TR-I-0273

A Study on Language Modeling for Speech Recognition

Kenji KITA

1992

Abstract

This report is concerned mainly with language modeling for automatic speech recognition. Speech recognition based on acoustic information alone does not permit good performance for large-vocabulary tasks. Linguistic information can assist in solving speech recognition problems. The report proposes efficient speech recognition algorithms that take full advantage of the grammatical constraints supplied by syntactic language models, and introduces some stochastic language models to improve speech recognition performance.

ATR Interpreting Telephony Research Laboratories

Contents

1 Introduction 1
- 1.1 Speech Recognition: Historical View . . . 1
- 1.2 Purpose of the Study . . . 2
- 1.3 Syntactic Language Models . . . 3
- 1.4 Stochastic Language Models . . . 5
- 1.5 Thesis Outline . . . 7

2 HMM-LR Speech Recognition 9
- 2.1 Introduction . . . 9
- 2.2 Hidden Markov Models . . . 10
  - 2.2.1 Basic Concepts . . . 10
  - 2.2.2 Computation of Acoustic Probability . . . 11
  - 2.2.3 Estimation of HMM Parameters . . . 12
- 2.3 Generalized LR Parsing . . . 13
  - 2.3.1 Context-Free Grammars . . . 13
  - 2.3.2 LR Parsing . . . 14
  - 2.3.3 Generalized LR Parsing . . . 16
  - 2.3.4 Construction of an LR Parsing Table . . . 17
- 2.4 HMM-LR Speech Recognition . . . 18
  - 2.4.1 Predictive LR Parsing . . . 18
  - 2.4.2 HMM-LR Overview . . . 19
  - 2.4.3 Algorithm . . . 23
  - 2.4.4 Refinements of the Algorithm . . . 24
- 2.5 Conclusion . . . 25

3 Implementations of the HMM-LR Speech Recognition System 27
- 3.1 Introduction . . . 27
- 3.2 Measure of Task Complexity . . . 28
  - 3.2.1 Task Entropy . . . 28
  - 3.2.2 Phone Perplexity . . . 29
- 3.3 Task and Grammars . . . 29
- 3.4 Speech Data . . . 30
- 3.5 Baseline HMM-LR System . . . 31

3.5.1 System Overview 31
3.5.2 Recognition Experiments 32
3.5.3 Comparison with the Lattice Parsing Approach 33
3.6 Improved HMM-LR System 33
3.6.1 HMM Phone Models 33
3.6.2 Multiple Codebooks 34
3.6.3 Duration Control 34
3.6.4 Fuzzy Vector Quantization 35
3.6.5 Speaker Adaptation 36
3.6.6 Recognition Experiments 37
3.7 Very Large Vocabulary Recognition 40
3.7.1 Vocabulary 40
3.7.2 Recognition Experiments 40
3.8 Conclusion 43

4 Stochastic Language Models **45**
4.1 Introduction 45
4.2 Trigram Model of Japanese Syllables 46
4.2.1 Trigram Model 46
4.2.2 Estimation of Interpolation Weights 47
4.3 Stochastic LR Parsing 47
4.3.1 Stochastic Context-Free Grammar 48
4.3.2 Estimation of Rule Probabilities 48
4.3.3 Stochastic LR Parsing 50
4.3.4 Construction of a Stochastic LR Parsing Table 50
4.4 Trigram Model of Context-Free Rewriting Rules 52
4.5 Evaluation of the Models 53
4.5.1 Estimation of Model Parameters 53
4.5.2 Evaluation by Perplexity Reduction 54
4.5.3 Evaluation by Recognition Experiments 55
4.6 Conclusion 59

5 Sentence Recognition Using Two-Level LR Parsing **61**
5.1 Introduction 61
5.2 Linguistic Constraints between Phrases 62
5.3 Two-Level LR Parsing 63
5.3.1 Inter- and Intra-Phrase Grammars 63
5.3.2 Speech Recognition Using Two-Level LR Parsing 65
5.3.3 Construction of an Intra-Phrase LR Parsing Table 66
5.4 Sentence Recognition Experiments 68
5.4.1 Grammars 68
5.4.2 Speech Data 69
5.4.3 Results 70

5.5 Improvement of Two-Level LR Parsing 71
5.5.1 Problem in Two-Level LR Parsing 71
5.5.2 LR Parsing with a Category Reachability Test 71
5.5.3 Construction of an Augmented LR Parsing Table 73
5.5.4 Improved Two-Level LR Parsing 74
5.5.5 Evaluation 76
5.6 Conclusion 77

6 Unknown Word Processing in Speech Recognition 79
6.1 Introduction 79
6.2 Phonetic Typewriter 80
6.2.1 Phonetic Typewriter Approach 80
6.2.2 HMM-LR Based Phonetic Typewriter 80
6.3 Unknown Word Processing 82
6.3.1 Unknown Word Processing in the HMM-LR System 82
6.3.2 Estimation of the Penalty Factor 83
6.3.3 Relationship to the Phonetic Typewriter 87
6.4 Recognition Experiments 88
6.5 Conclusion 91

7 Conclusion 93
7.1 Summary 93
7.2 Future Work 94
7.3 Final Remarks 96

Appendix A Phrase Recognition Results 97
A.1 The Grammar 97
A.2 Results 120

Appendix B Sentence Recognition Results 133

Bibliography 147

Index 159

List of Figures

2.1 Hidden Markov Model. 10
2.2 Trellis diagram illustrating the computation of the forward calculation. 12
2.3 Context-free grammar. 15
2.4 LR parsing table. 15
2.5 Stack operation in generalized LR parsing. 17
2.6 Stack operation in predictive LR parsing. 20
2.7 Schematic diagram of HMM-LR speech recognizer. 21
2.8 Stacking of a probability array. 22

3.1 Structure of phone models. 31
3.2 Comparison of phrase recognition performances. 39
3.3 Relationship between vocabulary size and recognition rates. 41
3.4 Relationship between vocabulary size and LR table size. 42
3.5 Relationship between vocabulary size and phone perplexity. 42

4.1 Stochastic context-free grammar. 49
4.2 Probability calculation of the derivation tree. 49
4.3 Stochastic LR parsing table. 51

5.1 Inter-phrase grammar. 63
5.2 Intra-phrase grammar. 63
5.3 Inter-phrase LR parsing table. 64
5.4 Intra-phrase LR parsing table. 64
5.5 Speech recognition system using two-level LR parsing. 65
5.6 Construction of an intra-phrase LR parsing table. 67
5.7 Strength of constraints imposed by the inter-phrase grammar. 69
5.8 Problematic intra-phrase grammar. 72
5.9 Problematic intra-phrase LR parsing table. 72
5.10 LR parsing table enhanced with reachable categories. 73
5.11 Phone prediction based on the category reachability. 76

6.1 Grammar for Japanese phonetic structure. 81
6.2 Schematic diagram of the unknown word processing. 84
6.3 Merging task and phonetic grammars. 85
6.4 Graph of the function $S(x)$. 86
6.5 Relationship among HMM-LR based systems. 87

List of Tables

3.1 Example phrases in the conference registration task. 29
3.2 Size and complexity of the grammars. 30
3.3 List of phones used in the baseline HMM-LR system. 31
3.4 The baseline HMM-LR results using the general grammar. 32
3.5 The baseline HMM-LR results using the task-specific grammar. . . . 32
3.6 Phone lattice parsing vs. HMM-LR. 33
3.7 List of phones used in the improved HMM-LR system. 34
3.8 The improved HMM-LR results in the speaker-dependent condition. . 37
3.9 The improved HMM-LR results, without adaptation. 37
3.10 The improved HMM-LR results, speaker-adapted using 25 words. . . 38
3.11 The improved HMM-LR results, speaker-adapted using 100 words. . . 38
3.12 The improved HMM-LR results, speaker-adapted with composite models using 100 words. 38
3.13 Vacabulary size of each linguistic source. 41

4.1 Interpolation weights for syllable trigram and rule trigram. 53
4.2 Test-set perplexity of the language models. 54
4.3 Scaling parameters for each language model. 55
4.4 Phrase recognition rates using stochastic language models. 56
4.5 Examples of improved recognition results using the trigram model of Japanese syllables. 56
4.6 Examples of improved recognition results using stochastic LR parsing. 57
4.7 Examples of improved recognition results using the trigram model of context-free rewriting rules. 57
4.8 Examples of improved recognition results using all the models. 58

5.1 Relative ability of linguistic knowledge. 62
5.2 Size and complexity of the intra- and inter-phrase grammars. 68
5.3 Word and sentence accuracy. 70
5.4 Word and sentence accuracy of the improved system. 76

6.1 List of unknown words. 88
6.2 Examples of recognition results that include unknown words. 89
6.3 Transcription rates for phrases that include unknown words. 89
6.4 Phrase recognition rates (with syllable trigrams). 89
6.5 Examples of recognition errors in which vocabulary words are recognized as unknown words. 90

Acknowledgements

The work reported in this thesis was done at ATR Interpreting Telephony Research Laboratories. ATR provided excellent computer resources and a fertile atmosphere.

There are many people whose encouragement and advice have contributed to this study:

First and foremost, I would like to thank Professor Katsuhiko Shirai, my thesis advisor, for his support, encouragement, and guidance. I would also like to thank Professors Kageo Akizuki, Seinosuke Narita, Hiroyoshi Ohara, and Tetsunori Kobayashi for serving on my thesis committee and for their helpful suggestions.

I wish to express my sincere thank to Akira Kurematsu, President of ATR Interpreting Telephony Research Laboratories, and Tsuyoshi Morimoto, Head of the Knowledge and Data Base Department, for their encouragement and for providing the opportunity to conduct this study.

I wish to thank Hiroshi Yasuhara of Oki Electric Industry Co., Ltd. for giving me the chance to come to ATR.

I also owe a great deal to the members of ATR Interpreting Telephony Research Laboratories. Toshiyuki Hanazawa (currently with Mitsubishi Electric Corporation) and Takeshi Kawabata (currently with NTT Basic Research Laboratories) helped me implement the HMM-LR speech recognition system. Hiroaki Saito (currently with Keio University) introduced me to generalized LR parsing. Kiyohiro Shikano (currently with NTT Human Interface Laboratories) gave me numerous important suggestions, and advice. Without their help, I could not possibly have completed this study. I am also grateful to Toshiyuki Takezawa, Terumasa Ehara (currently with NHK Broadcasting Science and Technical Research Laboratories), Akito Nagai, Shigeki Sagayama and the rest of the ATR staff for their kind encouragement and discussions.

Finally, I wish to thank my family and friends for their support. I am especially grateful to my parents for bringing me up, and to my wonderful wife, Sumiko, and our child, Nana, for surrounding me with comfort.

Chapter 1

Introduction

1.1 Speech Recognition: Historical View

Speech recognition is a technology that transforms an utterance into a text. This technology provides a natural, fast, hands- and eyes-free communication interface with computers. There are a number of applications, including a voice-activated typewriter that transcribes any speech, an interpreting telephone that allows interlingual communication, database query by voice.

Speech recognition research has a long history, and considerable progress has been made in the past twenty years. The ARPA (Advanced Research Projects Agency) project [Klatt 77] in the 1970's was the first large-scale effort to build speech understanding systems with a 1,000 word vocabulary. This project focused on the use of artificial intelligence and linguistic knowledge in speech understanding. Several notable systems were developed. The HEARSAY system of CMU [Erman 80] introduced a *blackboard architecture*, which permitted complex interaction among various knowledge sources. CMU's HARPY system [Lowerre 80] was the most successful system in the ARPA project. It used the integrated network representation of knowledge originally employed in the DRAGON system and the beam-search technique to achieve an efficient search. These systems attained impressive accuracy; however, their task was very constrained.

The IBM speech group has also been working on large vocabulary speech recognition since the 1970's. The IBM group introduced a number of very important new ideas, including the Hidden Markov Model (HMM), the stack decoding algorithm [Bahl 83], and deleted interpolation [Jelinek 80]. Around the mid 1980's, they constructed the TANGORA system [Averbuch 87] , a very large vocabulary isolated word dictation system that can scale from 5,000 to 20,000 words by using HMMs and a trigram language model.

In Japan, *dynamic time warping* (DTW) [Itakura 75, Sakoe 78] was introduced in the 1970's. DTW is a very powerful technique for nonlinear alignment of speech, and has been a great influence on the study of speech recognition. This technique was originally developed for isolated word recognition, but this restriction was soon removed. Several techniques for continuous speech recognition were proposed, such as the two-level algorithm [Sakoe 79], the level-building algorithm [Myers 81], and the one-pass algorithm [Bridle 82]. Notable speech recognition systems built in Japan includes the LITHAN system of Kyoto University [Nakagawa 76], the Voice Q-A system of NTT [Shikano 81], a Japanese text dictation system of Tohoku University [Makino 91], and the SUSKIT-II system of Kyoto Institute of Technology [Kobayashi 90].

More recently, several remarkable speech recognition systems have emerged due to the progress in acoustic modeling research, including the SPHINX system of CMU [Lee 89], the BYBLOS system of BBN [Chow 87], and the VOYAGER system of MIT [Zue 90]. Among these, the SPHINX system is known as the most successful speech recognition system today. It uses HMMs and a bigram language model, and achieves a word accuracy of 95.8% on a 997-word speaker-independent continuous speech recognition task. However, even the SPHINX system reaches a sentence accuracy of only around 70%.

In 1986, ATR Interpreting Telephony Research Laboratories was established. ATR has been dedicated to basic research aimed at developing an automatic telephone interpretation system, i.e., a facility that would enable a person speaking in one language to communicate readily by telephone with someone speaking in another language [Kurematsu 87]. At least three constituent technologies are necessary for such a system: speech recognition, natural language processing, and speech synthesis. Moreover, the integrated research of these technologies is also important. This study has been conducted as basic research into the integration of speech recognition and natural language processing and as such to show the feasibility of an automatic telephone interpretation system.

1.2 Purpose of the Study

Speech recognition research is divided into two basic areas: *acoustic modeling* and *language modeling*. This thesis is concerned exclusively with language modeling for automatic speech recognition.

Speech recognition based on acoustic information alone does not permit good performance for large vocabulary tasks. Linguistic information can assist in solving speech recognition problems. The purpose of the thesis is to solve the following problems:

1. **Search space reduction problem.**
   The major problem in computer speech recognition is coping with large search spaces. Blind brute force search is impractical. Suppose that we have a vocabulary of size N. If a speech recognition system uses no linguistic knowledge, that is, the system permits any word to follow any other word, the system's search space size would be of the order N^L for utterances of less than L words. Computationally inexpensive and powerful techniques for search space reduction are indispensable.

2. **Ambiguity resolution problem.**
   Speech inherently contains uncertainty that cannot be resolved by pure acoustic information. During recognition, many acoustically similar hypotheses are built. To effectively rank these hypotheses, the system is required to rely on linguistic likelihood as well as on acoustic likelihood.

3. **Unknown word problem.**
   Current speech recognition systems assume that the user speaks words in the system's vocabulary. If the user speaks an unknown word, i.e., a word not in the vocabulary, the system recognizes other words in place of the unknown word. To make a speech recognition system more user-friendly, the system must take into account unknown word in one way or another.

4. **Recognition scheme suitable for Japanese language.**
   Japanese is very different from European languages such as English. The grammatical structure of Japanese has two levels: the intra-phrase level and the inter-phrase level. Most researchers use the semantic relationship, called *kakariuke relationship*, to deal with the inter-phrase level constraints. However, it is very difficult to make kakariuke relationships for large vocabulary tasks. Moreover, kakariuke relationships are very difficult to utilize for dynamically constraining the search space during recognition. It is necessary to develop an efficient speech recognition mechanism suitable for Japanese language.

In order to address these problems, we will use syntactic language models and stochastic language models. In subsequent sections, these language models will be explained.

1.3 Syntactic Language Models

Speech recognition can be viewed as a search problem. During recognition, many partial hypotheses are built, and then expanded by an acoustic processor. With small vocabularies, it is not hard to perform an exhaustive search. However, as vocabulary size increases into the thousands of words, search space size becomes tremendously large.

A syntactic language model defines a set of sentences that are acceptable by a grammar. It can rule out many ungrammatical hypotheses, or provide a basis for predicting possible subsequent hypotheses.

Usually, phrase-structure grammars are used as syntactic language models. Chomsky classified phrase-structure grammars into four categories, known as the *Chomsky hierarchy* [Chomsky 57]: *unrestricted grammars* (type 0), *context-sensitive grammars* (type 1), *context-free grammars* (type 2), and *regular grammars* (type 3). Grammars with lower numbered types have more generative power.

Regular grammars are appealing because they are easy to implement and can be recognized very efficiently. Some early speech recognition systems, such as DRAGON [Baker 75] and HARPY [Lowerre 80], used finite-state networks of acceptable sentences, which are equivalent to regular grammars. Unfortunately, regular grammars cannot properly characterize natural languages. On the other hand, type 0 grammars are very powerful models in that they can define any recursively enumerable set. However, most linguists think that type 0 grammars are too strong from the viewpoint of human language processing abilities. [1]

Context-free grammars are moderate models. We use context-free grammars as syntactic language models for the following reasons:
(1) Many linguistic phenomena are well described by context-free grammars;
(2) Efficient parsing algorithms have been developed for context-free grammars.

A grammar naturally describes the hierarchical structure of a sentence, and this structure is determined through the process of parsing. Many parsing algorithms for context-free grammars have been developed, such as the *CYK (Cocke-Younger-Kasami) algorithm* [Younger 67], *Earley's algorithm* [Earley 70], and the *generalized LR parsing algorithm* [Tomita 86, Tomita 87]. Among these, the latter is known as the most efficient algorithm for natural language grammars.

The LR parsing algorithm was originally designed for compiler construction. This algorithm is a shift-reduce parsing algorithm which is deterministically guided by a parsing table indicating what action should be taken next. In other words, parsing time information from the grammar is pre-compiled into the parsing table, and the grammar never intervenes in the parsing process. Thus, it runs more efficiently than other parsing algorithms. The original LR algorithm works only for a small subset of context-free grammars; however, the algorithm has been extended to handle general context-free grammars. This extension is called the generalized LR parsing algorithm.

[1] Where the natural languages are located on the Chomsky hierarchy is still an open question. Recent linguistic research shows that some languages include phenomena that cannot be described by context-free grammars. Langendoen and Postal argue that natural languages are not even recursively enumerable. Interested readers are directed to [Savitch 87, Langendoen 84].

In this thesis, we introduce the predictive LR parsing algorithm, which is an extension of the generalized LR parsing algorithm. The predictive LR parser has symbol prediction capability as well as parsing capability. It predicts possible input symbols by referring to the parsing table. This prediction process is based on a table look-up, thus it can be done very efficiently. Accordingly, the predictive LR parsing algorithm provides a computationally inexpensive and powerful mechanism for search space reduction.

We further extend the predictive LR parsing algorithm for Japanese sentential speech recognition by proposing a two-level LR parsing algorithm. The grammatical structure of Japanese has two levels: the intra-phrase level and the inter-phrase level. Thus, using two kinds of grammars, namely an intra-phrase grammar and an inter-phrase grammar, is sufficient for recognizing sentences. The two-level LR parsing algorithm provides a mechanism to use these two kinds of grammatical constraints stepwise during speech recognition.

1.4 Stochastic Language Models

In a stochastic approach, the speech recognition problem is formulated as follows [Bahl 83, Jelinek 90]. First, an acoustic processor transforms a speech waveform into a string of phonetic symbols y. A linguistic processor then interprets the output string y and produces a most probable word sequence $\hat{w}$ that satisfies the following condition:

$$P(\hat{w}|y) = \max_{w} P(w|y) \tag{1.1}$$

By Bayes' rule,

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

Since $P(y)$ does not depend on w, maximizing $P(w|y)$ is equivalent to maximizing $P(y|w)P(w)$. Here, $P(y|w)$ is acoustic probability, and $P(w)$ is *a priori* probability that the word sequence w will be uttered.

The goal of stochastic language models is to estimate $P(w)$. $P(w)$ $(w = w_1, \ldots, w_n)$ can be written as follows:

$$P(w) = \prod_{i=1}^{n} P(w_i|w_1, \ldots, w_{i-1}) \tag{1.3}$$

In practice, we make the assumption that the probability of the word w_i depends only on the previous $(N-1)$ words $w_{i-N+1}, \ldots, w_{i-1}$, i.e.,

$$P(w) \cong \prod_{i=1}^{n} P(w_i|w_{i-N+1}, \ldots, w_{i-1}) \tag{1.4}$$

This model is called the *N-gram model.* N-gram models are computationally practical only for small Ns (typically two or three, and called *bigram* or *trigram*, respectively). Bigram or trigram models are effective for providing local constraints, and have been successfully used in speech recognition [Bahl 83, Shikano 87, Lee 89]. [2]

Another approach to stochastic language modeling is to make phrase-structure rules themselves stochastic [Fu 74]. That is, each rule has the form $< \alpha \rightarrow \beta, p >$ where p is the conditional probability of α being rewritten into β. In the stochastic grammar approach, $P(w)$ can be computed as the product of the conditional probabilities of the rules which are used for deriving w.

In any case, stochastic language models require a great amount of training data to estimate the conditional word probabilities or the rule probabilities. Fortunately, ATR Interpreting Telephony Research Laboratories is constructing a large-scale linguistic database, which will eventually have one million Japanese words and the same number of English words. We use this database to develop stochastic language models.

In this thesis, we will investigate the following three stochastic language models:

1. **The trigram model of Japanese syllables.**
   In Japanese character recognition, a kana (Japanese syllabary alphabet) trigram model has been shown to be effective for correcting recognition errors. Thus, the trigram model of syllables is considered effective and promising for correcting local errors in Japanese speech recognition because a kana roughly corresponds to a Consonant-Vowel (CV) syllable.

2. **Stochastic LR parsing.**
   Spoken sentences include frequently used expressions as well as rarely used ones. We exploit this phenomenon by using a stochastic context-free grammar, in which each grammar rule has a probability value indicating the likelihood of the rule being used. Stochastic LR parsing is very effective for handling the stochastic context-free grammar.

