. 5, pp. 45–57, 1996.

[35] T.Matsuoka, K.Ohtsuki, T.Mori, S.Furui, and K.Shirai. Japanese large-vocabulary continuous-speech recognition using a business-newspaper corpus. In *Proc. ICSLP*, pp. 22–25, 1996.

[36] Y.Matsumoto, A.Kitauchi, T.Yamashita, Y.Hirano, H.Matsuda, and M.Asahara. Japanese morphological analysis system chasen version 2.0 manual 2nd edition. Information Science Technical Report NAIST-IS-TR99013, Nara Institute of Science and Technology, 1999.

[37] *The CMU-Cambridge Statistical Language Modeling Toolkit v2*, 1997.

[38] N.Yodo, K.Shikano, and S.Nakamura. Compression algorithm of trigram language models based on maximum likelihood estimation. In *Proc. ICSLP*, Vol. 5, 1998.

List of Publications by the Author

Major Publications

[1] A.Lee, T.Kawahara, and S.Doshita. Large vocabulary continuous speech recognition based on multi-pass search (in Japanese). *Trans. IEICE*, J82-DII(1):1–9, 1999.

[2] A.Lee, T.Kawahara, and S.Doshita. Large vocabulary continuous speech recognition parser based on A* search using grammar category-pair constraint (in Japanese). *Trans. IPSJ*, 40(4):1374–1382, 1999.

[3] A.Lee, T.Kawahara, K.Takeda, and K.Shikano. Large vocabulary continuous speech recognition based on phonetic tied-mixture model (in Japanese). *Trans. IEICE*, (accepted for publication), 2000.

[4] T.Kawahara, A.Lee, T.Kobayashi, K.Takeda, N.Minematsu, K.Itou, A.Ito, M.Yamamoto, A.Yamada, T.Utsuro, and K.Shikano. Japanese dictation toolkit — 1997 version — (in Japanese; translated in *J. Acoust. Soc. Japan (E)*). *J. Acoust. Soc. Japan*, 55(3):175–180, 1999.

[5] T.Kawahara, A.Lee, T.Kobayashi, K.Takeda, N.Minematsu, K.Itou, M.Yamamoto, A.Yamada, T.Utsuro, and K.Shikano. Japanese dictation toolkit — 1998 version — (in Japanese). *J. Acoust. Soc. Japan*, 56(4):255–259, 2000.

[6] A.Lee, T.Kawahara, and S.Doshita. An efficient two-pass search algorithm using word trellis index. In *Proc. Int'l Conf. on Spoken Language Processing*, pages 1831–1834, 1998.

[7] A.Lee, T.Kawahara, K.Takeda, and K.Shikano. A new phonetic tied-mixture model for efficient decoding. In *Proc. IEEE Int'l Conf. Acoust., Speech & Signal Process.*, pages 1269–1272, 2000.

[8] T.Kawahara, A.Lee, T.Kobayashi, K.Takeda, N.Minematsu, K.Itou, A.Ito, M.Yamamoto, A.Yamada, T.Utsuro, and K.Shikano. Common platform of Japanese large vocabulary continuous speech recognizer assessment – proposal and initial results –. In *Proc. Oriental-COCOSDA workshop*, pages 117–122, 1998.

Technical Reports

[1] A.Lee, T.Kawahara, and S.Doshita. Large vocabulary continuous speech recognition based on A* search (in Japanese). In *IPSJ Tech. Report*, 96-SLP-11-4, 1996.

[2] A.Lee, T.Kawahara, and S.Doshita. Multi-pass search for large vocabulary continuous speech recognition using word N-gram (in Japanese). In *IPSJ Tech. Report*, 97-SLP-16-4, 1997.

[3] T.Kawahara, A.Lee, K.Itou, T.Kobayashi, A.Itou, T.Utsuro, T.Shimizu, M.Tamoto, K.Arai, N.Minematsu, M.Yamamoto, T.Takezawa, K.Takeda, T.Matsuoka, and K.Shikano. Common platform of Japanese large vocabulary continuous speech recognition research (in Japanese). In *IPSJ Tech. Report*, 97-SLP-18-1, 1997.