3. **The trigram model of context-free rewriting rules.**
   The structure of a sentence is represented as its derivation, which is a linear sequence of applying rewriting rules. Using the co-occurrence of rewriting rules is very helpful in avoiding rules that generate incorrect sentences. The trigram model of rewriting rules predicts which rewriting rule is likely to be used after a particular rule is used. In a context-free grammar, there is no constraint on the contexts in which a rewriting rule can be applied. This model, however, is considered to have context-sensitivity in terms of a probability.

[2] N-gram models are also considered syntactic models because they describe the constraints between words. For example, a bigram grammar is basically a word network grammar which specifies the words that can follow a particular word. However, it is difficult to approximate linguistic phenomena precisely with N-gram models.

1.5 Thesis Outline

Throughout the thesis, we use Hidden Markov Models (HMM) as the acoustic model. Chapter 2 first provides background information on HMMs and generalized LR parsing. We will then present predictive LR parsing and an efficient speech recognition algorithm, the *HMM-LR algorithm*, which combines HMMs and predictive LR parsing.

The HMM-LR speech recognition algorithm is evaluated in Chapter 3. We will describe implementations of the Japanese phrase recognition system and present our results.

In Chapter 4, three stochastic language models will be investigated: (1) the trigram model of Japanese syllables, (2) stochastic LR parsing, and (3) the trigram model of context-free rewriting rules. These models are incorporated into the HMM-LR speech recognition system, and evaluated by phrase recognition experiments.

In Chapter 5, we will discuss sentential speech recognition. The grammatical structure of Japanese has two levels: intra-phrase and inter-phrase structures. We propose *two-level LR parsing*, which makes it possible to use both intra- and inter-phrase grammatical constraints during speech recognition.

Current speech recognition systems essentially ignore unknown words. Systems are designed to recognize words in the lexicon. However, for using speech recognition systems in real applications, it is very important to process unknown words. Chapter 6 discusses unknown word processing in speech recognition.

Finally, Chapter 7 summarizes the thesis and presents conclusions, including discussion of future work.

Chapter 2

HMM-LR Speech Recognition

2.1 Introduction

One of the major problems in speech recognition is coping with large search spaces. As search space size increases, recognition performance decreases. Syntactic models are effective in reducing the search space and hence increase processing speed and recognition accuracy.

Language models based on word-pair grammars, bigram grammars and trigram grammars are effective for providing local constraints. However, these models cannot capture the global syntax of a language because they describe only the constraints between words. Context-free grammars are increasingly used in speech recognition to overcome this problem [Ney 87, Nakagawa 87].

This chapter is concerned with speech recognition that takes full advantage of context-free grammar constraints. *Generalized LR parsing* [Tomita 86, Tomita 87] is adopted as the basis of the context-free parsing algorithm, and is extended to include symbol prediction capability. This extension, called *predictive LR parsing* [Kita 89a, Kita 89b, Kita 90a, Kita 90b], provides an efficient mechanism for constraining searches in the speech recognition process. We will present the *HMM-LR speech recognition algorithm* [Kita 89a, Kita 89b, Kita 90a, Kita 90b], which is an integration of *Hidden Markov Models* [Levinson 83, Poritz 88, Rabiner 89, Lee 89] and predictive LR parsing. In this algorithm, the LR parser drives Hidden Markov Models directly without any intervening structures such as a phone/word *lattice.* [1] Thus, very accurate and efficient speech parsing is achieved.

[1] A lattice is a set of hypothesized phones/words with different starting and ending positions in the input speech. Speech recognition via a lattice degrades recognition accuracy, because of the information loss due to signal-symbol conversion.

2.2 Hidden Markov Models

The Hidden Markov Model (HMM) is a stochastic approach for modeling speech, and has been used widely for speech recognition [Chow 87, Lee 89, Cohen 90]. It is suitable for handling the uncertainty that arises in speech, for example, contextual effects, speaker variabilities, etc. Moreover, if the HMM unit is a phone, then any word models can be composed of phone models. Thus, it is easy to construct a large vocabulary speech recognition system.

This section gives an overview of *discrete* HMM which first converts speech signals into discrete symbols. [2] Typically, *vector quantization* (VQ) [Linde 80] is used as the acoustic front-end for discrete HMM. More detailed descriptions of HMMs are found in [Levinson 83, Poritz 88, Rabiner 89, Lee 89].

2.2.1 Basic Concepts

An example of a Hidden Markov Model is shown in Figure 2.1. A model has a collection of *states* connected by *transitions*. Two sets of probabilities are attached to each transition. One is a *transition probability* a_{ij}, which provides the probability for taking a transition from state i to state j. The other is an *output probability* $b_{ij}(k)$, which provides the probability of emitting symbol k when taking a transition from state i to state j.

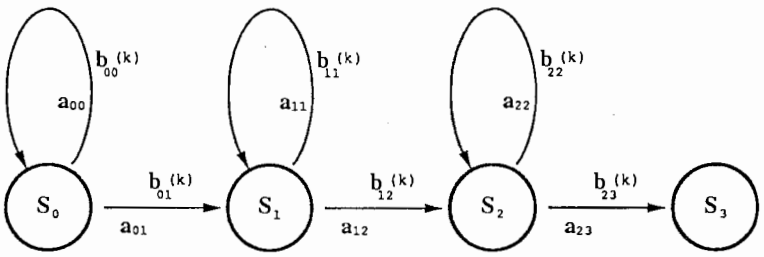


Figure 2.1: Hidden Markov Model.

[2] A set of fixed code-vectors is called a *codebook*.

Formally, a Hidden Markov Model M is defined by a 4-tuple $M =< S, Y, A, B >$.

- S : A set of states $\{s_i\}$ including an initial state S_I and a final state S_F.
- Y : A set of output symbols.
- A : A set of transitions $\{a_{ij}\}$ where a_{ij} is the probability of taking a transition from state i to state j, and $\sum_j a_{ij} = 1$.
- B : The output probability distribution $\{b_{ij}(k)\}$ where $b_{ij}(k)$ is the probability of emitting symbol k when taking a transition from state i to state j, and $\sum_k b_{ij}(k) = 1$.

2.2.2 Computation of Acoustic Probability

We describe the computation of the acoustic probability using HMMs. The problem is to compute $P(y|M)$ from a model M and an observation sequence $y = y_1, y_2, \ldots, y_T$. We first define the forward calculation $\alpha_i(t)$ with recursion on t.

$$\alpha_i(t) = \begin{cases} 0 & t = 0 \;\&\; i \neq S_I \\ 1 & t = 0 \;\&\; i = S_I \\ \sum_j \alpha_j(t-1) a_{ji} b_{ji}(y_t) & t > 0 \end{cases} \tag{2.1}$$

$\alpha_i(t)$ is the probability that the Markov process is in state i having generated the sequence $y_1, y_2, \ldots, y_t$. Finally, $P(y|M)$ is given as follows:

$$P(y|M) = \alpha_{S_F}(T) \tag{2.2}$$

This algorithm is called the *forward algorithm*. The trellis diagram in Figure 2.2 illustrates the computation of α, in which an arrow indicates a legal state transition. Each circle includes the cumulative probability at a particular state and time.

A slightly modified version of the forward algorithm is known as the *Viterbi algorithm*.

$$\alpha_i(t) = \begin{cases} 0 & t = 0 \;\&\; i \neq S_I \\ 1 & t = 0 \;\&\; i = S_I \\ \max_j \alpha_j(t-1) a_{ji} b_{ji}(y_t) & t > 0 \end{cases} \tag{2.3}$$

The Viterbi algorithm is effective in finding the optimal state sequence. To do this, the best transition to each state must be kept at each recursion. After the forward calculation, the optimal state sequence is recovered by backtracking.

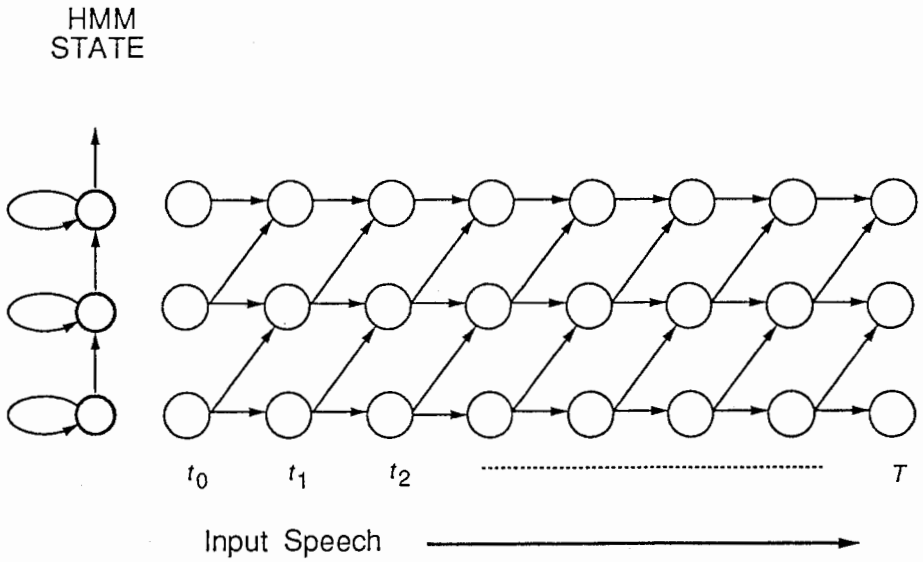


Figure 2.2: Trellis diagram illustrating the computation of the forward calculation.

2.2.3 Estimation of HMM Parameters

The parameters of the model (transition probabilities and output probabilities) can be estimated by the *forward-backward algorithm* given a collection of training data. The forward-backward algorithm is also known as the *Baum-Welch algorithm.* It is based on the *maximum likelihood estimation.*

First, we define the backward calculation $\beta_i(t)$, which is the counterpart of the forward calculation $\alpha_i(t)$.

$$\beta_i(t) = \begin{cases} 0 & t = T \;\&\; i \neq S_F \\ 1 & t = T \;\&\; i = S_F \\ \sum_j a_{ij} b_{ij}(y_{t+1}) \beta_j(t+1) & 0 \leq t < T \end{cases} \tag{2.4}$$

$\beta_i(t)$ is the probability that the Markov process is in state i, and will generate the sequence $y_{t+1}, y_{t+2}, \ldots, y_T$.

We now define $\gamma_{ij}(t)$, which is the probability of taking the transition from state i to state j at time t, conditioned on the observation sequence $y_1, \ldots, y_T$.

$$\gamma_{ij}(t) = \frac{\alpha_i(t-1) a_{ij} b_{ij}(y_t) \beta_j(t)}{\alpha_{S_F}(T)} \tag{2.5}$$

The model's parameters are estimated using the following iterative algorithm, which has been proven to converge.

1. Set initial parameters on $\{a_{ij}, b_{ij}\}$.

2. Re-estimate $\{a_{ij}, b_{ij}\}$ according to Equation 2.6 and 2.7.

$$\begin{aligned} \overline{a}_{ij} &= \frac{\text{expected number of transitions from state } i \text{ to state } j}{\text{expected number of transitions from state } i} \\ &= \frac{\sum_{t=1}^{T} \gamma_{ij}(t)}{\sum_{t=1}^{T} \sum_{k} \gamma_{ik}(t)} \end{aligned} \tag{2.6}$$

$$\begin{aligned} \overline{b}_{ij}(k) &= \frac{\text{frequency of occurrence of symbol } k}{\text{frequency of occurrence of any symbol}} \\ &= \frac{\sum_{t:y_t=k} \gamma_{ij}(t)}{\sum_{t=1}^{T} \gamma_{ij}(t)} \end{aligned} \tag{2.7}$$

3. Replace $\{a_{ij}, b_{ij}\}$ with $\{\overline{a}_{ij}, \overline{b}_{ij}\}$ and repeat from step 2.

2.3 Generalized LR Parsing

2.3.1 Context-Free Grammars

Context-free grammars (CFG) [Aho 86] are extensively used to describe both formal and natural languages. A context-free grammar G is defined by a 4-tuple $G =< N, T, R, S >$.

- N : A set of nonterminal symbols including the start symbol S.
- T : A set of terminal symbols.
- R : A set of rewriting rules (productions) of the form $A \rightarrow \alpha$ where $A \in N$ and $\alpha \in (N \cup T)^*$. [3] A is called the *left side* of the rule, and α is called the *right side* of the rule.
- S : The start symbol.

[3] L^* is called the *Kleene closure* of L. It is the set of all finite-length strings of symbols in a set L.

We introduce some notations and terminologies. A *derivation* is a linear sequence of applications of rules. The notation $\alpha \Longrightarrow \beta$ is used to show a derivation. This means β is derived from α in one step. If β is derived from α in zero or more steps, the notation $\alpha \overset{*}{\Longrightarrow} \beta$ is used. If $S \overset{*}{\Longrightarrow} \alpha$, then α is called a *sentential form*. The derivation is called *rightmost* (or *canonical*) if at each step only the rightmost nonterminal symbol is rewritten. The *leftmost derivation* is defined in the same way. The notation $\alpha \overset{rm}{\Longrightarrow} \beta$ or $\alpha \overset{lm}{\Longrightarrow} \beta$ is used in the appropriate case. A grammar is called *ambiguous* if there exists a string that has at least two rightmost derivations from the start symbol S.

An example of a context-free grammar is shown in Figure 2.3. In this grammar, nonterminal and terminal symbols are designated by upper-case strings and lower-case strings, respectively. This grammar is ambiguous because there are two rightmost derivations for the string "*koreokure*".

Derivation 1:

$$
\begin{aligned}
S &\overset{rm}{\Longrightarrow} NP\ V && \text{(by rule 1)}\\
&\overset{rm}{\Longrightarrow} NP\ k\ u\ r\ e && \text{(by rule 7)}\\
&\overset{rm}{\Longrightarrow} N\ P\ k\ u\ r\ e && \text{(by rule 4)}\\
&\overset{rm}{\Longrightarrow} N\ o\ k\ u\ r\ e && \text{(by rule 6)}\\
&\overset{rm}{\Longrightarrow} k\ o\ r\ e\ o\ k\ u\ r\ e && \text{(by rule 5)}
\end{aligned}
$$

Derivation 2:

$$
\begin{aligned}
S &\overset{rm}{\Longrightarrow} NP\ V && \text{(by rule 1)}\\
&\overset{rm}{\Longrightarrow} NP\ o\ k\ u\ r\ e && \text{(by rule 8)}\\
&\overset{rm}{\Longrightarrow} N\ o\ k\ u\ r\ e && \text{(by rule 3)}\\
&\overset{rm}{\Longrightarrow} k\ o\ r\ e\ o\ k\ u\ r\ e && \text{(by rule 5)}
\end{aligned}
$$

2.3.2 LR Parsing

LR parsing [Aho 86] was originally developed for programming languages. It constructs a rightmost derivation in reverse by scanning an input from left to right. It is applicable for a subset of context-free grammars, called *LR grammars* or *deterministic context-free grammars*.

The LR parser is deterministically guided by an LR parsing table with two subtables (*action table* and *goto table*). The action table determines the next parser action $Action[s, a]$ from the state s currently on top of the stack and the current input symbol a. There are four kinds of actions, *shift*, *reduce*, *accept* and *error*. "Shift" means shift one word from the input buffer onto the stack, "reduce" means reduce constituents on the stack using the grammar rule, "accept" means the input is accepted by the grammar, and "error" means the input is not accepted by the grammar. The goto table determines the next parser state $Goto[s, A]$ from the state s and the grammar symbol A.

(1)	S	$\rightarrow$	$NP\ V$
(2)	S	$\rightarrow$	V
(3)	NP	$\rightarrow$	N
(4)	NP	$\rightarrow$	$N\ P$
(5)	N	$\rightarrow$	$k\ o\ r\ e$
(6)	P	$\rightarrow$	o
(7)	V	$\rightarrow$	$k\ u\ r\ e$
(8)	V	$\rightarrow$	$o\ k\ u\ r\ e$

Figure 2.3: Context-free grammar.

	e	o	u	k	r	\$	S	NP	V	P	N
0		$s3$		$s4$			1	5	2		6
1						acc					
2						$r2$					
3				$s7$							
4		$s8$	$s9$								
5		$s3$		$s11$					10		
6		$s13, r3$		$r3$						12	
7			$s14$								
8					$s15$						
9					$s16$						
10						$r1$					
11			$s9$								
12		$r4$		$r4$							
13		$r6$		$r6$							
14					$s17$						
15	$s18$										
16	$s19$										
17	$s20$										
18		$r5$		$r5$							
19						$r7$					
20						$r8$					

Figure 2.4: LR parsing table.

The LR parsing algorithm is summarized below.

1. *Initialization.* Set p to point to the first symbol of the input. Push the initial state 0 on top of the stack.

2. Consult $Action[s, a]$ where s is the state on top of the stack and a is the symbol pointed to by p.

3. If $Action[s, a]$ ="*shift* s'", push s' on top of the stack and advance p to the next input symbol.

4. If $Action[s, a]$ ="*reduce* $A \rightarrow \beta$", pop $|\beta|$ symbols off the stack and push $Goto[s', A]$ where s' is the state now on top of the stack.

5. If $Action[s, a]$ ="*accept*", parsing is completed.

6. If $Action[s, a]$ ="*error*", parsing fails.

7. Return to 2.

2.3.3 Generalized LR Parsing

Standard LR parsing cannot handle ambiguous grammars. For an ambiguous grammar, the LR parsing table will have *multiple entries* (or *conflicts*). As a general method, a stack-splitting mechanism can be used to cope with multiple entries. Whenever a multiple entry is encountered, the stack is divided into two stacks, and each stack is processed in parallel. Thus, it is possible to use LR parsing to handle an ambiguous grammar.

Actually, it is not necessary to maintain multiple stacks. A single directed acyclic graph representation, called a *graph-structured stack*, is known. Also a *shared-packed forest* can be used to represent parse trees efficiently. As for these techniques, see [Tomita 86, Tomita 87].

Compared with other parsing algorithms such as the CYK (Cocke-Younger-Kasami) algorithm [Younger 67] or Earley's algorithm [Earley 70], the generalized LR parsing algorithm is the most efficient algorithm for natural language grammars.

Figure 2.4 shows the LR parsing table for the grammar in Figure 2.3. The left part is the action table and the right part is the goto table. The entry "*acc*" stands for the action "accept", and blank spaces represent "error". The terminal symbol "$" represents the end of the input. This table has one multiple entry on the row of state 6 at the column labeled "*o*".

Figure 2.5 illustrates the stack operation for the input "*koreokure*" using the grammar in Figure 2.3 and the LR parsing table in Figure 2.4. For simplicity, stacks are not graph-structured, but represented as multiple stacks.

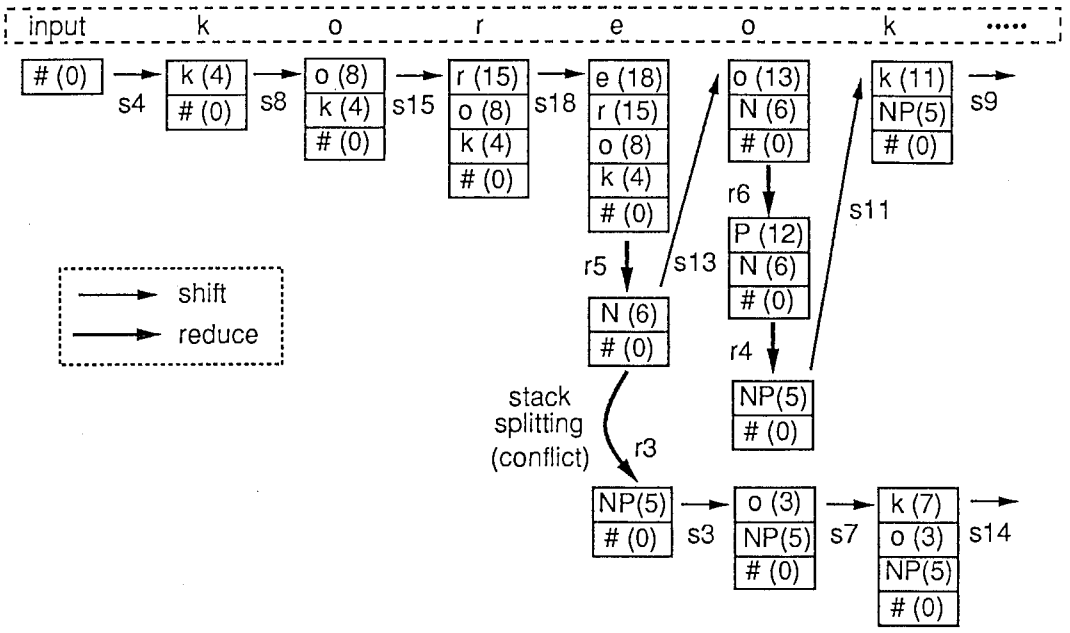


Figure 2.5: Stack operation in generalized LR parsing.

2.3.4 Construction of an LR Parsing Table

In this subsection, we will review the LR parsing table construction algorithm. Although there are several LR parsing table variations, for example, simple LR (SLR), canonical LR, LALR, etc., we will show the construction algorithm for simple LR. A more detailed description is found in [Aho 86].

An LR parsing table can be created through an item set construction algorithm supported by *closure* and *goto* functions. An *item* is of the form $< A \rightarrow \alpha \cdot \beta >$ where $A \rightarrow \alpha\beta$ is a rewriting rule.

Closure Function

If I is an item set, then $Closure(I)$ is constructed from I by the following rules.

1. Every item in I is in $Closure(I)$.

2. If $< A \rightarrow \alpha \cdot B\beta >$ is in $Closure(I)$ and $B \rightarrow \gamma$ is a rewriting rule, then add the item $< B \rightarrow \cdot\gamma >$ to $Closure(I)$. Repeat this rule until no more new items can be added to $Closure(I)$.

Goto Function

If I is an item set and X is a grammar symbol, then $Goto(I, X)$ is defined to be the closure of the item set $< A \rightarrow \alpha X \cdot \beta >$ such that $< A \rightarrow \alpha \cdot X\beta >$ is in I.

Canonical Collection Construction

The canonical collection C of item sets is constructed as follows.

1. Initially $C := Closure(< S' \rightarrow \cdot S >)$ (S' is an auxiliary start symbol).
2. Add $Goto(I, X)$ to C for each item set I in C and each grammar symbol X until no further item sets are created.

LR Parsing Table Construction

Finally, an LR parsing table can be constructed from the canonical collection $C = \{I_0, I_1, \ldots, I_n\}$ as follows.

1. If $< A \rightarrow \alpha \cdot a\beta >$ is in I_i and $Goto(I_i, a) = I_j$, then set $Action[i, a]$ to "*shift* j". Here a must be a terminal.
2. If $< A \rightarrow \alpha\cdot >$ is in I_i, then set $Action[i, a]$ to "*reduce* $A \rightarrow \alpha$" for all a in $Follow(A)$. Here $Follow(A)$ is the set of terminals that can appear immediately to the right of A in some sentential form.
3. If $< S' \rightarrow S\cdot >$ is in I_i, then set $Action[i, \$]$ to "*accept*".
4. If $Goto(I_i, A) = I_j$, then $Goto[i, A] = j$.

The initial state of the parser is the one constructed from the item set containing $< S' \rightarrow \cdot S >$.

2.4 HMM-LR Speech Recognition

2.4.1 Predictive LR Parsing

In standard LR parsing, the next parser action (shift, reduce, accept or error) is determined using the current parser state and next input symbol to check the LR parsing table. However, in speech recognition, it is difficult to know the input symbol because the speech is connected and phone/word boundaries are not clear.

In our approach, the LR parser is used as a language source model for symbol prediction/generation. Thus, we will hereafter call the LR parser the *predictive LR parser*. The predictive LR parser sequentially predicts next symbols from left to right, and splits the parsing stack not only for grammatical ambiguity but also for phone variation.

An action table in an LR parsing table is used for symbol prediction. The action table is a two-dimensional table, where one dimension is indicated by states and the other by terminal symbols. Thus, if the state is given, terminal symbols which might come next will be obtained. To be more precise, terminal symbols associated with shift actions are predicted. The following function returns a set of terminal symbols that are predicted at state s.

$$Prediction(s) = \{x | Action[s, x] \text{ is a shift action}\} \tag{2.8}$$

Figure 2.6 illustrates the stack operation in predictive LR parsing using the grammar in Figure 2.3 and the LR parsing table in Figure 2.4.

We use predictive LR parsing for guiding the search of an HMM-based speech recognizer. However, predictive LR parsing can be applied to other speech recognition approaches. Indeed, it has been successfully integrated with a neural network approach [Sawai 90].

2.4.2 HMM-LR Overview

This subsection gives an informal description of the HMM-LR speech recognition algorithm. We assume here that the HMM unit is the phone, although HMM can be used to represent any unit of speech, e.g., word-based HMM or syllable-based HMM. The predictive LR parser uses context-free rules whose terminal symbols are phone names, thus the phonetic lexicon for the specified task is embedded in the grammar.

The HMM-LR speech recognizer (see Figure 2.7) consists of the predictive LR parser and the HMM phone verifier. First, the parser picks up all phones predicted by the initial state of the LR parsing table, and invokes the HMM models to verify each phone according to the prediction. The parser then proceeds to the next state in the LR parsing table. During this process, all possible partial parses are constructed in parallel. The HMM phone verifier receives a *probability array* (see Figure 2.8) which includes end point candidates and their probabilities, and updates it using an HMM probability calculation. This probability array is attached to each partial parse. When the highest probability in the array is under a certain threshold, the partial parse is pruned. The recognition process proceeds in this way until the entire speech input is exhausted. Very accurate, efficient speech recognition is achieved through integrating the processes of speech recognition and language analysis.

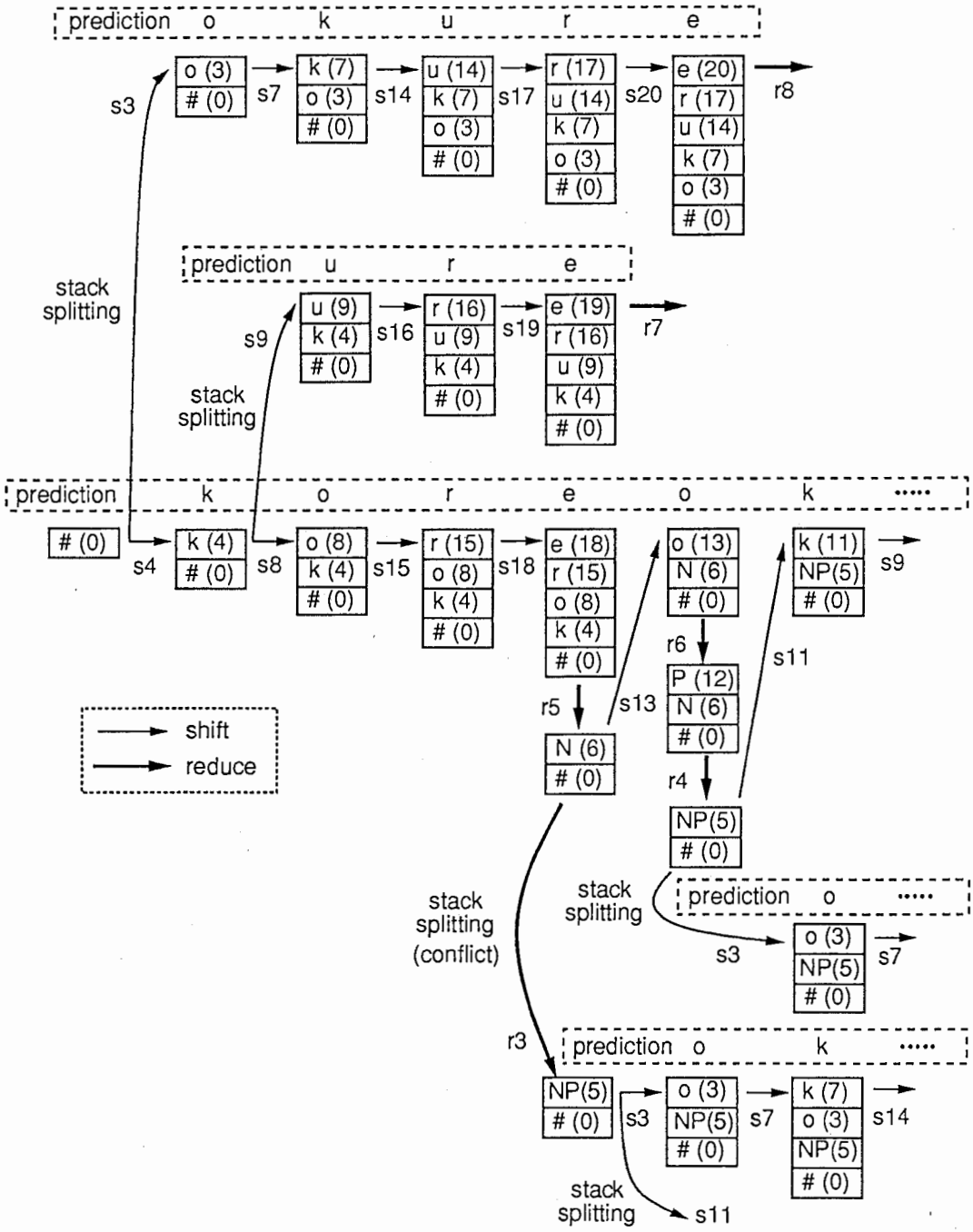


Figure 2.6: Stack operation in predictive LR parsing.

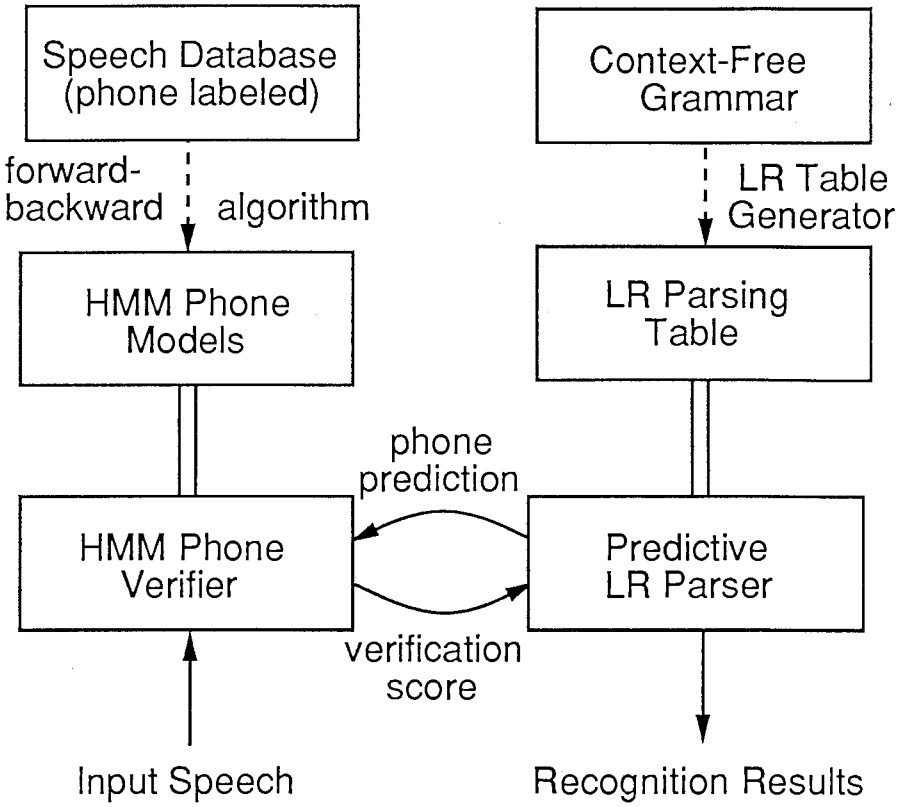


Figure 2.7: Schematic diagram of HMM-LR speech recognizer.

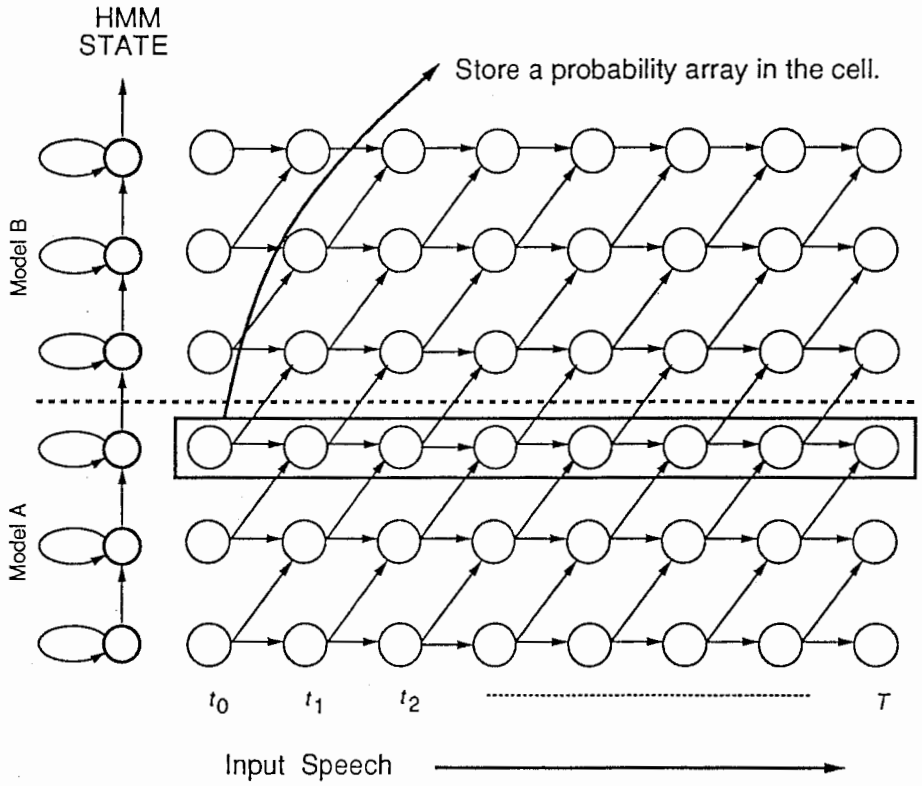


Figure 2.8: Stacking of a probability array.

2.4.3 Algorithm

This subsection gives the algorithm for HMM-LR as a recognizer, which produces no parse trees. It is, however, easy to extend the algorithm to produce parse trees.

First, we introduce a data structure called a *cell*. A cell is a structure with information about one recognition candidate. The following are kept in the cell.

- *LR parsing stack*, with information for parsing control.
- *Input stack*, which includes a recognition hypothesis.
- *Probability array*, which includes end point candidates and their probabilities.

The algorithm is summarized below.

1. *Initialization.* Create a new cell C. Push the LR initial state 0 on top of the LR parsing stack of C. Initialize the probability array Q of C;

$$Q(t) = \begin{cases} 1 & t = 0 \\ 0 & 1 \leq t \leq T \end{cases} \tag{2.9}$$

2. *Ramification of cells.* Construct a set

$$\begin{aligned} S \;=\; & \{(C, s, a, x) | \exists C, s, a, x (C \text{ is a cell which is not accepted} \\ & \& \; s \text{ is the state on top of the LR parsing stack of } C \\ & \& \; x = Action[s, a] \\ & \& \; x \neq \text{“}error\text{”})\} \end{aligned} \tag{2.10}$$

For each element $(C, s, a, x) \in S$, do operations below. If a set S is empty, parsing is completed.

3. If $x =$“*shift* s'”, verify the existence of phone a. In this case, update the probability array Q of the cell C using the following computation.

$$\alpha_i(t) \;=\; \begin{cases} Q(t) & i = S_I \\ 0 & i \neq S_I \;\&\; t = 0 \\ \sum_j \alpha_j(t-1) a_{ji} b_{ji}(y_t) & i \neq 0 \;\&\; t > 0 \end{cases} \tag{2.11}$$

$$Q(t) \;=\; \begin{cases} 0 & t = 0 \\ \alpha_{S_F}(t) & t > 0 \end{cases} \tag{2.12}$$

If $\max_{1 \leq t \leq T} Q(t)$ is below a certain threshold, cell C is abandoned. Else push s' on top of the LR parsing stack of cell C, and push a onto the input stack of C.

4. If x = "*reduce* $A \rightarrow \beta$", pop $|\beta|$ symbols off the LR parsing stack and push $Goto[s', A]$ where s' is the current state on top of the stack.

5. If x = "*accept*" and $Q(T)$ exceeds a certain threshold, cell C is accepted. If not, cell C is abandoned.

6. Return to 2.

Recognition results are kept in accepted cells. Generally, many recognition candidates exist, and it is possible to rank these candidates using a value $Q(T)$ of each cell.

2.4.4 Refinements of the Algorithm

The algorithm described above is a simple one. It is possible to make some refinements to the algorithm.

1. **Using the beam-search technique.**
   The *beam-search* technique was first used in the HARPY speech recognition system [Lowerre 80]. It is a modification of the breadth-first search technique, in which a group of near-miss alternatives around the best path are selected and processed in parallel. The beam-search technique reduces search cost and maintains search efficiency. Generally, a set S constructed in step 2 in the algorithm is quite large. The beam-search technique can be used to select a group of likely cells. The value $\max_{1 \leq t \leq T} Q(t)$ of each cell can be used as an evaluation score.

2. **Using a graph-structured stack.**
   The graph-structured stack is one of the key ideas in generalized LR parsing. In the above algorithm, when making a set S, copies of an LR parsing stack are created. By using a graph-structured stack, it is not necessary to copy the whole stack. Copying only the necessary portion of the stack is sufficient. Thus, the amount of computation is reduced.

3. **Using HMM duration control.**
   It is practical to limit minimum and maximum duration for each HMM state. Accordingly, the computation in step 3 is sufficient for t such that

$$\sum_{s \in S} D_{min}(s) \leq t \leq min \left(\sum_{s \in S \cup \{s'\}} D_{max}(s),\ T \right) \tag{2.13}$$

   Here, $D_{min}(s)$ and $D_{max}(s)$ are a minimum and maximum duration for a state s, respectively. S is a set of states previously verified, and s' is the state that will be verified next.

2.5 Conclusion

In this chapter, we have proposed an efficient speech recognition algorithm, the HMM-LR algorithm, using Hidden Markov Models and generalized LR parsing. This algorithm allows a tight coupling of speech recognition and natural language processing and has the following advantages:

1. It can deal with grammatical constraints based on a context-free grammar.
2. In it, the LR parser is used not only for parsing but also a languge source model for symbol prediction/generation.
3. In it, the LR parser drives Hidden Markov Models directly without any intervening structures such as a phone/word lattice. Thus, very accurate, efficient speech recognition is achieved.

Chapter 3

Implementations of the HMM-LR Speech Recognition System

3.1 Introduction

This chapter describes implementations of the HMM-LR speech recognition system, which is based on the HMM-LR algorithm proposed in the previous chapter. Two versions of the HMM-LR speech recognition system are developed:

- **The baseline version.** [Kita 89a, Kita 90b]
  The first version employs basic HMM techniques, and uses only LPC cepstral parameters in one codebook. HMMs are duration controlled, but their mechanisms are very simple.

- **The improved version.** [Kita 90a, Hanazawa 90a, Hanazawa 90b]
  The second version uses enhanced HMM phone models. Multiple codebooks (separate vector quantization) [Gupta 87], accurate HMM state duration control [Hanazawa 90a, Hanazawa 90b], and fuzzy vector quantization [Tseng 87] are used. A speaker adaptation algorithm based on fuzzy VQ codebook mapping [Nakamura 89] is also incorporated. This version will be used in the succeeding chapters for the language model evaluation experiments.

In both versions, HMM phone models are trained using isolated word utterances from the *ATR speech database* [Takeda 88, Kuwabara 89]. We will describe a series of recognition experiments using Japanese phrase-wise utterances from the *conference registration task*. We will also show the feasibility of the HMM-LR in very large vocabulary (several thousand words) speech recognition.

3.2 Measure of Task Complexity

In order to compare different systems or different language models, it is necessary to have a common measure of the complexity of a speech recognition task. *Perplexity* [Bahl 83, Jelinek 90] is a widely used measure, and it means the information theoretic average branching factor. Usually, word-unit perplexity is used for English speech recognition systems. However, word-unit perplexity has the following disadvantages [Kawabata 89a, Nakagawa 90]:

1. In agglutinative languages like Japanese, the definition of words is sometimes ambiguous.
2. The length of sentences is not taken into account.

In this section, two kinds of measures are introduced to represent language model quality. They are *task entropy* and *phone perplexity* [Kawabata 89a].

3.2.1 Task Entropy

Let S be a sentence in a language L whose generation probability is $P(S)$. Then the entropy $H(L)$ of the language L is given by

$$H(L) = -\sum_{S} P(S) \log_2 P(S). \tag{3.1}$$

Generally, since the language L generates infinite sentences, $H(L)$ is not calculated directly. However, given a set of sample sentences, we can estimate $H(L)$.

1. Compile statistics on the sentence length from the sample sentences. Let P_k be the probability that a sentence is of length k.
2. Generate all the sentences of length k. Let N_k be the number of sentences of length k.
3. Task entropy is calculated as follows:

$$\begin{aligned} H(L) &= -\sum_{S} P(S) \log_2 P(S) \\ &= -\sum_{S} \frac{P_k}{N_k} \log_2 \frac{P_k}{N_k} \\ &= -\sum_{k} N_k \frac{P_k}{N_k} \log_2 \frac{P_k}{N_k} \\ &= -\sum_{k} P_k \log_2 \frac{P_k}{N_k} \end{aligned} \tag{3.2}$$

3.2.2 Phone Perplexity

Let $K(S)$ be the number of phones in the sentence S. Then the per-phone entropy is calculated as follows:

$$\begin{aligned} H_{phone}(L) &= -\sum_S \frac{1}{K(S)} P(S) \log_2 P(S) \\ &= -\sum_S \frac{1}{K(S)} \frac{P_k}{N_k} \log_2 \frac{P_k}{N_k} \\ &= -\sum_k N_k \frac{1}{k} \frac{P_k}{N_k} \log_2 \frac{P_k}{N_k} \\ &= -\sum_k \frac{P_k}{k} \log_2 \frac{P_k}{N_k} \end{aligned} \tag{3.3}$$

The phone perplexity $F_p(L)$ is given by

$$F_p(L) = 2^{H_{phone}(L)}. \tag{3.4}$$

3.3 Task and Grammars

This section describes the task and grammars we use for evaluation experiments. Evaluation is carried out using Japanese phrase-wise utterances from the *conference registration task*. This task consists of dialogues between secretaries and participants of international conferences. Some example phrases in this task are shown in Table 3.1.

Two kinds of grammars were developed: a *general grammar* [1] and a *task-specific grammar*. The general grammar was designed to cover many linguistic phenomena common in Japanese, while the task-specific grammar was designed to cover only linguistic phenomena in our task.

Table 3.1: Example phrases in the conference registration task.

/daiikkai/ /tsuuyaku/ /deNwa/ /kokusai/ /kaigini/ /saNkano/ /tourokuo/ /gokibousareru/ /katawa/ /shoteino/ /moushikomi/ /youshini/ $\cdots$
(*You can make a registration for the first international conference on interpreting telephony* $\cdots$)

[1] The rules of the general grammar are listed in Appendix A.1.

We have a large-scale linguistic database, the *ATR linguistic databse* [Ehara 90], including dialogues on the conference registration task. The task-specific grammar was created by examining actual word sequence constraints in this database.