[4] A.Lee, T.Kawahara, and S.Doshita. JULIUS — A Japanese LVCSR engine using word trellis index (in Japanese). In *IEICE Tech. Report*, SP98-3, 1998.

[5] T.Kawahara, A.Lee, T.Kobayashi, K.Takeda, N.Minematsu, K.Itou, A.Itou, M.Yamamoto, A.Yamada, T.Utsuro, and K.Shikano. Evaluation of Japanese dictation toolkit – 1997 version – (in Japanese). In *IPSJ Tech. Report*, 98-SLP-21-10, 1998.

[6] A.Lee, T.Kawahara, and S.Doshita. Large vocabulary continuous speech recognition parser based on A* search using grammar category-pair constraint (in Japanese). In *IEICE Tech. Report*, SP98-110, 1998.

[7] K.Kato, A.Lee, and T.Kawahara. Topic independent language model and its adaptation for dictation of lecture speech (in Japanese). In *IPSJ Tech. Report*, 99-SLP-26-2, 1999.

[8] T.Kawahara, A.Lee, T.Kobayashi, K.Takeda, N.Minematsu, K.Itou, M.Yamamoto, A.Yamada, T.Utsuro, and K.Shikano. Evaluation of Japanese dictation toolkit – 1998 version – (in Japanese). In *IPSJ Tech. Report*, 99-SLP-26-6, 1999.

[9] A.Lee and T.Kawahara. Improvements of A*-based search algorithm in LVCSR engine Julius (in Japanese). In *IPSJ Tech. Report*, 99-SLP-27-5, 1999.

[10] H.Nanjo, A.Lee, and T.Kawahara. Automatic diagnosis of recognition errors in large vocabulary continuous speech recognition system (in Japanese). In *IPSJ Tech. Report*, 99-SLP-27-6, 1999.

[11] A.Lee, T.Kawahara, K.Takeda, and K.Shikano. Phonetic tied-mixture model for LVCSR (in Japanese). In *IEICE Tech. Report*, SP99-100, 1999.

[12] T.Kawahara, A.Lee, T.Kobayashi, K.Takeda, N.Minematsu, S.Sagayama, K.Itou, A.Ito, M.Yamamoto, A.Yamada, T.Utsuro, and K.Shikano. Evaluation of japanese dictation toolkit – 1999 version – (in Japanese). In *IPSJ Tech. Report*, 99-SLP-31-2, 2000.

[13] H.Nanjo, K.Kato, M.Mimura, A.Lee, and T.Kawahara. Diagnosis and evaluation of various LVCSR systems (in Japanese). In *IPSJ Tech. Report*, 2000-SLP-31-11, 2000.

Oral Presentations

[1] A.Lee, T.Kawahara, and S.Doshita. JULIUS — A Japanese large vocabulary continuous speech recognition engine based on word n-gram and multi pass search strategy (in Japanese). In *Proc. Meeting Acoust. Soc. Japan*, 1-6-24, spring 1998.

[2] A.Lee, T.Kawahara, and S.Doshita. Evaluation of A* search on FA-based large vocabulary continuous speech recognition (in Japanese). In *Proc. Meeting Acoust. Soc. Japan*, 3-1-10, autumn 1998.

[3] A.Lee, T.Kawahara, and S.Doshita. Improvement of large vocabulary continuous speech recognition engine JULIUS (in Japanese). In *Proc. Meeting Acoust. Soc. Japan*, 2-1-12, spring 1999.

[4] A.Lee and T.Kawahara. The improvement of inter-word triphone handling in LVCSR engine Julius (in Japanese). In *Proc. Meeting Acoust. Soc. Japan*, 2-1-1, autumn 1999.

[5] A.Lee, T.Kawahara, K.Takeda, and K.Shikano. Large vocabulary continuous speech recognition using phonetic tied-mixture model (in Japanese). In *Proc. Meeting Acoust. Soc. Japan*, 2-8-5, spring 2000.