Both grammars describe Japanese phrase structure, and each is written in the form of a context-free grammar. Phonetic spellings for each word are also included in the grammars. The grammars are pre-compilied into LR parsing tables in advance. The size and complexity of the grammars are shown in Table 3.2. [2]

Table 3.2: Size and complexity of the grammars.

Grammar	*General*	*Task-Specific*
Rule	1,461 rules	582 rules
Vocabulary	1,035 words	275 words
Task entropy	17.0	13.2
Phone perplexity	5.9	3.9
Estimated word perplexity	more than 100	more than 30

3.4 Speech Data

Japanese phrases and isolated words from the *ATR speech database* [Takeda 88, Kuwabara 89] are used for recognition experiments and evaluation. The data consists of 4 speakers, three male (MAU, MNM, MHT) and one female (FSU). All speech data were uttered by professional announcers and narrators, recorded in noise-free environments, and phonetically labeled by hand. This data is used for training and testing as described below.

- **HMM training data.**
  Phones extracted from 5,240 important Japanese words, and 216 phonetically balanced words are used for HMM training.

- **Speaker adaptation data.**
  25 or 100 words are extracted from 216 phonetically balanced words for speaker adaptation based on fuzzy codebook mapping.

- **Test data.**
  279 Japanese phrases (25 sentences) from the database refered to as *Task SB3* [3] are used for testing. The data is uttered phrase by phrase.

[2]The vocabulary size varies according to a way of word counting. There is a grammar rule, $A \rightarrow \alpha B \beta$ where $B \in N$ and $\alpha, \beta \in T^*$. In such a case, we considered α and β to be two words.

[3]Task SB3 really contains 281 phrases. However, two utterances of "ATR" are excluded from the test data because the correspondent phone does not exist in Japanese.

The speech data is sampled at 12 KHz, pre-emphasized with a filter whose transform function is $(1 - 0.97z^{-1})$, and windowed using a 256-point Hamming window every 3 msec (9 msec in the improved system). Then, 12-order LPC analysis is carried out, and finally the VQ code sequence is generated. For VQ codebook generation, 216 phonetically balanced words are used.

3.5 Baseline HMM-LR System

3.5.1 System Overview

The baseline HMM-LR system uses only LPC cepstral parameters in one codebook, and context-independent phone models. It also uses the Viterbi algorithm for phone verification.

We classified Japanese phones into two categories: *transient phones* and *stationary phones*. The transient phone model has 4 states and 3 loops to represent its time structure. The stationary phone model has 2 states and 1 loop.

Figure 3.1 shows the structure of phone models, and Table 3.3 lists the set of phones used in the baseline system. In Table 3.3, /N/ and /Q/ represent the syllabic nasal and a silence, respectively.

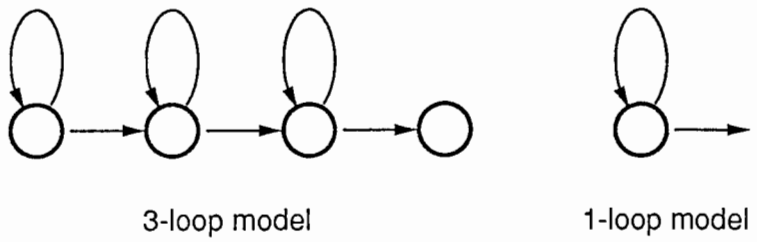


Figure 3.1: Structure of phone models.

Table 3.3: List of phones used in the baseline HMM-LR system.

Structure	*Phones*
3-loop model	/b/, /d/, /g/, /p/, /t/, /k/, /m/, /n/, /ng/, /r/, /z/, /ch/, /ts/, /y/, /w/, /gy/, /hy/, /ky/, /my/, /ngy/, /ny/, /py/, /ry/, /zy/
1-loop model	/i/, /e/, /a/, /o/, /u/, /N/, /ii/, /ee/, /aa/, /oo/, /uu/, /ei/, /ou/, /s/, /sh/, /h/, /Q/

Each phone model is duration controlled with minimum and maximum loop limits. As described above, HMM phone models were trained using the word utterances, whereas the recognition is carried out for continuous speech. To realize accurate duration control, HMM state duration parameters were modified according to the speaking rates of word and phrase utterances [Kawabata 89b].

3.5.2 Recognition Experiments

The baseline HMM-LR system was tested for one male speaker (MAU). Tables 3.4 and 3.5 show phrase recognition rates for several beam widths, using the general grammar or the task-specific grammar. As shown in these tables, a larger beam width makes the system more accurate. It stands to reason that the recognition rate is higher using the task-specific grammar than using the general grammar, because the perplexity of the former is lower.

The baseline system attains rates of 72.0% for the top candidate and 95.3% for the top five candidates using the general grammar. Using the task-specific grammar, rates are 79.9% and 98.6%, respectively. From these results, we can say that the baseline system already attains reasonable performance.

Table 3.4: The baseline HMM-LR results using the general grammar.

	Phrase Recognition Rate (%)				
Rank	*beam* = 5	*beam* = 10	*beam* = 20	*beam* = 50	*beam* = 100
1	38.4	56.6	66.0	71.7	72.0
~ 2	44.8	67.0	78.9	86.0	85.7
~ 3	44.8	69.9	82.4	90.3	92.1
~ 4	44.8	70.6	84.2	92.8	94.3
~ 5	44.8	70.6	85.0	93.4	95.3

Table 3.5: The baseline HMM-LR results using the task-specific grammar.

	Phrase Recognition Rate (%)			
Rank	*beam* = 5	*beam* = 10	*beam* = 20	*beam* = 30
1	72.4	77.4	80.3	79.9
~ 2	82.1	88.2	91.8	92.8
~ 3	83.9	91.4	94.9	96.1
~ 4	84.2	92.1	96.8	97.5
~ 5	84.2	92.8	97.5	98.6

3.5.3 Comparison with the Lattice Parsing Approach

One of the advantages of the HMM-LR algorithm is that speech recognition and natural language processing are tightly coupled. In order to demonstrate this advantage, we compared the HMM-LR algorithm with lattice parsing. Lattice parsing is the serial coupling of speech recognition and natural language processing.

In the lattice parsing experiment, we used HMMs and LR parsing independently. That is, HMM phone spotting is used to produce a phone lattice, then the LR parsing is applied to obtain phrase candidates. An HMM spotting algorithm and a lattice parsing algorithm using generalized LR parsing are presented in [Kawabata 89b] and [Saito 88a], respectively.

Table 3.6 shows recognition results. Phone lattice parsing attains only the phrase recognition rate of 51%, while the HMM-LR attains a rate of 72%. These results shows the superiority of the HMM-LR.

Table 3.6: Phone lattice parsing vs. HMM-LR.

Rank	*Phrase Recognition Rate* (%)	
	Phone Lattice Parsing	*HMM-LR*
1	51	72
~ 5	82	95
~ 10	85	-
~ 30	87	-

3.6 Improved HMM-LR System

This section describes the improvement of the HMM-LR speech recognition system. The improved HMM-LR system uses the forward algorithm for phone verification, instead of the Viterbi algorithm. Since the Viterbi algorithm is an approximation of the forward algorithm, it is better to use the forward algorithm for computing the acoustic probability precisely.

3.6.1 HMM Phone Models

Table 3.7 shows a list of phones used in the improved system. Vowels, the syllabic nasal and a silence are modeled as a 1-loop model, and other phones as a 3-loop model. In Table 3.7, phones suffixed with a '1' or '3' indicate that these phones are used at the beginning or ending portions of speech. /Ui/ and /Uu/ represent devocalized /i/ and /u/, respectively.

Table 3.7: List of phones used in the improved HMM-LR system.

Structure	*Phones*
3-loop model	/s/, /sh/, /h/, /z/, /ch1/, /ch/, /ts1/, /ts/, /p1/, /p/, /t1/, /t/, /k1/, /k/, /b1/, /b/, /d1/, /d/, /g/, /ng/, /m/, /n/, /r1/, /r/, /w/, /y/ /sy/, /hy/, /zy/, /cy/, /py/, /ky/, /by/, /gy/, /ngy/, /my/, /ny/, /ry/
1-loop model	/N/, /N3/, /a/, /a3/, /i/, /i3/, /Ui/, /Ui3/, /u/, /u3/, /Uu/, /Uu3/, /e/, /e3/, /o/, /o3/, /aa/, /aa3/, /ii/, /ii3/, /uu/, /uu3/, /ee/, /ee3/, /oo/, /oo3/, /ei/, /ei3/, /ou/, /ou3/, /Q/

3.6.2 Multiple Codebooks

To represent phone models with less distortion, *multiple codebooks* (*separate vector quantization*) [Gupta 87] are used, where the following three parameters are vector-quantized separately:

1. Spectrum (WLR),
2. LPC cepstral difference,
3. Power.

In the training stage, the output vector probabilities of these three codebooks are estimated simultaneously and independently. In the recognition stage, the output probability of emitting multiple symbols is computed as the product of the output vector probabilities in these codebooks. That is, the forward calculation α is replaced with Equation 3.5:

$$\alpha_i(t) = \sum_j \left\{ \alpha_j(t-1) a_{ji} \prod_{k=1}^{K} b_{ji}^{(k)}(y_t) \right\} \tag{3.5}$$

where K is the codebook size.

3.6.3 Duration Control

HMMs are effective in expressing speech data statistically, but phone duration information from speech data is not modeled statistically in the HMM phone models. In order to make a statistical duration model, an HMM state duration control algorithm is introduced as follows [Hanazawa 90a, Hanazawa 90b]:

1. After the HMM training, the HMM state duration distribution is determined by the Viterbi alignment on training data. Then each state distribution is approximated by the Gaussian distribution.

2. In the recognition stage, state duration control is carried out by adding a state duration penalty. The modified forward probability is calculated as follows:

$$\alpha_i(t) = \sum_j \sum_\tau \alpha_j(t-\tau-1) a_{ji} b_{ji}(y_{t-\tau}) \left(\prod_{k=t-\tau+1}^{t} a_{ii} b_{ii}(y_k) \right) d(j,\tau)^w \tag{3.6}$$

where $d(j,\tau)$ is a state duration distribution, and w is a constant.

As is stated in the baseline system, HMM duration parameters are modified according to the speaking rates of word and phrase utterances.

3.6.4 Fuzzy Vector Quantization

Fuzzy vector quantization (fuzzy VQ) [Tseng 87] represents an input vector as a linear combination of code-vectors. Fuzzy VQ makes it possible to represent various kinds of vectors beyond the limitation caused by the codebook size, and to approximate input vectors more precisely than conventional VQ. Fuzzy VQ is defined as follows:

$$x \longrightarrow \sum_{i=1}^{k} m_i v_i \tag{3.7}$$

$$m_i = 1 / \left[\sum_{j=1}^{k} (d_i/d_j)^{1/(F-1)} \right] \tag{3.8}$$

where x is an input vector,
v_i is a code-vector,
m_i is the fuzzy membership function,
$d_i = ||x - v_i||$ (distance between x and v_i),
k is the number of code-vectors,
F is the degree of fuzziness.

In our case, the k-nearest neighbor rule is adopted from the viewpoint of computation reduction. The values of k and fuzziness F are set at 6 and 1.6, respectively. Since there are plenty of training data, hard vector quantization without fuzzy VQ is performed for HMM training. Fuzzy VQ is used in the recognition stage.

3.6.5 Speaker Adaptation

Two speaker adaptation algorithms are implemented in the improved system. One is based on fuzzy VQ codebook mapping, and the other is based on composite modeling of an adapted model and a speaker-dependent model.

Fuzzy VQ Codebook Mapping

A speaker adaptation algorithm based on fuzzy VQ codebook mapping is introduced as follows [Nakamura 89]:

1. **Make VQ codebooks.**
   A new speaker A and a standard speaker B utter the same words (adaptation training words). Multiple VQ codebooks are generated using the training word utterances from each speaker.

2. **Make a correspondence histogram.**
   The training word utterances are vector-quantized using fuzzy VQ. The optimal correspondence path between the same word utterances from speakers A and B are determined using dynamic time warping. A histogram is made by accumulating the correspondence between speaker A's and speaker B's codevectors.

3. **Make adapted models.**
   Using the correspondence histogram, the output probabilities for speaker A are calculated as follows:

$$b_i^{(a)} = \sum_{j=1}^{K} h_{ij} b_j \tag{3.9}$$

where $b_i^{(a)}$ is the output probability for A,
b_j is the output probability for B,
h_{ij} is the correspondence histogram,
K is the codebook size.

Composite Models

After the speaker adaptation procedure based on codebook mapping, the adapted models are re-trained by *HMM phone concatenated training* [Maruyama 89], using the training word utterances. The composite models are obtained by combining the newly trained parameters $b^{(d)}$ and the adapted model's parameters $b^{(a)}$.

$$b^{(c)} = wb^{(a)} + (1-w)b^{(d)} \tag{3.10}$$

3.6.6 Recognition Experiments

The improved HMM-LR speech recognition system was tested for four speakers (three male, one female) in the speaker-dependent condition, using the general grammar. [4] The result is shown in Table 3.8. The average phrase recognition rate was 89.5%, and a rate of 99.2% was achieved for the top five choices.

The system was also tested in the speaker-adapted condition. In the experiments, one male speaker (MHT) was selected as a standard speaker and the others (two male and one female) were used for testing speakers. 25 or 100 words were used for speaker adaptation training data. Table 3.9 shows the result without speaker adaptation, and Tables 3.10, 3.11 and 3.12 show the results for the speaker-adapted condition using 25 words, using 100 words, and with composite models using 100 words, respectively. Using composite models, in the case of 100 words, the average phrase recognition rate was 81.6%, and a rate of 98.0% was achieved for the top five choices. Figure 3.2 shows a comparison of recognition rates under these various conditions.

Table 3.8: The improved HMM-LR results in the speaker-dependent condition.

	Phrase Recognition Rate (%)				
Rank	MAU (male)	MNM (male)	MHT (male)	FSU (female)	Average
1	88.2	89.6	88.5	91.7	89.5
~ 2	98.6	94.2	95.7	97.1	96.4
~ 3	99.3	96.8	98.9	99.3	98.6
~ 4	99.3	97.8	98.9	100.0	99.0
~ 5	99.3	98.6	98.9	100.0	99.2

Table 3.9: The improved HMM-LR results, without adaptation.

	Phrase Recognition Rate (%)			
Rank	MAU (male)	MNM (male)	FSU (female)	Average
1	66.3	73.7	38.8	59.6
~ 2	79.9	88.5	49.3	72.6
~ 3	88.2	93.5	55.8	79.2
~ 4	91.4	94.2	59.7	81.8
~ 5	93.2	95.7	63.7	84.2

[4] Appendix A.2 contains the recognition results for the speaker MAU.

Table 3.10: The improved HMM-LR results, speaker-adapted using 25 words.

Rank	*Phrase Recognition Rate* (%)			
	MAU (male)	MNM (male)	FSU (female)	Average
1	72.8	78.8	73.7	75.1
~ 2	84.6	92.4	88.1	88.4
~ 3	91.0	96.0	90.6	92.5
~ 4	93.5	97.1	93.2	94.6
~ 5	95.3	97.5	94.2	95.7

Table 3.11: The improved HMM-LR results, speaker-adapted using 100 words.

Rank	*Phrase Recognition Rate* (%)			
	MAU (male)	MNM (male)	FSU (female)	Average
1	76.3	79.5	79.9	78.7
~ 2	91.4	91.7	91.7	91.6
~ 3	95.7	95.7	93.9	95.1
~ 4	97.5	97.1	95.7	96.8
~ 5	97.8	97.8	96.4	97.3

Table 3.12: The improved HMM-LR results, speaker-adapted with composite models using 100 words.

Rank	*Pharse Recognition Rate* (%)			
	MAU (male)	MNM (male)	FSU (female)	Average
1	78.1	83.1	83.5	81.6
~ 2	91.8	93.2	92.4	92.5
~ 3	96.8	96.0	94.6	95.8
~ 4	97.8	97.1	97.1	97.3
~ 5	98.6	97.5	97.8	98.0

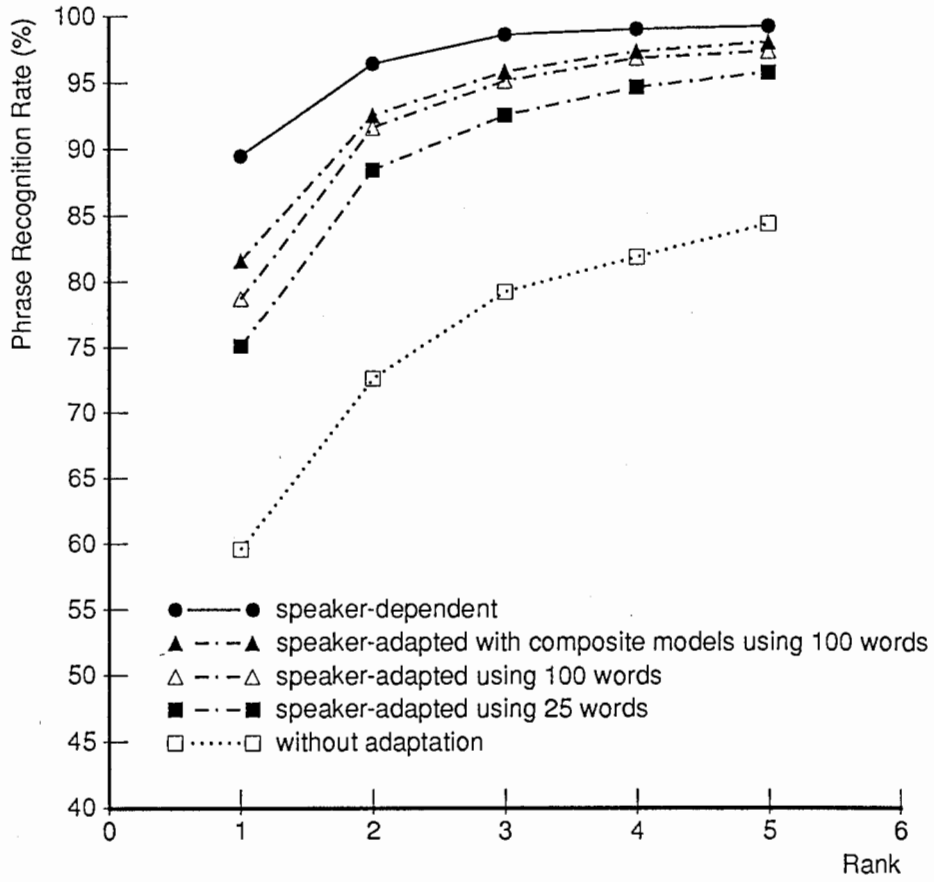


Figure 3.2: Comparison of phrase recognition performances.

3.7 Very Large Vocabulary Recognition

We already proved that the HMM-LR speech recognition system can attain high recognition performance for a task including 1,000 words. In this section, we show the feasibility of the HMM-LR in very large vocabulary speech recognition [Kita 91b]. Here, *very large vocabulary* means a vocabulary of several thousand words or more. We investigate the relationship between vocabulary size and various factors, such as recognition rates, LR parsing table size and phone perplexity.

As the basis of experiments, we use 353 Japanese phrase utterances refered to as the *Model Conversation*, and an intra-phrase grammar which we will hereafter call the *base grammar*. The base grammar includes 744 words and has phone perplexity of 3.57. The speech data and the grammar are further described in Chapter 5.

3.7.1 Vocabulary

The vocabulary was derived from a Japanese word dictionary and the ATR linguistic database. We extracted 2,050 frequently used words in the linguistic database, and 5,140 commonly used words in the dictionary. These words were then converted to sequences of phonetic symbols. Table 3.13 shows the number of words and the number of different words from each linguistic source.

3.7.2 Recognition Experiments

In the experiments, we randomly selected words from each linguistic source, and added these words to the base grammar. However, a recognition rate depends on a word set, thus we repeated experiments three times for each vocabulary size and calculated the average.

Figure 3.3 shows the relationship between vocabulary size and phrase recognition rates. As vocabulary size increases, the recognition rate decreases. However, the system attained over 95% for the top five choices in any vocabulary size. We also carried out recognition experiments using the HMM-LR system enhanced with the trigram model of syllables, which will be further described in Chapter 4. This system attained over 85% for the top choice in any vocabulary size.

Figure 3.4 shows the relationship between vocabulary size and LR table size, where LR table size means the number of states in the table, and Figure 3.5 indicates the relationship between vocabulary size and phone perplexity. Even if vocabulary size increases, phone perplexity does not increase steeply, but increases gradually. This suggests the feasibility of the HMM-LR speech recognition system in very large vocabulary speech recognition of more than 8,000 words.

Table 3.13: Vacabulary size of each linguistic source.

	Number of Words	*Number of Different Words*
Base grammar	744	497
Frequently used words in LDB	2,050	1,747
Common words in dictionary	5,140	4,445
Total	7,934	6,689

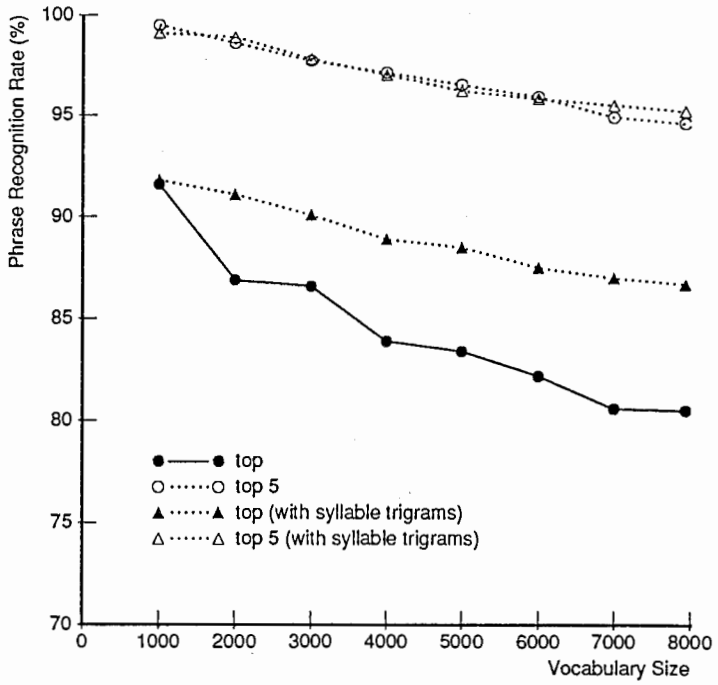


Figure 3.3: Relationship between vocabulary size and recognition rates.

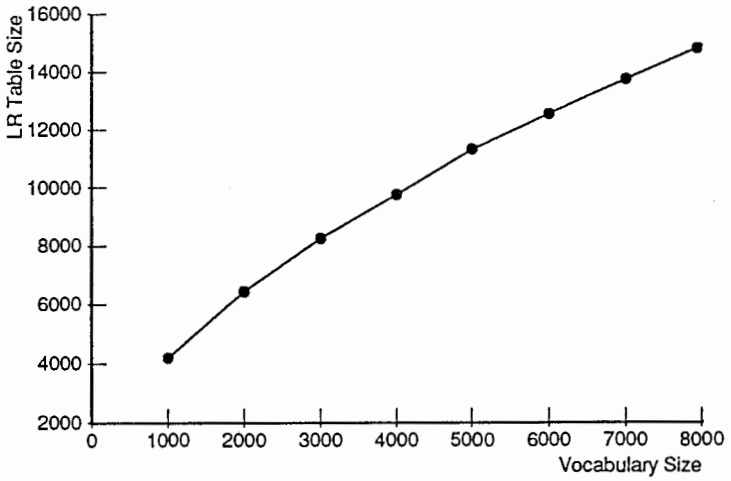


Figure 3.4: Relationship between vocabulary size and LR table size.

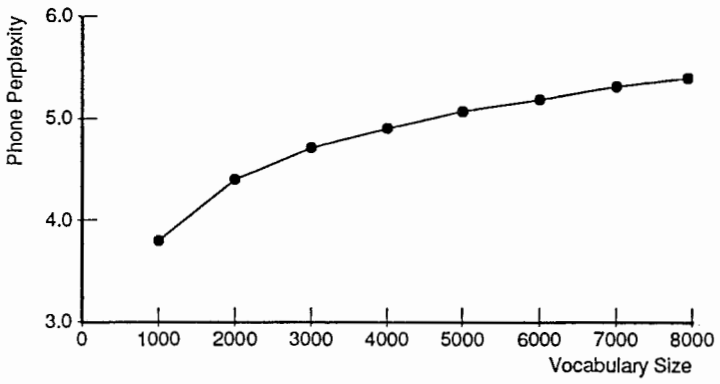


Figure 3.5: Relationship between vocabulary size and phone perplexity.

3.8 Conclusion

In this chapter, we first introduced two measures of complexity of a language model, task entropy and phone perplexity, and presented a method for computing them. These measures are suitable for agglutinative languages like Japanese.

We then described two implemantations of the HMM-LR speech recognition system: the baseline HMM-LR system and the improved HMM-LR system.

The baseline HMM-LR system uses basic HMM techniques. It uses only LPC cepstral parameters in one codebook. This system attained a phrase recognition rate of 72.0% using the general grammar, which has phone perplexity of 5.9. Using the task-specific grammar with a phone perplexity of 3.9, the system attained a phrase recognition rate of 79.9%.

The improved HMM-LR system uses various techniques, such as multiple codebooks, accurate HMM state duration control, and fuzzy vector quantization. A speaker adaptation algorithm based on fuzzy VQ codebook mapping was also incorporated. The improved system was tested for four speakers, using the general grammar. In the speaker-dependent condition, the average phrase recognition rate was 89.5%, and a rate of 99.2% was achieved for the top five choices. In the speaker-adapted condition, rates were 81.6% and 98.0%, respectively.

We also demonstrated the superiority of the HMM-LR algorithm by comparing it with lattice parsing, and showed the feasibility of the HMM-LR in very large vocabulary speech recognition. All these results show that the HMM-LR speech recognition algorithm is very effective in continuous speech recognition.

Chapter 4

Stochastic Language Models

4.1 Introduction

In chapter 2, we have discussed speech recognition using syntactic language models. The syntactic model defines a set of sentences that are acceptable by the grammar. Although the syntactic model is effective in search space reduction, it ignores the stochastic characteristics of a language.

In this chapter, we will discuss stochastic language models. The goal of stochastic language models is to assign the *a priori* probabilities to any word/phone sequences. These probabilities, together with acoustic probabilities, guide the search of a speech recognition system. Many stochastic language models, including word bigram/trigram models and stochastic grammars, have been proposed [Bahl 83, Shikano 87, Lee 89, Paeseler 89].

The following three models are investigated [Kita 90c, Kita 91c]:

1. Trigram model of Japanese syllables.
2. Stochastic LR parsing.
3. Trigram model of context-free rewriting rules.

The first model utilizes the underlying stochastic structure of syllable sequences, and is effective in correcting local recognition errors. The second and third models provide hybrid modeling of a language, in which stochastic models augment syntactic models quantitatively.

These stochastic models are trained using a large number of texts from the *ATR linguistic database* [Ehara 90]. The models are incorporated into the HMM-LR speech recognition system, and evaluated by perplexity reduction and phrase recognition experiments.

4.2 Trigram Model of Japanese Syllables

4.2.1 Trigram Model

An N-gram language model is an extremely rough approximation of a language, but it is effective in correcting local syntax errors. Actually, word bigram/trigram models are extensively used to correct recognition errors and improve recognition accuracy [Bahl 83, Shikano 87, Lee 89, Paeseler 89].

The general idea of a trigram model can easily be applied to Japanese syllables. A typical syllable in Japanese is in the form of a CV, namely one consonant followed by one vowel, and the number of syllables is very small (about one hundred). Moreover, Japanese syllables seem to have a special stochastic structure. Araki et al. [Araki 89] suggest that a statistical method based on Japanese syllable sequences is effective for ambiguity resolution in speech recognition systems. Thus, a syllable trigram model is effective for recognizing Japanese syllable sequences.

Syllable trigrams are estimated using a large number of training samples. The training samples, however, cannot cover every actual syllable sequence. Thus, it is necessary to smooth the trigram probabilities. This can be done by interpolating trigram, bigram, unigram, and zerogram probabilities.

In our syllable trigram model, the *a priori* probability $P(S)$ that the syllable sequence $S = s_1, s_2, \ldots, s_n$ will be uttered is calculated as follows.

$$P(s_1, \ldots, s_n) = P(s_1 \mid \#)P(s_2 \mid \#, s_1) \prod_{k=3}^{n} P(s_k \mid s_{k-2}, s_{k-1})P(\# \mid s_{n-1}, s_n) \tag{4.1}$$

$$P(s_k \mid s_{k-2}, s_{k-1}) = q_1 f(s_k \mid s_{k-2}, s_{k-1}) + q_2 f(s_k \mid s_{k-1}) + q_3 f(s_k) + q_4 C \tag{4.2}$$

$$\sum_k q_k = 1 \tag{4.3}$$

$$f(s_k \mid s_{k-2}, s_{k-1}) = \frac{N(s_{k-2}, s_{k-1}, s_k)}{N(s_{k-2}, s_{k-1})} \tag{4.4}$$

In the above expressions, "#" indicates the phrase boundary marker, and C is a uniform probability that each syllable will occur. The function N counts the number of occurrences of its arguments in the training data. q_i are interpolation weights, which will be discussed in the next subsection.

4.2.2 Estimation of Interpolation Weights

The optimal interpolation weights q_i in Equation 4.2 are determined using *deleted interpolation* [Jelinek 80]. This technique automatically determines the optimal weights based on the maximum likelihood approach.

The problem is stated as: given a collection of training data $\{b_j | j = 1, \ldots, K\}$, estimate the interpolation weights q_i that maximize Equation 4.5 under the condition $\sum q_i = 1$.

$$P = \prod_{j=1}^{K} P(b_j) \tag{4.5}$$

The following iterative procedure solves this problem [Kawabata 90]:

1. Make an initial guess of q_i such that $\sum_i q_i = 1$ holds.
2. Calculate i-gram probabilities f_i^j when the j-th data b_j is removed from the training data.
3. Re-estimate q_i by the following formula.

$$\overline{q_i} = \frac{1}{N} \sum_{j=1}^{N} C_i^j \tag{4.6}$$

where N is the number of syllables in training data, and

$$C_i^j = \frac{q_i f_i^j}{\sum_k q_k f_k^j} \tag{4.7}$$

4. Replace q_i with $\overline{q_i}$ and repeat from step 2.

In our task, the average number of syllables in one word is 3.31. This value is not particularly high, which means that syllable trigrams include considerable connective information between words.

4.3 Stochastic LR Parsing

The HMM-LR speech recognition system uses an LR parser. In a traditional LR parser, each shift/reduce action is treated equally. However, some actions occur frequently, others rarely. It is natural to take their frequency of occurrence into account when calculating the recognition likelihood. For that purpose, we introduce stochastic LR parsing. Stochastic LR parsing is closely related to a stochastic context-free grammar.

4.3.1 Stochastic Context-Free Grammar

A *stochastic context-free grammar* [Fu 74, Wright 87] is an extension of a context-free grammar, in which each rewriting rule is of the form $< A \rightarrow \alpha, p >$, where p is a probability associated with the rule (hereafter, we denote the rule probability as $P(\alpha|A)$). The probabilities of all A-productions (rules having A on the LHS) should sum to 1. Figure 4.1 shows an example of a stochastic context-free grammar.

In the stochastic context-free grammar, the probability of a derivation can be computed as the product of the probabilities of the rules used. Suppose that

$$S \stackrel{r_1}{\Longrightarrow} \gamma_1 \stackrel{r_2}{\Longrightarrow} \gamma_2 \stackrel{r_3}{\Longrightarrow} \cdots \stackrel{r_n}{\Longrightarrow} \gamma_n = x \tag{4.8}$$

is a derivation of x from the start symbol S, then the probability of this derivation is given by

$$P(x) = \prod_{i=1}^{n} P(r_i). \tag{4.9}$$

Figure 4.2 illustrates the probability calculation of a derivation tree.

4.3.2 Estimation of Rule Probabilities

The parsing process can be viewed as Markov transitions whose states are sentential forms, so probabilities for rules can be estimated using the forward-backward algorithm from the training data. The actual estimation procedure is as follows [Fujisaki 91]:

1. Make an initial guess of $P(\alpha|A)$ such that $\sum_\alpha P(\alpha|A) = 1$ holds.

2. Parse the i-th sentence. Let D_j^i represent the j-th derivation for the i-th sentence, and $n_j^i(A, \alpha)$ represent the number of times the rule $A \rightarrow \alpha$ is used for deriving D_j^i.

3. Re-estimate $P(\alpha|A)$ by the following formula.

$$\overline{P(\alpha|A)} = \frac{\sum_i C_A^i(\alpha)}{\sum_i \sum_\beta C_A^i(\beta)} \tag{4.10}$$

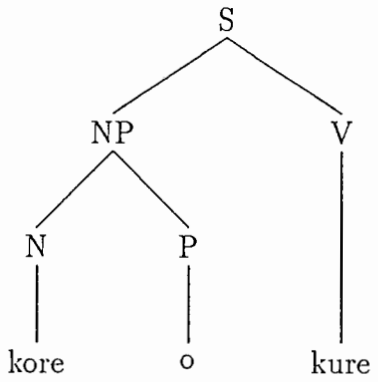
where

$$C_A^i(\alpha) = \sum_i \left(\frac{P(D_j^i)}{\sum_k P(D_k^i)} n_j^i(A, \alpha) \right) \tag{4.11}$$

4. Replace $P(\alpha|A)$ with $\overline{P(\alpha|A)}$ and repeat from step 2.

(1)	S	$\rightarrow$	$NP\ V$	0.7
(2)	S	$\rightarrow$	V	0.3
(3)	NP	$\rightarrow$	N	0.2
(4)	NP	$\rightarrow$	$N\ P$	0.8
(5)	N	$\rightarrow$	$k\ o\ r\ e$	1.0
(6)	P	$\rightarrow$	o	1.0
(7)	V	$\rightarrow$	$k\ u\ r\ e$	0.4
(8)	V	$\rightarrow$	$o\ k\ u\ r\ e$	0.6

Figure 4.1: Stochastic context-free grammar.



$$
\begin{aligned}
&P(kore \cdot o \cdot kure) \\
&\quad = P(NP \cdot V|S) \times P(kure|V) \\
&\qquad \times P(N \cdot P|NP) \times P(o|P) \times P(kore|N) \\
&\quad = 0.7 \times 0.4 \times 0.8 \times 1.0 \times 1.0 \\
&\quad = 0.224
\end{aligned}
$$

Figure 4.2: Probability calculation of the derivation tree.

4.3.3 Stochastic LR Parsing

A stochastic LR parser uses a stochastic LR parsing table in which each action is associated with a probability. A stochastic LR parsing table has the same stochastic information as a stochastic context-free grammar. Using action probabilities, the probability of a partial derivation can be computed, and the probability of a complete derivation is equal to the probability given by a stochastic context-free grammar.

When using a stochastic context-free grammar, the language model probability cannot be calculated until a reduce action is executed. However, a stochastic LR parsing table makes it possible to calculate the language model probability every time one phone is recognized.

Figure 4.3 shows the stochastic LR parsing table created from the grammar in Figure 4.1. Using this stochastic parsing table, the probability for the derivation in Figure 4.2 is given as follows:

$$
\begin{aligned}
& P(kore \cdot o \cdot kure) \\
&= P(s4|0) \times P(s8|4) \times P(s15|8) \times P(s18|15) \times P(r5|18) \\
&\quad \times P(s13|6) \times P(r6|13) \times P(r4|12) \times P(s11|5) \times P(s9|11) \\
&\quad \times P(s16|9) \times P(s19|16) \times P(r7|19) \times P(r1|10) \times P(acc|1) \\
&= 0.82 \times 0.85 \times 1.0 \times 1.0 \times 1.0 \\
&\quad \times 0.8 \times 1.0 \times 1.0 \times 0.4 \times 1.0 \\
&\quad \times 1.0 \times 1.0 \times 1.0 \times 1.0 \times 1.0 \\
&= 0.224
\end{aligned}
$$

where $P(a|s)$ indicates the probability of an action a at state s.

4.3.4 Construction of a Stochastic LR Parsing Table

We will review the stochastic LR parsing table construction algorithm proposed in [Wright 88, Wright 89]. In a stochastic context-free grammar, an item has the form $< A \rightarrow \alpha \cdot \beta, p >$.

Closure Function

If I is an item set, then $Closure(I)$ is constructed from I by the following rules.

1. Every item in I is in $Closure(I)$.
2. If $< A \rightarrow \alpha \cdot B\beta, p >$ is in $Closure(I)$ and $< B \rightarrow \gamma, p' >$ is a rewriting rule, then add the item $< B \rightarrow \cdot\gamma, p'p_B >$ to $Closure(I)$, where p_B is the total probability of items currently in $Closure(I)$ with B after the dot. Repeat this rule until no more new items can be added to $Closure(I)$.

	e	o	u	k	r	$	S	NP	V	P	N
0		(s3, 0.18)		(s4, 0.82)			1	5	2		6
1						(acc, 1.0)					
2						(r2, 1.0)					
3				(s7, 1.0)							
4		(s8, 0.85)	(s9, 0.15)								
5		(s3, 0.6)		(s11, 0.4)					10		
6		(s13, 0.8)		(r3, 0.2)						12	
		(r3, 0.2)									
7			(s14, 1.0)								
8					(s15, 1.0)						
9					(s16, 1.0)						
10						(r1, 1.0)					
11			(s9, 1.0)								
12		(r4, 1.0)		(r4, 1.0)							
13		(r6, 1.0)		(r6, 1.0)							
14					(s17, 1.0)						
15	(s18, 1.0)										
16	(s19, 1.0)										
17	(s20, 1.0)										
18		(r5, 1.0)		(r5, 1.0)							
19						(r7, 1.0)					
20						(r8, 1.0)					

Figure 4.3: Stochastic LR parsing table.

Goto Function

If I is an item set and X is a grammar symbol, then $Goto(I, X)$ is defined to be the closure of the item set

$$\{< A_i \rightarrow \alpha_i X \cdot \beta_i, \frac{p_i}{p_X} >; i = 1, \ldots, n\}$$

where $\{< A_i \rightarrow \alpha_i \cdot X\beta_i, p_i >; i = 1, \ldots, n\}$ is in I, and $p_X = \sum_{i=1}^{n} p_i$.

Canonical Collection Construction

The canonical collection C of item sets is constructed as follows.

1. Initially $C := Closure(< S' \rightarrow \cdot S, 1.0 >)$ (S' is an auxiliary start symbol).
2. Add $Goto(I, X)$ to C for each item set I in C and each grammar symbol X until no further item sets are created.

LR Parsing Table Construction

Finally, an LR parsing table can be constructed from the canonical collection $C = \{I_0, I_1, \ldots, I_n\}$ as follows.

1. If $< A \rightarrow \alpha \cdot a\beta, p_i >$ is in I_i and $Goto(I_i, a) = I_j$, then set $Action[i, a]$ to "*shift* j, p" where p is the sum of the item probabilities in I_i with a after the dot. Here a must be a terminal.
2. If $< A \rightarrow \alpha\cdot, p >$ is in I_i, then set $Action[i, a]$ to "*reduce* $A \rightarrow \alpha, p$" for all a in $Follow(A)$.
3. If $< S' \rightarrow S\cdot, p >$ is in I_i, then set $Action[i, \$]$ to "*accept*, p".
4. If $Goto(I_i, A) = I_j$, then $Goto[i, A] = j$.

The initial state of the parser is the one constructed from the item set containing $< S' \rightarrow \cdot S, p >$.

This algorithm does not work for left-recursive grammars. Fortunately, the algorithm can be generalized to deal with left recursions [Ng 91]. Another solution for the left recursive problem is to remove left recursions from a grammar (see [Aho 86]).

4.4 Trigram Model of Context-Free Rewriting Rules

We use the LR parser as a language source model for symbol prediction or generation. Using the co-occurrence of rewriting rules, it is possible to avoid context-free rewriting rules generating incorrect strings. This mechanism is implemented as a trigram model of context-free rewriting rules.

In this model, if the derivation is given by

$$S \stackrel{r_1}{\Longrightarrow} \gamma_1 \stackrel{r_2}{\Longrightarrow} \gamma_2 \stackrel{r_3}{\Longrightarrow} \cdots \stackrel{r_n}{\Longrightarrow} \gamma_n = x, \tag{4.12}$$

then the probability of this derivation is calculated as follows.

$$\begin{aligned} P(r_1, \ldots, r_n) &= P(r_1 \mid \#)P(r_2 \mid \#, r_1) \\ &\quad \prod_{k=3}^{n} P(r_k \mid r_{k-2}, r_{k-1})P(\# \mid r_{n-1}, r_n) \end{aligned} \tag{4.13}$$

As in the syllable trigram model, a trigram probability is interpolated from bigram, unigram and zerogram probabilities. Also the optimal interpolation weights are determined using deleted interpolation.

In Equation 4.13, the rule sequence $r_1, \ldots, r_n$ is derived from top-down parsing. However, the LR parser is based on bottom-up parsing. Therefore, in our case, the probability of a derivation is defined as $P(r_n, \ldots, r_1)$.

4.5 Evaluation of the Models

To confirm the effectiveness of the stochastic language models, two kinds of evaluations were performed. One is evaluation from the viewpoint of perplexity reduction, and the other is evaluation by recognition experiments. As the basis of the experiments, we used the general grammar described in Chapter 3.

4.5.1 Estimation of Model Parameters

Parameters in the models (syllable trigrams, rule probabilities, etc.) were estimated using a large number of training texts extracted from the *ATR linguistic database* [Ehara 90]. This database contains not only raw texts but also various kinds of syntactic/semantic information, such as *parts of speech*, *pronunciation* and *conjugational pattern*. The training texts include approximately 73,000 phrases and 300,000 syllables. These texts consist of written texts and spoken language texts collected from keyboard or telephone dialogue simulation.

We estimated parameters in the stochastic CFG model and the rule trigram model by actually parsing the phrases in the text database. However, about half of the phrases were not parsed because (1) the text database contained words that did not exist in the grammar, and (2) the definitions of a phrase adopted by the text database and by the grammar were slightly different.

The interpolation weights q_i $(i = 1, \ldots, 4)$ in Equation 4.2 were also estimated using this text database by deleted interpolation. The weights for syllable trigram and rule trigram are shown in Table 4.1.

Table 4.1: Interpolation weights for syllable trigram and rule trigram.

q_i	*Syllable Trigram*	*Rule Trigram*
q_1 (*trigram*)	0.934	0.754
q_2 (*bigram*)	0.052	0.244
q_3 (*unigram*)	0.011	0.001
q_4 (*zerogram*)	0.003	0.001

4.5.2 Evaluation by Perplexity Reduction

To measure the constraint imposed by the language models, we calculated *test-set perplexity* [Jelinek 90]. Test-set perplexity corresponds to the average branching of the language model along the test set.

The test-set perplexity per phone is calculated by

$$L = 2^K \tag{4.14}$$

where K is given by

$$K = -\frac{1}{N_S}\sum_{i=1}^{N_B}\log_2 P(W_i) \tag{4.15}$$

Here N_S is the number of phones in the test set, N_B is the number of test phrases, and $P(W_i)$ is the language model probability of the i-th test phrase W_i.

When only context-free grammar is used, all test sentences are parsed and the number of distinct phones following each phone is counted. Then the test-set perplexity is calculated as the geometric mean of phone choices along the test set.

Test-set perplexity is shown in Table 4.2. With each stochastic language model used, test-set perplexity is reduced from 4.73 to 2.86, 2.53 and 2.54. These results suggest the effectiveness of the stochastic language models.

Table 4.2: Test-set perplexity of the language models.

Model	*Test-set perplexity/phone*
CFG only	4.73
CFG + Trigram model of syllables	2.86
Stochastic CFG	2.53
Trigram model of CFG rules	2.54

4.5.3 Evaluation by Recognition Experiments

The stochastic language models were incorporated into the HMM-LR speech recognition system, and tested by Japanese phrase recognition experiments using 279 phrases uttered by one male speaker MAU.

The HMM-LR speech recognition system uses the beam-search technique to reduce the search space. A group of likely recognition candidates are selected according to the likelihood of each candidate. The likelihood P is calculated as follows:

$$\begin{aligned} P &= (1-\lambda_1-\lambda_2-\lambda_3)P^{(HMM)} \\ &\quad +\lambda_1 P^{(SYLLABLE)} + \lambda_2 P^{(SLR)} + \lambda_3 P^{(RULE)} \end{aligned} \tag{4.16}$$

$P^{(HMM)}$, $P^{(SYLLABLE)}$, $P^{(SLR)}$ and $P^{(RULE)}$ are log likelihoods based on HMM, syllable trigram, stochastic LR parsing and rule trigram, where $P^{(HMM)}$ and $P^{(SYLLABLE)}$ are normalized by the number of frames and syllables, respectively. The scaling parameters λ_i are introduced to adjust the scaling of the four kinds of likelihoods, as determined by preliminary experiments. The values λ_i sre shown in Table 4.3.

The experimental results are shown in Table 4.4. By utilizing syllable trigrams, stochastic LR parsing and rule trigrams, the phrase recognition rate was improved from 88.2% to 92.5%, 92.1%, and 91.4%, respectively. By utilizing all three stochastic language models, a rate of 93.2% was achieved.

Table 4.4 shows that rates for the top two or three choices did not improve with all the models. This is because stochastic language model probabilities for proper nouns were very small as there were few proper nouns in the training texts. If the input speech includes a proper noun such as a person's name, then the likelihood will become worse. This explains why rates for the top two or three choices dropped with all the models.

Tables 4.5 $\sim$ 4.8 show examples of improved recognition results by using stochastic language models.

Table 4.3: Scaling parameters for each language model.

Models	λ_1	λ_2	λ_3
Trigram model of syllables	0.15	0.0	0.0
Stochastic CFG	0.0	0.08	0.0
Trigram model of CFG rules	0.0	0.0	0.02
All the models	0.26	0.05	0.01

Table 4.4: Phrase recognition rates using stochastic language models.

Model 1: Trigram model of Japanese syllables,
Model 2: Stochastic LR parsing,
Model 3: Trigram model of context-free rewriting rules.

Rank	*Phrase Recognition Rate* (%)				
	HMM-LR	With Model 1	With Model 2	With Model 3	With Model 1,2,3
1	88.2	92.5	92.1	91.4	93.2
~ 2	98.6	98.2	98.2	98.6	97.1
~ 3	99.3	99.3	99.6	99.6	98.2
~ 4	99.3	99.3	99.6	99.6	100
~ 5	99.6	99.6	99.6	99.6	100

Table 4.5: Examples of improved recognition results using the trigram model of Japanese syllables.

Phrase No.	*Correct Phrase*	*Top Candidate*	
		Without the model	*With the model*
6	saNkano	saN-dano	saNka-no
48	kaigini	kai-ni	kaigi-ni
53	youshio	youshi-yo	youshi-o
87	tourokuno	touroku-mo	touroku-no
103	syoteino	syotei-eno	syotei-no
104	youshio	youshi-yo	youshi-o
158	kyoutoekikara	kyoutoeki-ka	kyoutoeki-kara
163	chikatetsude	chika-ku-sure	chikatetsu-de
166	sokokara	sou-kara	soko-kara
172	riyou	doyou	riyou
186	machizikaNo	machizikaN-mo	machizikaN-o
217	kaigini	kai-ni	kaigi-ni

Table 4.6: Examples of improved recognition results using stochastic LR parsing.

Phrase No.	*Correct Phrase*	*Top Candidate*	
		Without the model	*With the model*
6	saNkano	saN-dano	saNka-no
44	yoroshiinodesyouka	yoroshi-iNdesyouka	yoroshiinodesyouka
48	kaigini	kai-ni	kaigi-ni
53	youshio	youshi-yo	youshi-o
80	yokousyuudaio	yokousyuudai-yo	yokousyuudai-o
87	tourokuno	touroku-mo	touroku-no
103	syoteino	syotei-eno	syotei-no
104	youshio	youshi-yo	youshi-o
131	youshio	youshi-yo	youshi-o
139	seNkyuuhyakukyuu zyuunananeNno	seNkyuuhyakukyuu zyuunana-neN-o	seNkyuuhyakukyuu zyuunana-neN-no
158	kyoutoekikara	kyoutoeki-ka	kyoutoeki-kara
163	chikatetsude	chika-ku-sure	chikatetsu-de
171	takushiio	takushii-yo	takushii-o
172	riyou	doyou	riyou
217	kaigini	kai-ni	kaigi-ni

Table 4.7: Examples of improved recognition results using the trigram model of context-free rewriting rules.

Phrase No.	*Correct Phrase*	*Top Candidate*	
		Without the model	*With the model*
6	saNkano	saN-dano	saNka-no
44	yoroshiinodesyouka	yoroshi-iNdesyouka	yoroshiinodesyouka
48	kaigini	kai-ni	kaigi-ni
53	youshio	youshi-yo	youshi-o
87	tourokuno	touroku-mo	touroku-no
95	yoroshiinodesyouka	yoroshi-iNdesyouka	yoroshiinodesyouka
103	syoteino	syotei-eno	syotei-no
104	youshio	youshi-yo	youshi-o
163	chikatetsude	chika-ku-sure	chikatetsu-de
186	machizikaNo	machizikaN-mo	machizikaN-o
217	kaigini	kai-ni	kaigi-ni

Table 4.8: Examples of improved recognition results using all the models.

Phrase No.	*Correct Phrase*	*Top Candidate*	
		Without the model	*With all the models*
6	saNkano	saN-dano	saNka-no
42	tetsuzukio	tetsuzuki-yo	tetsuzuki-o
44	yoroshiinodesyouka	yoroshi-iNdesyouka	yoroshiinodesyouka
48	kaigini	kai-ni	kaigi-ni
53	youshio	youshi-yo	youshi-o
80	yokousyuudaio	yokousyuudai-yo	yokousyuudai-o
87	tourokuno	touroku-mo	touroku-no
95	yoroshiinodesyouka	yoroshi-iNdesyouka	yoroshiinodesyouka
103	syoteino	syotei-eno	syotei-no
104	youshio	youshi-yo	youshi-o
131	youshio	youshi-yo	youshi-o
139	seNkyuuhyakukyuu zyuunananeNno	seNkyuuhyakukyuu zyuunana-neN-o	seNkyuuhyakukyuu zyuunana-neN-no
158	kyoutoekikara	kyoutoeki-ka	kyoutoeki-kara
161	syudaNga	syudaN-da	syudaN-ga
163	chikatetsude	chika-ku-sure	chikatetsu-de
166	sokokara	sou-kara	soko-kara
171	takushiio	takushii-yo	takushii-o
172	riyou	doyou	riyou
217	kaigini	kai-ni	kaigi-ni
241	kaigino	kai-no	kaigi-no
260	mata	naka	mata

4.6 Conclusion

In this chapter, three stochastic language models were investigated. The first model is based on a trigram model of Japanese syllables, and this model can capture the local properties of syllable sequences. The second and third models are based on stochastic LR parsing and a trigram model of context-free rewriting rules. They can capture the global syntax of a language.

The effectiveness of these stochastic language models was confirmed by perplexity reduction and recognition experiments. We incorporated these models into the HMM-LR speech recognition system, and successfully improved the initial phrase recognition rate of 88.2% to 93.2%.

Chapter 5

Sentence Recognition Using Two-Level LR Parsing

5.1 Introduction

In Chapter 3, we have implemented the HMM-LR speech recognition system, and successfully applied it to Japanese phrase recognition. Basically, the HMM-LR can deal with grammatical constraints based on a context-free grammar. However, up to now, this system has used the grammatical constraints only within phrases. That is, grammatical constraints between phrases are not considered during speech recognition. To use the HMM-LR system in spoken-language processing applications, it is quite desirable to use between-phrase grammatical constraints which can capture the sentence-level syntax.

In Japanese, a sentence is composed of several phrases, and thus using two kinds of grammars, namely an intra-phrase grammar and an inter-phrase grammar, is sufficient for recognizing sentences. For example, Matsunaga et al. [Matsunaga 90] use an intra-phrase transition network grammar and an inter-phrase dependency grammar to recognize Japanese sentences.

In this chapter, we propose *two-level LR parsing* [Kita 90d, Kita 91e] and apply it to Japanese sentence recognition. Two-level LR parsing is an extension of predictive LR parsing, in which two-level symbol prediction is used: phrase category prediction and phone prediction. Two-level LR parsing makes it possible to use not only intra-phrase grammatical constraints but also inter-phrase grammatical constraints during speech recognition. We also describe an improvement of two-level LR parsing, using a new parsing algorithm called *LR parsing with a Category Reachability Test* (LR-CRT algorithm) [Kita 91g].

5.2 Linguistic Constraints between Phrases

The grammatical structure of Japanese has two levels: *intra-phrase level* and *inter-phrase level.* [1] The word order at the intra-phrase level is fairly constrained. However, the phrase order is said to be much less constrained than in English. This leads many researchers to use the semantic dependency relationship between phrases, called *kakariuke relationship* [Ozeki 87, Matsunaga 90].

The accuracy of a speech recognition system heavily depends on the linguistic knowledge used in the system. However, it has not been clarified what kinds of linguistic knowledge are really effctive for speech recognition, especially at the inter-phrase level. We observed phrase recognition results from the HMM-LR system, and examined the relative ability of each kind of linguistic knowledge [Takezawa 90]. For that purpose, we categorized linguistic knowledge into four groups: syntactic, sematic, pragmatic and contextual knowledge. Because applying the latter (higher-level) knowledge is computationally more expensive, it is assumed that the former knowledge is used at the earlier stage.

Table 5.1 shows the rejection rates of incorrect sentential hypotheses for each kind of linguistic knowledge. Most incorrect hypotheses are filtered out by using syntactic knowledge. This means syntactic constraints can be effectively used at the inter-phrase level.

Table 5.1: Relative ability of linguistic knowledge.

Knowledge	*Ability Ratio* (%)
Syntax	77.1
Semantics	11.5
Pragmatics	0.4
Context	11.0

[1] A Japanese phrase is called a *bunsetsu.* A bunsetsu is a grammatical and phonological unit in Japanese. It consists of an independent word such as a noun, verb or adverb followed by a sequence of zero or more dependent words such as auxiliary verbs, post-positional particles or sentence final particles.

5.3 Two-Level LR Parsing

5.3.1 Inter- and Intra-Phrase Grammars

Two-level LR parsing uses two kinds of grammars: an *inter-phrase grammar* and an *intra-phrase grammar*. The inter-phrase grammar describes grammatical constraints between phrases, while the intra-phrase grammar describes constraints inside phrases. Both grammars are written in the form of a context-free grammar. Examples of grammars are shown in Figures 5.1 and 5.2. In these examples, the inter-phrase grammar shows that S (sentence) is composed of NP (noun phrase) and VP (verb phrase), or that it is composed of a single VP. The intra-phrase grammar describes the construction rules of NP and VP.

The inter- and intra-phrase grammars are automatically compiled into the inter- and intra-phrase LR parsing tables, respectively. The inter-phrase LR parsing table is used for phrase category prediction, and the intra-phrase LR parsing table is used for phone prediction. Examples of LR parsing tables are shown in Figures 5.3 and 5.4.

(1)	S	$\rightarrow$	$NP\ VP$
(2)	S	$\rightarrow$	VP

Figure 5.1: Inter-phrase grammar.

(1)	PH	$\rightarrow$	NP
(2)	PH	$\rightarrow$	VP
(3)	NP	$\rightarrow$	N
(4)	NP	$\rightarrow$	$N\ P$
(5)	VP	$\rightarrow$	V
(6)	N	$\rightarrow$	$k\ o\ r\ e$
(7)	P	$\rightarrow$	o
(8)	V	$\rightarrow$	$k\ u\ r\ e$
(9)	V	$\rightarrow$	$o\ k\ u\ r\ e$

Figure 5.2: Intra-phrase grammar.

	NP	*VP*	$	*S*
0	*s*1	*s*2		3
1		*s*4		
2			*r*2	
3			*acc*	
4			*r*1	

Figure 5.3: Inter-phrase LR parsing table.

	e	*o*	*u*	*k*	*r*	$	*PH*	*NP*	*VP*	*N*	*P*	*V*
0							3					
1				*s*4				6		5		
2		*s*8		*s*7					10			9
3						*acc*						
4		*s*11										
5		*s*12				*r*3					13	
6						*r*1						
7			*s*14									
8				*s*15								
9						*r*5						
10						*r*2						
11					*s*16							
12						*r*7						
13						*r*4						
14					*s*17							
15			*s*18									
16	*s*19											
17	*s*20											
18					*s*21							
19		*r*6				*r*6						
20						*r*8						
21	*s*22											
22						*r*9						

goto-phrase table

Phrase Category	*Initial State*
NP	1
VP	2

Figure 5.4: Intra-phrase LR parsing table.

Unlike a standard LR parsing table, an intra-phrase LR parsing table consists of three parts: an *action table*, a *goto table* and a *goto-phrase table*. The third table is a new one, and decides the state to which the parser should go when a phrase category is given. That is, a goto-phrase table indicates the initial state of the parser for each phrase category. An inter-phrase LR parsing table is the same as a standard LR parsing table.

5.3.2 Speech Recognition Using Two-Level LR Parsing

Two-level LR parsing is an extension of the HMM-LR algorithm, and two-level symbol prediction (phrases category prediction and phone prediction) is used. Figure 5.5 shows a speech recognition system using two-level LR parsing. The part of the figure enclosed with a dotted line shows the HMM-LR phrase recognizer, and the right-hand part is the inter-phrase LR parser.

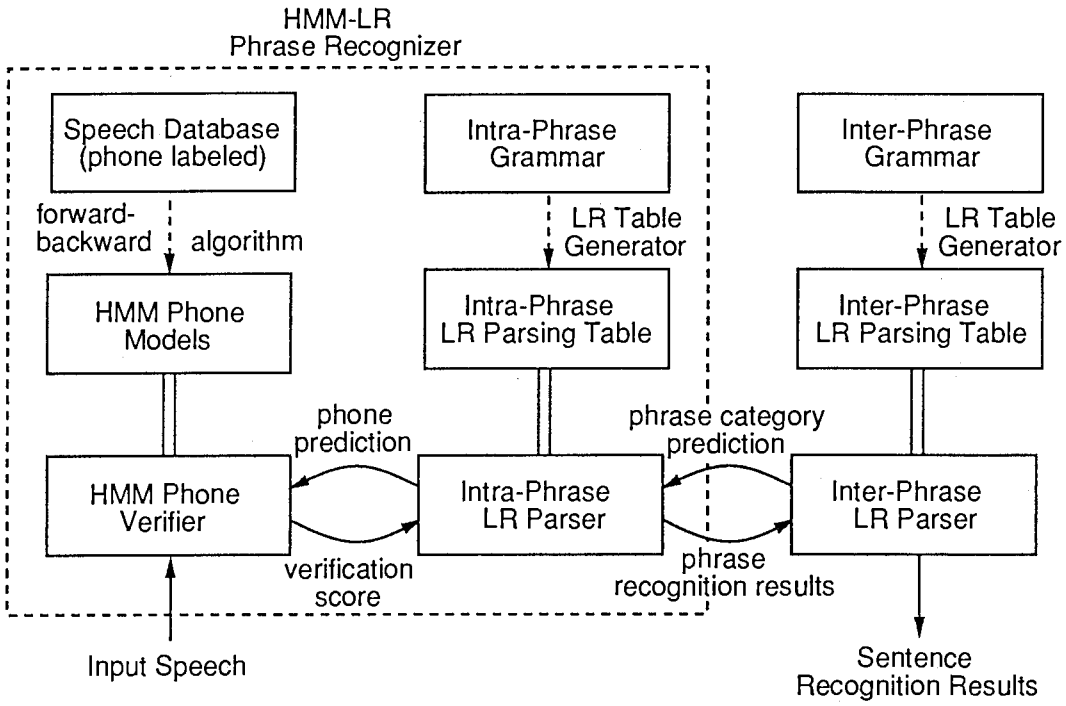


Figure 5.5: Speech recognition system using two-level LR parsing.

The recognition process is summarized below. First, the inter-phrase LR parser predicts the next phrase categories, e.g. *NP* (noun phrase), using the inter-phrase LR parsing table. Next, the intra-phrase LR parser decides the *NP* initial state using the goto-phrase table in the intra-phrase LR parsing table. Then, the HMM-LR speech recognition system recognizes the *NP* candidates. After recognizing the *NP* candidates, the next phrase categories, e.g. *VP* (verb phrase), are predicted using the inter-phrase LR parsing table. The recognition process proceeds in this way until the entire speech data input is processed.

If the intra- and inter-phrase grammars are merged, it is possible to recognize speech with one-level LR parsing. However, two-level LR parsing has the following advantages:

1. Since a recognition unit in two-level LR parsing is a phrase, two-level LR parsing is suitable for recognizing phrase-wise utterances.

2. When using a speech recognition system in real applications, there would be many problems or difficulties, such as *filled pauses*, *interjections*, *restarts*, and *ungrammatical constructions*, which are peculiar to spontaneous speech. [2] Phrases are stable and not overly influenced by these effects. Two-level LR parsing can be thought of as phrase-spotting-based recognition, and it is a promising and robust approach to spontaneous speech recognition.

5.3.3 Construction of an Intra-Phrase LR Parsing Table

This subsection describes how to construct an intra-phrase LR parsing table from a grammar. Here we use *Grammar 1* in Figure 5.6. In this grammar, the start symbol is indicated by *START*, and the phrase categories are *NP* and *VP*.

1. For a rewriting rule $S \rightarrow A$ where S is a start symbol and A is a phrase category symbol, prepare a new terminal symbol $@A$ that does not appear in the grammar, and replace the rewriting rule with $S \rightarrow @A\ A$. In the case of *Grammar 1*, the new terminal symbols are $@NP$ and $@VP$, and we get *Grammar 2*.

2. Construct *Parsing Table 1* from *Grammar 2* using a standard LR parsing table construction algorithm.

3. Remove the new terminal symbols from *Parsing Table 1*. We get *Action and Goto Tables 2-1*.

4. Create *Goto-Phrase Table 2-2* by examining states after new terminal symbols are shifted at the initial state 0.

[2] *Filled pauses* are noises made by the speaker that don't correspond to words such as "ah" and "uh". *Interjections* are extraneous phrases. *Restarts* means repeating a word or phrase. Problems posed by spontaneous speech are discussed in [Ward 89].

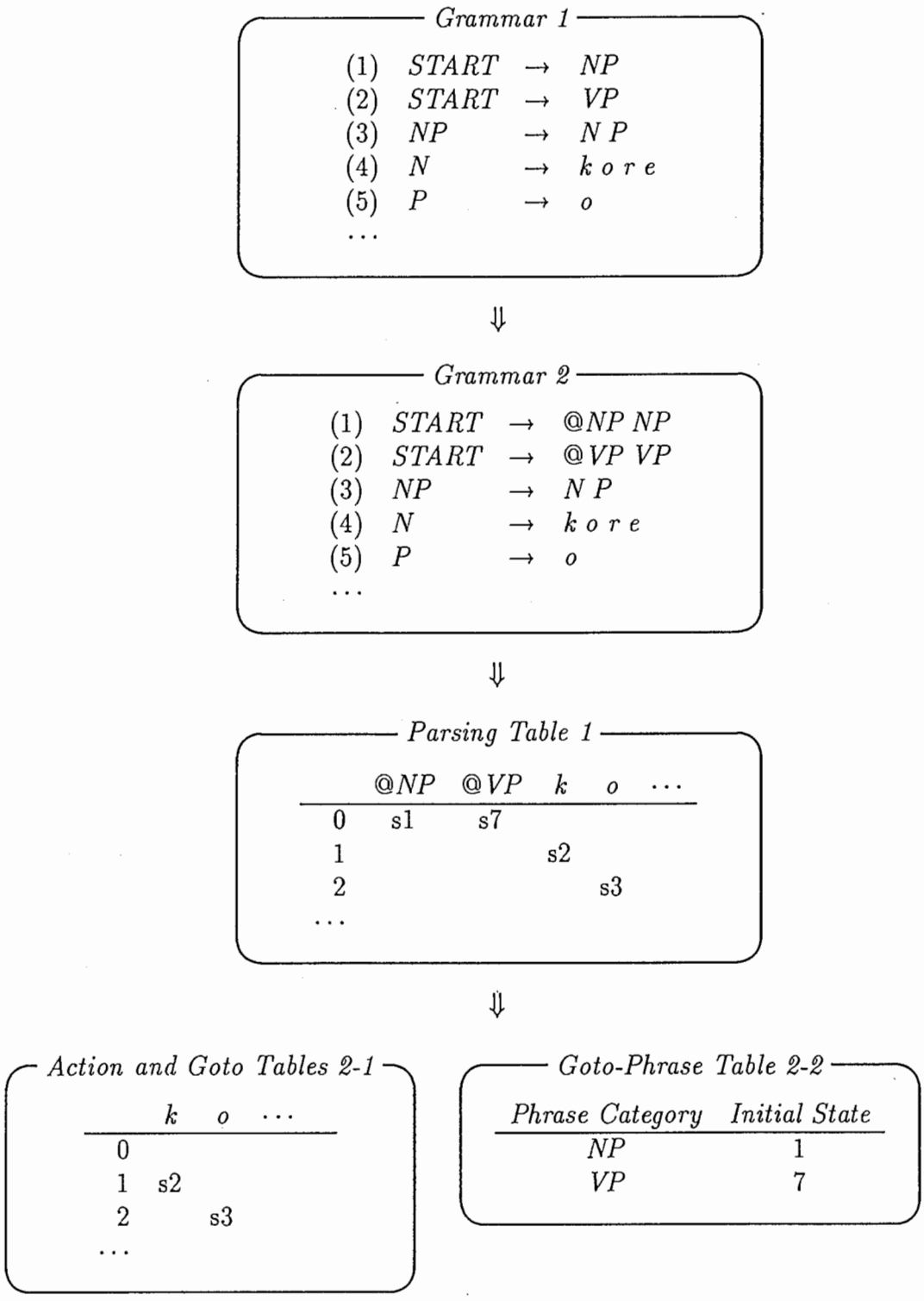


Figure 5.6: Construction of an intra-phrase LR parsing table.

5.4 Sentence Recognition Experiments

To confirm the effectiveness of two-level LR parsing, we compared two-level LR parsing with phrase lattice parsing. In the phrase lattice parsing experiment, the HMM-LR speech recognition system based on the intra-phrase grammar was used to recognize phrase-wise utterances and output the top five candidates for each phrase. Then the LR parser based on the inter-phrase grammar was applied to obtain sentence candidates from the phrase lattice.

5.4.1 Grammars

Two kinds of context-free grammars, the intra- and the inter-phrase grammars, were developed. [3] These grammars describe the conference registration task. The phonetic lexicon for the task is embedded in the intra-phrase grammar. The size and complexity of the grammars are shown in Table 5.2.

The general grammar mentioned in Chapter 3 is also a grammar for Japanese phrase structure. From the viewpoint of speech recognition, however, the general grammar is too loose because it occasionally generates incorrect sentences. This is because the general grammar was originally developed for written language processing. Many researchers point out that a grammar for speech recognition should reject illegal sentences. The intra-phrase grammar is a more realistic grammar for speech recognition,

Table 5.2: Size and complexity of the intra- and inter-phrase grammars.

Grammar	*Intra-Phrase*	*Inter-Phrase*
Rule	1,973 rules	471 rules
Vocabulary	744 words	133 phrase categories
Perplexity	3.57 / phone	65.5 / phrase

[3] A detailed explanation of these grammars is given in [Hosaka 91].

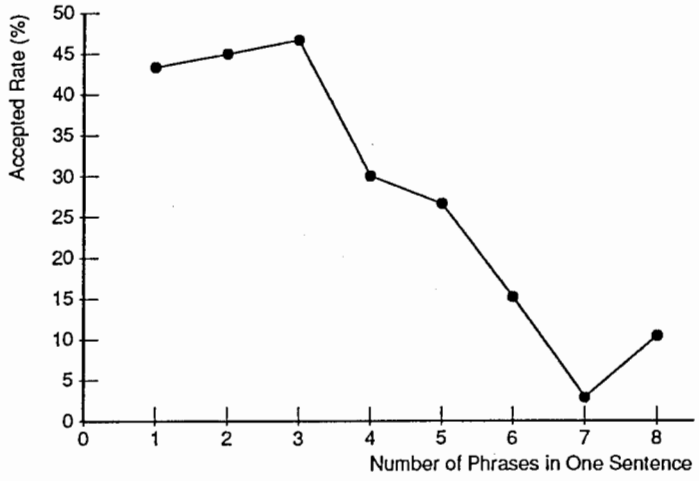


Figure 5.7: Strength of constraints imposed by the inter-phrase grammar.

We measured the strength of constraints imposed by the inter-phrase grammar by calculating the *accepted rate* [Takezawa 91], which is defined as follows:

$$accepted\ rate = \frac{accepted}{total} \times 100 \tag{5.1}$$

Here *total* indicates the number of sentences where the the phrase candidates are simply combined, and *accepted* indicates the number of accepted sentence candidates according to the inter-phrase grammar. Figure 5.7 depicts the relationship between the number of phrases in one sentence and the accepted rate. The inter-phrase grammar can rule out, on the average, 72.5% of sentence candidates. This result suggests that the inter-phrase grammar provides effective constraints for continuous speech recognition.

5.4.2 Speech Data

137 Japanese sentence utterances (353 phrases) refered to as *Model Conversation* were used for recognition experiments. These sentences were uttered by one male speaker (MAU) phrase by phrase. The average number of phrases in one sentence is 2.58. In our task, there are many sentences composed of one word, e.g. "hai" (yes), "iie" (no) and "moshimoshi" (hello). Accordingly, the average is somewhat low.

5.4.3 Results

Results were evaluated by *accuracy* [Lee 89], which can be calculated as follows.

$$accuracy = \frac{total - sub - ins - del}{total} \times 100 \tag{5.2}$$

where *total* indicates the number of words in all the sentences, and *sub*, *ins* and *del* are the number of words recognized as incorrect, deleted or inserted, respectively. Results of recognition experiments are shown in Table 5.3. By using two-level LR parsing, word accuracy and sentence accuracy are improved from 87.8% and 78.8% to 95.9% and 84.7%, respectively.

In the phrase lattice parsing experiment, correct phrases were sometimes missed within the top five choices. However, using inter-phrase grammatical constraints during recognition, the number of missed phrases decreased. This is why word accuracy and sentence accuracy are improved by using two-level LR parsing.

In phrase lattice parsing, if we examine more phrase candidates, it is expected that the recognition accuracy will increase. Assuming that a sentence is composed of five phrases and each phrase has 10 possible candidates, then 10^5 combinations must be considered. As this example demonstrates, taking more phrase candidates results in a computational explosion in some cases. From a computational viewpoint, we must conclude that taking more phrase candidates is an impractical approach.

Table 5.3: Word and sentence accuracy.

	Phrase Lattice Parsing		*Two-Level LR Parsing*	
Rank	*Word Accuracy*	*Sent. Accuracy*	*Word Accuracy*	*Sent. Accuracy*
1	87.8	78.8	95.9	84.7
~ 2	-	85.4	-	90.5
~ 3	-	86.9	-	93.4
~ 4	-	86.9	-	94.9
~ 5	-	86.9	-	94.9

5.5 Improvement of Two-Level LR Parsing

As discussed in Section 5.4, two-level LR parsing was successfully applied to sentence recognition. However, two-level LR parsing does not run efficiently for some grammars. In this section, we first describe a problem in two-level LR parsing, and then we propose an improved algorithm.

5.5.1 Problem in Two-Level LR Parsing

In two-level LR parsing, the system dynamically decides the initial state of the HMM-LR phrase recognizer. If multiple phrase categories are predicted, the HMM-LR will have multiple initial states. In some cases, this causes a problem. Consider the grammar in Figure 5.8 and its LR parsing table in Figure 5.9. If three phrase categories $S1$, $S2$ and $S3$ are predicted, the intra-phrase LR parser has initial states 1, 2 and 3. Since these states expect same phone /k/s, the HMM-LR phrase recognizer must verify phone /k/s three times. This sacrifices the performance of the system.

5.5.2 LR Parsing with a Category Reachability Test

We introduce a new parsing algorithm *LR parsing with a Category Reachability Test* (LR-CRT for short) [Kita 91g]. This algorithm enhances the flexibility of the generalized LR parsing in that it allows multiple start symbols in a grammar. The parser can recognize those sentences that belong to a specified start symbol. In this algorithm, an LR parser uses a standard LR parsing table except that action entries are augmented, i.e., there is one initial state in an LR parsing table. By adopting the LR-CRT algorithm in the intra-phrase LR parser, we can avoid the problem of redundant phone verification.

The key idea of the LR-CRT algorithm is to use an augmented LR parsing table whose action entries contain an additional field which is a set of reachable start symbols. That is, each action entry has the form $< A, S >$ where A is a parser action and S is a set of start symbols. For a given start symbol P, if P is a member of S, the action A is called *legal*. A partial parse obtained after executing a legal action can reach a given start symbol. The parsing algorithm is very similar to the standard algorithm except for an additional step to check the action legality when executing a shift or reduce action.

For an intra-phrase grammar, phrase categories can be considered as start symbols. Therefore the grammar in Figure 5.8 has three start symbols $S1$, $S2$ and $S3$, and its LR parsing table shown in Figure 5.10 contains subsets of $\{S1, S2, S3\}$ at each action entry.

(1)	*PH*	→	*S1*
(2)	*PH*	→	*S2*
(3)	*PH*	→	*S3*
(4)	*S1*	→	*k a*
(5)	*S2*	→	*k i*
(6)	*S3*	→	*k a i*

Figure 5.8: Problematic intra-phrase grammar.

	a	*i*	*k*	$	*PH*	*S*1	*S*2	*S*3
0					4			
1			*s*5			6		
2			*s*7				8	
3			*s*9					10
4				*acc*				
5	*s*11							
6				*r*1				
7		*s*12						
8				*r*2				
9	*s*13							
10				*r*3				
11				*r*4				
12				*r*5				
13		*s*14						
14				*r*6				

goto-phrase table

Phrase Category	*Initial State*
*S*1	1
*S*2	2
*S*3	3

Figure 5.9: Problematic intra-phrase LR parsing table.

	a	*i*	*k*	$	*PH*	*S1*	*S2*	*S3*
0			$< s3, \{S1, S2, S3\} >$		5	1	2	3
1				$< r1, \{S1\} >$				
2				$< r2, \{S2\} >$				
3	$< s7, \{S1, S3\} >$	$< s6, \{S2\} >$						
4				$< r3, \{S3\} >$				
5				*acc*				
6				$< r5, \{S2\} >$				
7		$< s8, \{S3\} >$		$< r4, \{S1\} >$				
8				$< r6, \{S3\} >$				

Figure 5.10: LR parsing table enhanced with reachable categories.

5.5.3 Construction of an Augmented LR Parsing Table

In order to construct an augmented LR parsing table, items must be redefined to include a set of categories as a second component. Thus, an item has the form $< A \rightarrow \alpha \cdot \beta, \{X_i\}_{i=1,\ldots,n} >$ where X_i is a start symbol.

Closure Function

Closure(I) is constructed from an item set I by the following rules.

1. For an item $< S \rightarrow \cdot A, \{\} >$ in I where S is a start symbol and A is a phrase category, add $< S \rightarrow \cdot A, \{A\} >$ to *Closure*(I). Other items in I are simply added to *Closure*(I).

2. Suppose $< A \rightarrow \alpha \cdot B\beta, \{X_i\} >$ is in *Closure*(I) and $B \rightarrow \gamma$ is a rewriting rule. If there is no item in *Closure*(I) of the form $< B \rightarrow \cdot\gamma, \ldots >$, add the item $< B \rightarrow \cdot\gamma, \{X_i\} >$ to *Closure*(I). Otherwise, for every item currently in *Closure*(I) of the form $< C \rightarrow \cdot\gamma, \{Y_j\} >$ such that $B \stackrel{lm}{\Longrightarrow} C\delta$ (leftmost derivation), replace that item with $< C \rightarrow \cdot\gamma, \{X_i\} \cup \{Y_j\} >$.

Goto Function

If I is an item set and X is a grammar symbol, then *Goto*(I, X) is defined to be the closure of the item set $< A \rightarrow \alpha X \cdot \beta, \{X_i\} >$ such that $< A \rightarrow \alpha \cdot X\beta, \{X_i\} >$ is in I.

Canonical Collection Construction

The canonical collection C of item sets is constructed as follows.

1. Initially $C := Closure(< S' \rightarrow \cdot S, \{\} >)$ (S' is an auxiliary start symbol).
2. Add $Goto(I, X)$ to C for each item set I in C and each grammar symbol X until no further item sets are created.

LR Parsing Table Construction

An augmented LR parsing table can be constructed from the canonical collection $C = \{I_0, I_1, \ldots, I_n\}$ as follows.

1. If $< A \rightarrow \alpha \cdot a\beta, \{X_k\} >$ is in I_i and $Goto(I_i, a) = I_j$, then set $Action[i, a]$ to "*shift* $j, \{Y_m\}$" where $\{Y_m\}$ is the union of $\{X_k\}$ in I_i with a after the dot. Here a must be a terminal.
2. If $< A \rightarrow \alpha\cdot, \{X_k\} >$ is in I_i, then set $Action[i, a]$ to "*reduce* $A \rightarrow \alpha, \{X_k\}$" for all a in $Follow(A)$.
3. If $< S' \rightarrow S\cdot, \{\} >$ is in I_i, then set $Action[i, \$]$ to "*accept*".
4. If $Goto(I_i, A) = I_j$, then $Goto[i, A] = j$.

5.5.4 Improved Two-Level LR Parsing

We will describe an improved two-level LR parsing algorithm for sentence speech recognition. The fundamental difference from the original algorithm is that the improved one uses the LR-CRT algorithm in the intra-phrase LR parser.

The LR-CRT algorithm is guided by an augmented LR parsing table, in which each action entry contains a set of reachable phrase categories. Thus, the action table has the form $Action[s, a] =< A, S >$, where s is a state, a is a phone name, A is an action (*shift*, *reduce*, *accept*, or *error*), and S is a set of phrase categories.

Supposing that a phrase category p is predicted, a set of phones predicted at state s is given as follows:

$$Prediction(s) = \{x | Action[s, x] =< \text{"}shift\text{"}, S > \ \& \ p \in S\} \tag{5.3}$$

Generally, the inter-phrase LR parser predicts multiple phrase categories. Provided that P is a set of predicted phrase categories, Equation 5.3 can be generalized as follows:

$$Prediction(s) = \{x | Action[s, x] =< \text{"}shift\text{"}, S > \ \& \ P \cap S \neq \phi\} \tag{5.4}$$

Figure 5.11 illustrates the phone prediction, in which an arrow indicates a phrase category reachability.

A sentence recognition algorithm is summarized below:

Definition of Symbols

$ST^{(inter)}$	...	A set of inter-phrase LR states. This set is initialized to $\{0\}$ when recognition is started, where 0 indicates the initial state of the inter-phrase LR parser.
$Action^{(inter)}[s, a]$	...	The action table of the inter-phrase LR parsing table.
$P^{(inter)}$	...	A set of phrase categories. This set contains the next categories predicted by the inter-phrase LR parser.

$ST^{(intra)}$, $Action^{(intra)}[s, a]$, and $P^{(intra)}$ are used in the intra-phrase LR parser and can be defined in the same way.

Algorithm

1. Initialize $ST^{(inter)}$.
2. If the end of a sentence is reached, then stop.
3. Predict the next phrase categories as follows:

$$\begin{aligned} P^{(inter)} = \{x | & s \in ST^{(inter)} \\ & \& \ Action^{(inter)}[s, x] = \text{“}shift\text{”}\} \end{aligned} \tag{5.5}$$

4. Call the HMM-LR phrase recognizer.
   (a) Initialize $ST^{(intra)}$.
   (b) If the end of a phrase is reached, then break.
   (c) Predict the next phones as follows:

$$\begin{aligned} P^{(intra)} = \{x | & s \in ST^{(intra)} \\ & \& \ Action^{(intra)}[s, x] = < \text{“}shift\text{”}, S > \\ & \& \ S \cap P^{(inter)} \neq \phi\} \end{aligned} \tag{5.6}$$

   (d) Verify each phone in $P^{(intra)}$ using HMMs.
   (e) Update $ST^{(intra)}$ based on the LR algorithm.
   (f) Return to 4b.
5. Update $ST^{(inter)}$ based on the LR algorithm.
6. Return to 2.

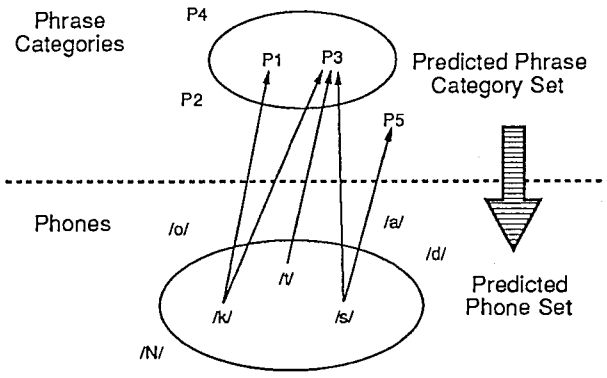


Figure 5.11: Phone prediction based on the category reachability.

5.5.5 Evaluation

The improved system using the LR-CRT algorithm was evaluated on the same speech data described in Section 5.4. The recognition result is shown in Table 5.4. The system attained a word accuracy of 97.5% and a sentence accuracy of 91.2%. These rates are particularly high, which suggests the effectiveness of the improved algorithm. [4]

Table 5.4: Word and sentence accuracy of the improved system.

Rank	*Word Accuracy*	*Sent. Accuracy*
1	97.5	91.2
~ 2	-	93.4
~ 3	-	96.4
~ 4	-	97.1
~ 5	-	98.5

[4]Appendix B contains the sentence recognition results.

5.6 Conclusion

This chapter described Japanese sentence speech recognition using two-level LR parsing. Two-level LR parsing is an extension of predictive LR parsing, in which two-level symbol prediction (phrase category prediction and phone prediction) is used to reduce the search space of a speech recognition system. It provides a mechanism to use both intra- and inter-phrase grammatical constraints during speech recognition.

Moreover, two-level LR parsing can be thought of as phrase-spotting-based recognition. Phrases are stable and not overly influenced by phenomena such as filled pauses, interjections, etc. Thus, our approach is robust to spontaneous speech recognition, which is problematic for current speech recognition systems.

We compared two-level LR parsing with phrase lattice parsing by sentence recognition experiments. Two-level LR parsing attained a word accuracy of 95.9% and a sentence accuracy of 84.7%, while phrase lattice parsing produced rates of 87.8% and 78.8%, respectively. These results show that two-level LR parsing is effective in sentence speech recognition.

In Japanese, phrase order is said to be much less constrained than in English. This leads many researchers to use the semantic dependency relationship between phrases. However, our experiments showed that syntactic knowledge also provides effective inter-phrase constraints.

We also introduced a new parsing algorithm, called the LR-CRT algorithm, and improved the sentence speech recognition system. The LR-CRT algorithm resolves the problem of maintaing multiple initial states, which reduced the recognition performance of the original system. The system finally attained a word accuracy of 97.5% and a sentence accuracy of 91.2%.

Chapter 6

Unknown Word Processing in Speech Recognition

6.1 Introduction

For natural language applications, processing unknown words is one of the most critical problems. In the area of written language processing, some methods for handling unknown words have been proposed (for example, see [Tomita 86]). Unfortunately, in the area of continuous speech recognition, there has been little progress in unknown word processing. Unlike written language processing, in continuous speech recognition, word boundaries are not clear and the correct input is not known. Thus, the problem is more difficult. Recently, Asadi et al. [Asadi 90] proposed a method for automatically detecting new words in a speech input. In their method, an explicit model of new words is used to recognize the existence of new words. However, little attention is paid to transcribing new words.

This chapter proposes a continuous speech recognition method that accepts any utterance, even though it might include unknown words [Kita 91a, Kita 91f]. Two ideas are introduced in the HMM-LR speech recognition system for processing unknown words. The first idea is to merge a task grammar and a phonetic grammar. The phonetic grammar describes the general Japanese phonetic structure [Kawabata 90], and is used to produce a phonetic transcription for an unknown word. The second idea is to impose a penalty on a recognition candidate that includes an unknown word in order to prevent a vocabulary word from being recognized as an unknown word. We evaluate the proposed method by phrase recognition experiments, in which proper nouns are treated as unknown words.

6.2 Phonetic Typewriter

6.2.1 Phonetic Typewriter Approach

A phonetic typewriter aims to transcribe arbitrary speech from an unlimited vocabulary. The idea itself is rather old; however, it has been a mere dream until recently because accurate and detailed acoustic models were not available. Recently, stimulated by remarkable progress in acoustic modeling research, some attempts have emerged toward developing the phonetic typewriter [Kohonen 88, Kawabata 90, Komori 91].

A phonetic typewriter generally does not assume any lexical information. It relys greatly on acoustic-phonetic information. However, human speech includes many factors, such as coarticulation effects and noises, that interfere with the acoustic analysis determining the unique phonetic identity of a given sound. It is very helpful to use the underlying syntactic and stochastic characteristics of a language. In Kohonen's study [Kohonen 88], context-dependent inference rules were used to correct coarticulation errors in speech recognition. In Kawabata's study [Kawabata 90], a general Japanese phonetic structure and a trigram model of Japanese syllables were used.

It would seem that the unknown word problem could be resolved by using a phonetic typewriter, because a phonetic typewriter is free from vocabulary limitations. However, in the phonetic typewriter approach, output phone sequences must be again analyzed by a linguistic processor to obtain word sequences, or to get syntactic/semantic structures. The disadvantage of this approach is that there is little interaction between the acoustic processor and the linguistic processor.

6.2.2 HMM-LR Based Phonetic Typewriter

The basic idea for unknown word processing is to introduce a phonetic typewriter approach into a task-specific HMM-LR speech recognition system. This subsection reviews a Japanese phonetic typewriter based on the HMM-LR speech recognition system [Kawabata 90].

The HMM-LR system is a syntax-directed system, and normally uses a task-specific context-free grammar that includes lexical information. However, by adopting a grammar that describes the general Japanese phonetic structure, the HMM-LR system becomes a phonetic typewriter. The grammar for Japanese phonetic structure includes constraints like "a sequence of consonants doesn't occur" or "the syllabic nasal /N/ doesn't occur at the head of a word". However, because the perplexity of this grammar is quite large, the trigram model of Japanese syllables is used at the same time (see Chapter 4). Kawabata et al. [Kawabata 90] reported that the syllable trigram model reduced the phone perplexity from 18.3 to 4.0. Figure 6.1 shows the grammar for Japanese phonetic structure.

(1)	*S*	→	*SEQ*
(2)	*SEQ*	→	*SYLLABLE_F*
(3)	*SEQ*	→	*SYLLABLE_F N*
(4)	*SEQ*	→	*SEQ SYLLABLE*
(5)	*SYLLABLE*	→	*SYLLABLE_F*
(6)	*SYLLABLE*	→	*SYLLABLE_F N*
(7)	*SYLLABLE*	→	*SYLLABLE_N*
(8)	*SYLLABLE*	→	*SYLLABLE_N N*
(9)	*SYLLABLE_F*	→	*w a*
(10)	*SYLLABLE_F*	→	*VOWEL*
(11)	*SYLLABLE_F*	→	*b VOWEL*
(12)	*SYLLABLE_F*	→	*g VOWEL*
(13)	*SYLLABLE_F*	→	*m VOWEL*
(14)	*SYLLABLE_F*	→	*n VOWEL*
(15)	*SYLLABLE_F*	→	*r VOWEL*
(16)	*SYLLABLE_F*	→	*z VOWEL*
(17)	*SYLLABLE_F*	→	*d A_E_O*
(18)	*SYLLABLE_F*	→	*t A_E_O*
(19)	*SYLLABLE_F*	→	*s A_E_O*
(20)	*SYLLABLE_F*	→	*h A_E_O*
(21)	*SYLLABLE_F*	→	*p A_E_O*
(22)	*SYLLABLE_F*	→	*k A_E_O*
(23)	*SYLLABLE_F*	→	*y A_U_O*
(24)	*SYLLABLE_F*	→	*by A_U_O*
(25)	*SYLLABLE_F*	→	*gy A_U_O*
(26)	*SYLLABLE_F*	→	*hy A_U_O*
(27)	*SYLLABLE_F*	→	*my A_U_O*
(28)	*SYLLABLE_F*	→	*ny A_U_O*
(29)	*SYLLABLE_F*	→	*ry A_U_O*
(30)	*SYLLABLE_F*	→	*zy A_U_O*
(31)	*SYLLABLE_F*	→	*sy A_U_O*
(32)	*SYLLABLE_F*	→	*py A_U_O*
(33)	*SYLLABLE_F*	→	*cy A_U_O*
(34)	*SYLLABLE_F*	→	*ky A_U_O*
(35)	*SYLLABLE_F*	→	*sh Ui*
(36)	*SYLLABLE_F*	→	*h Ui*
(37)	*SYLLABLE_F*	→	*ch Ui*
(38)	*SYLLABLE_F*	→	*p Ui*
(39)	*SYLLABLE_F*	→	*k Ui*
(40)	*SYLLABLE_F*	→	*s Uu*
(41)	*SYLLABLE_F*	→	*h Uu*
(42)	*SYLLABLE_F*	→	*ts Uu*
(43)	*SYLLABLE_F*	→	*p Uu*
(44)	*SYLLABLE_F*	→	*k Uu*
(45)	*SYLLABLE_F*	→	*sh ii*
(46)	*SYLLABLE_F*	→	*h ii*
(47)	*SYLLABLE_F*	→	*ch ii*
(48)	*SYLLABLE_F*	→	*p ii*
(49)	*SYLLABLE_F*	→	*k ii*
(50)	*SYLLABLE_F*	→	*s uu*
(51)	*SYLLABLE_F*	→	*h uu*
(52)	*SYLLABLE_F*	→	*ts uu*
(53)	*SYLLABLE_F*	→	*p uu*
(54)	*SYLLABLE_F*	→	*k uu*
(55)	*SYLLABLE_N*	→	*Q t A_E_O*
(56)	*SYLLABLE_N*	→	*Q s A_E_O*
(57)	*SYLLABLE_N*	→	*Q p A_E_O*
(58)	*SYLLABLE_N*	→	*Q k A_E_O*
(59)	*SYLLABLE_N*	→	*Q sy A_U_O*
(60)	*SYLLABLE_N*	→	*Q py A_U_O*
(61)	*SYLLABLE_N*	→	*Q cy A_U_O*
(62)	*SYLLABLE_N*	→	*Q ky A_U_O*
(63)	*SYLLABLE_N*	→	*Q sh Ui*
(64)	*SYLLABLE_N*	→	*Q ch Ui*
(65)	*SYLLABLE_N*	→	*Q p Ui*
(66)	*SYLLABLE_N*	→	*Q k Ui*
(67)	*SYLLABLE_N*	→	*Q s Uu*
(68)	*SYLLABLE_N*	→	*Q ts Uu*
(69)	*SYLLABLE_N*	→	*Q p Uu*
(70)	*SYLLABLE_N*	→	*Q k Uu*
(71)	*SYLLABLE_N*	→	*Q sh ii*
(72)	*SYLLABLE_N*	→	*Q ch ii*
(73)	*SYLLABLE_N*	→	*Q p ii*
(74)	*SYLLABLE_N*	→	*Q k ii*
(75)	*SYLLABLE_N*	→	*Q s uu*
(76)	*SYLLABLE_N*	→	*Q ts uu*
(77)	*SYLLABLE_N*	→	*Q p uu*
(78)	*SYLLABLE_N*	→	*Q k uu*
(79)	*VOWEL*	→	*i*
(80)	*VOWEL*	→	*ii*
(81)	*VOWEL*	→	*e*
(82)	*VOWEL*	→	*ee*
(83)	*VOWEL*	→	*ei*
(84)	*VOWEL*	→	*A_U_O*
(85)	*A_U_O*	→	*a*
(86)	*A_U_O*	→	*aa*
(87)	*A_U_O*	→	*u*
(88)	*A_U_O*	→	*uu*
(89)	*A_U_O*	→	*o*
(90)	*A_U_O*	→	*oo*
(91)	*A_U_O*	→	*ou*
(92)	*A_E_O*	→	*a*
(93)	*A_E_O*	→	*aa*
(94)	*A_E_O*	→	*e*
(95)	*A_E_O*	→	*ee*
(96)	*A_E_O*	→	*ei*
(97)	*A_E_O*	→	*o*
(98)	*A_E_O*	→	*oo*
(99)	*A_E_O*	→	*ou*

Figure 6.1: Grammar for Japanese phonetic structure.

The recognition process is summarized below. The predictive LR parser generates legitimate Japanese phone sequences based on the above-mentioned grammar, and these phone sequences are matched with the input speech using HMMs. The parser then calculates phone sequence likelihoods based on HMM and syllable trigram, and finally recognizes the speech as the phone sequence with the highest likelihood. Phone sequences with low likelihood scores are pruned during recognition by the beam-search technique.

The likelihood $P^{(TOTAL)}$ of a phone sequence is calculated as follows:

$$P^{(TOTAL)} = (1-\lambda) \times P^{(HMM)} + \lambda \times P^{(SYLLABLE)} \tag{6.1}$$

Here, λ is a scaling parameter, and $P^{(HMM)}$ and $P^{(SYLLABLE)}$ are log likelihoods based on HMM and syllable trigram, respectively.

Assuming here that the phone sequence is $s_1, \ldots, s_i, \ldots, s_n$, and $P_i^{(SYLLABLE)}$ is the sub-sequence likelihood up to the i-th phone, then $P^{(SYLLABLE)}$ is given by the following recurrence equation:

$$P_i^{(SYLLABLE)} =$$

$$\begin{cases} \dfrac{(i-1)P_{i-1}^{(SYLLABLE)} + P(s_i|\cdots, s_{i-2}, s_{i-1})}{i} & \text{if } s_i \text{ is a vowel} \\ \dfrac{(i-1)P_{i-1}^{(SYLLABLE)} + \sum_s P(s|\cdots, s_{i-1}, s_i)}{i} & \text{if } s_i \text{ is a consonant} \end{cases} \tag{6.2}$$

where $\sum$ is the sum over all syllables that contain the recognized consonant. Finally, $P^{(SYLLABLE)}$ is calculated as $P_n^{(SYLLABLE)}$.

6.3 Unknown Word Processing

6.3.1 Unknown Word Processing in the HMM-LR System

Figure 6.2 shows a schematic diagram of the unknown word processing. The system is based on the HMM-LR speech recognition system, and is enhanced with the trigram model of syllables to take into account the stochastic characteristics of a language.

For processing unknown words, the system needs two kinds of grammars. The first grammar is a normal grammar that describes our task. The phonetic spellings for each word are also included in this grammar. The second grammar is a grammar for Japanese phonetic structure, mentioned in the previous section. This grammar does not include phonetic spellings for each word, so it is suitable for transcribing an unknown word as a phone sequence. Hereafter, these two grammars are referred to as the *task grammar* and the *phonetic grammar*.

These two grammars are merged in the HMM-LR system. When merging two grammars, the start symbol of the phonetic grammar is replaced with pre-terminal names that might include unknown words (in our case, *proper-noun* is allowed to include unknown words). This enables the system to assign a part-of-speech to an unknown word. Figure 6.3 illustrates the merging of two grammars.

The fundamentals of unknown word processing are described below. If a speech input includes an unknown word, then a segment of the speech input does not match well with any word in the system's lexicon. In this case, the grammar for phonetic structure produces the phone sequence that matches well with the unknown word. If the speech input includes no unknown word, then the HMM-LR system outputs words in the lexicon.

The likelihood of recognition candidates is calculated from the HMM likelihood and the phone sequence likelihood based on the syllable trigrams. When recognition is completed, the likelihood of a recognition result that includes an unknown word is penalized a small value to reduce false alarms.

When the speech includes an unknown word, the system may output several candidates for the unknown word as phone sequences. Our main interest is, however, to get the best matching phone sequence for the unknown word. Therefore, it is quite desirable to ignore unknown word candidates except for the top one.

6.3.2 Estimation of the Penalty Factor

As we have stated, a penalty factor is added to the likelihood of a recognition result that includes an unknown word. This is effective in preventing a word in the lexicon from being recognized as an unknown word. The penalty factor is estimated from the test phrases $\{b_i | i = 1, \ldots, K\}$ by the following procedure:

1. Recognize test phrases twice, using the task and the phonetic grammars individually. Let R_i^{task} be a top recognition result for the i-th phrase b_i when using the task grammar, and P_i^{task} be a log likelihood of R_i^{task}. $R_i^{phonetic}$ and $P_i^{phonetic}$ can be defined in the same way.

2. For a small value x, create new results $\{R_i^{(x)} | i = 1, \ldots, K\}$ as follows:

$$R_i^{(x)} = \begin{cases} R_i^{task} & \text{if } P_i^{task} > P_i^{phonetic} + x \\ R_i^{phonetic} & \text{otherwise} \end{cases} \tag{6.3}$$

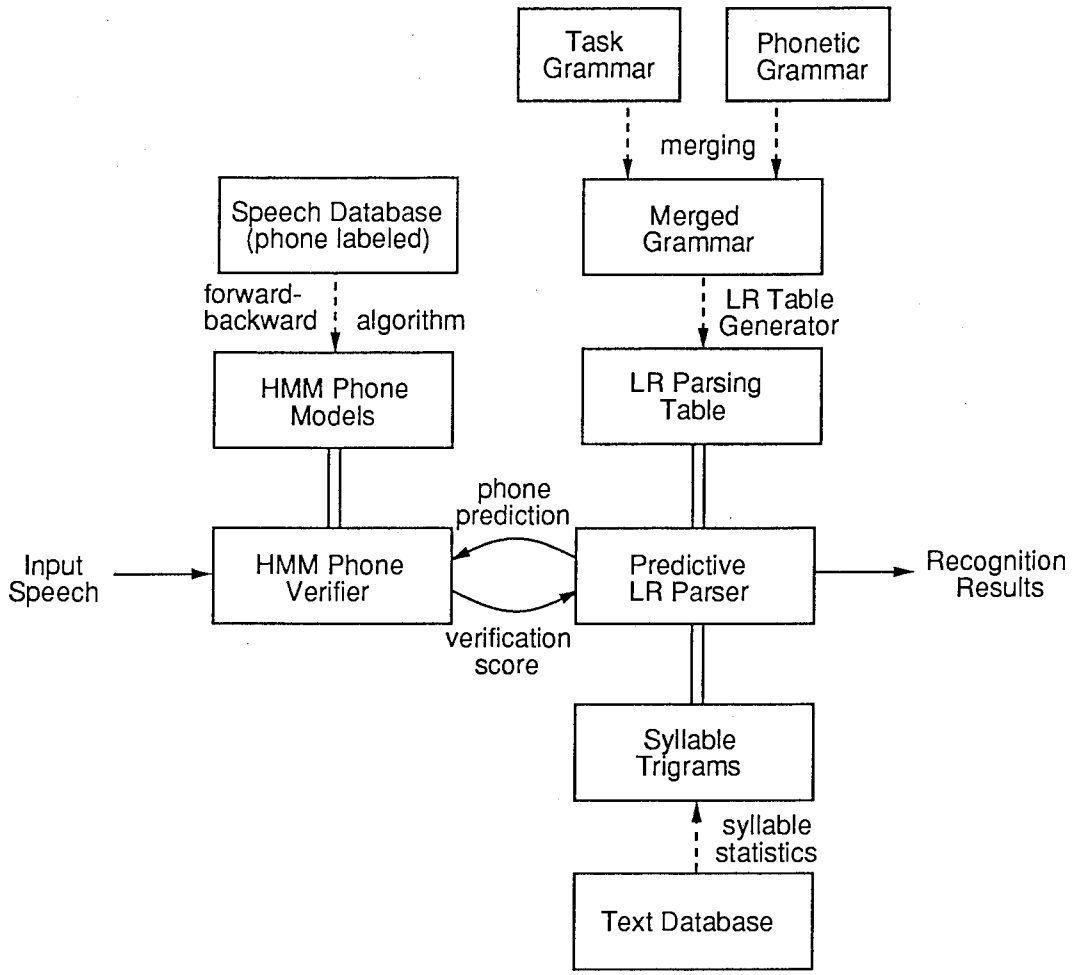


Figure 6.2: Schematic diagram of the unknown word processing.

A task grammar

S	$\rightarrow$	$NP\ VP$
NP	$\rightarrow$	N
NP	$\rightarrow$	$N\ P$
N	$\rightarrow$	$PROPER\_NOUN$
...		

A phonetic grammar

S	$\rightarrow$	SEQ
SEQ	$\rightarrow$	$SYLLABLE\_F$
...		

$\Downarrow$

A merged grammar

S	$\rightarrow$	$NP\ VP$
NP	$\rightarrow$	N
NP	$\rightarrow$	$N\ P$
N	$\rightarrow$	$PROPER\_NOUN$
...		
$PROPER\_NOUN$	$\rightarrow$	SEQ
SEQ	$\rightarrow$	$SYLLABLE\_F$
...		

Figure 6.3: Merging task and phonetic grammars.

3. Calculate $S(x)$ from $\{R_i^{(x)} | i = 1, \ldots, K\}$ and $\{b_i | i = 1, \ldots, K\}$ as follows:

$$S(x) = \left[\sum_{i=1}^{K} \Delta(R_i^{(x)}, b_i) \right] / K \tag{6.4}$$

$$\Delta(s,t) = \begin{cases} 1 & \text{if } s \text{ equals } t \\ 0 & \text{otherwise} \end{cases} \tag{6.5}$$

Here, $S(x)$ means a recognition rate for $\{R_i^{(x)} | i = 1, \ldots, K\}$.

4. The penalty factor *penalty* is determined to be a minimal x that maximizes Equation 6.4, i.e.,

$$penalty = \min_y \{y | y = \operatorname*{argmax}_x S(x)\} \tag{6.6}$$

We examined values $S(x)$ for various x, using the task grammar for the conference registration task. Table 6.4 plots a graph for the function $S(x)$. The graph indicates 0.25 as the best candidate for the penalty factor for unknown words.

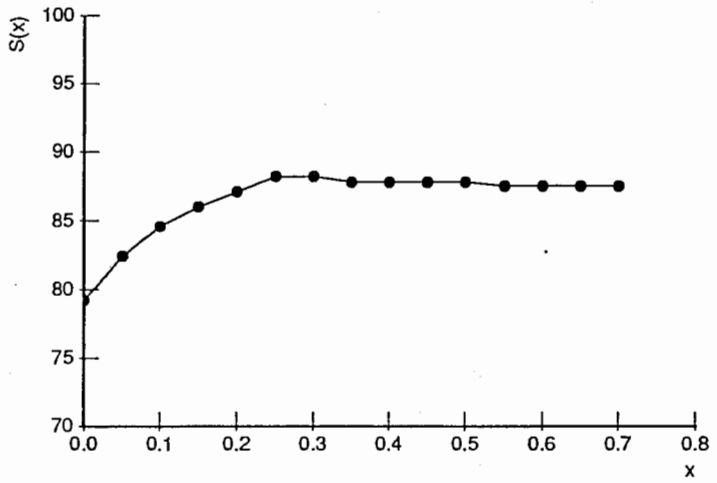


Figure 6.4: Graph of the function $S(x)$.

6.3.3 Relationship to the Phonetic Typewriter

The unknown word processing method introduced in this section is implemented in the HMM-LR speech recognition system, and shares some principles with the HMM-LR based phonetic typewriter. At this point, we have the following three similar systems:

- The pure HMM-LR speech recognition system.
- The HMM-LR based phonetic typewriter.
- The HMM-LR speech recognition system with unknown word processing capability.

Of course, these systems are strongly mutually related. The relationships among these systems are clarified in Figure 6.5.

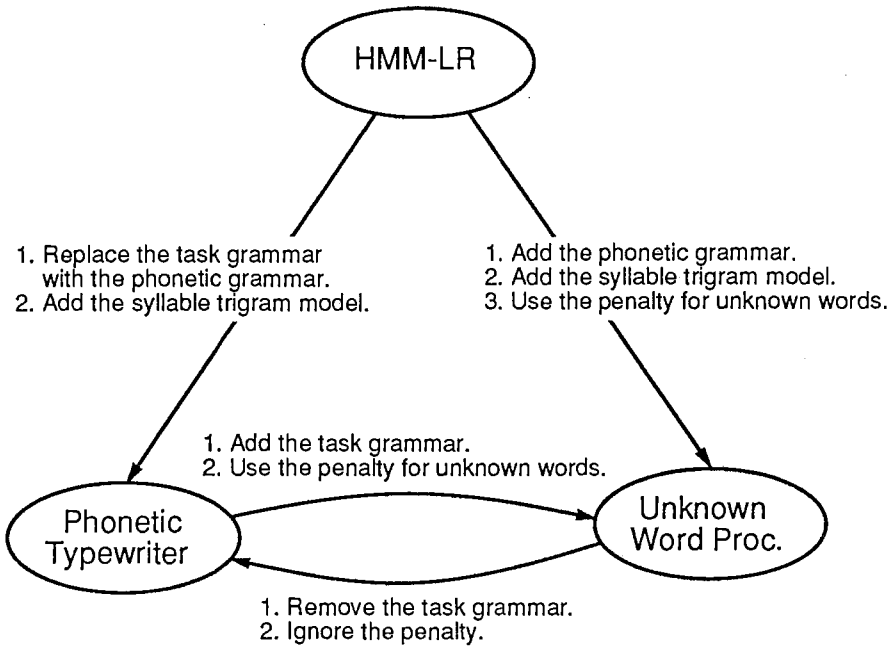


Figure 6.5: Relationship among HMM-LR based systems.

6.4 Recognition Experiments

The recognition experiments were carried out using 279 Japanese phrases uttered by one male speaker and the task grammar for the conference registration task, which were described in Chapter 3. To evaluate the unknown word processing method, all proper nouns (8 words), such as personal names and place names, were removed from the task grammar. The list of unknown words is shown in Table 6.1.

In the experiments, there were 14 phrases that inluded unknown words, and 11 phrases were transcribed correctly. Table 6.2 shows recognition results that include unknown words. By using the trigram model of Japanese syllables, the system can output very close phonetic transcriptions for unknown words. Table 6.3 shows the transcription rates for phrases that include unknown words. Here the transcription rate is equal to phone accuracy.

By using the phonetic grammar, the system can output a phonetic transcription for an unknown word. However, adding the phonetic grammar leads to increasing the size of the search space and hence degrades the recognition performance. Table 6.4 shows the phrase recognition rates for two cases: one where only the task grammar was used, and the other where the merged grammar, consisting of the task grammar and the phonetic grammar, was used. By adding the phonetic grammar, the phrase recognition rate dropped from 84.2% to 81.7%. In some recognition errors, the phonetic grammar causes a word in the lexicon to be recognized as unknown. This case represents 6.8% out of the error rate of 18.3%. Table 6.5 shows examples of recognition errors in this category.

Table 6.1: List of unknown words.

Unknown Words	*Occurrences in Test Data*
kyoto (*place name*)	4
kyotoeki (*place name*)	3
kitaoojieki (*place name*)	1
oosakashi (*place name*)	1
higashiku (*place name*)	1
shiromi (*place name*)	1
takara (*person's name*)	2
ichitarou (*person's name*)	1
Total	14

Table 6.2: Examples of recognition results that include unknown words.

Phrase No.	*Input*		*Result*	
	Correct	*Meaning*	*Without syllable trigrams*	*With syllable trigrams*
108	oosakashi	Osaka city	oosakashi	oosakashi
109	higashiku	Higashiku	shigashiku	higashiku
110	shiromi	Shiromi	shidoumi	shiryouni
120	takara	Takara	kakara	kaikara
121	ichitaroudesu	(I am) Ichitarou	ishitaoouutsusu	ishitaroudesu
127	takarasamadesune	(You're) Mr. Takara (aren't you)	takaasabautsunu	takarasamadesune
144	kyouto	Kyoto	hyoupo	kyouto
154	kyouto	Kyoto	kyouto	kyouto
158	kyoutoekikara	from Kyoto station	kuuootorechikaa	kyoutoekikara
164	kitaoojiekimade	to Kitaoji station	shitaouziekimare	kitaoojiekimade
174	kyoutoekikara	from Kyoto station	hyotorekitaahu	kyoutoekikara
175	kyouto	Kyoto	kyouto	kyouto
206	kyoutoekikara	from Kyoto station	hyootouriiishikaa	kyoutoekikara
207	kyouto	Kyoto	kyouto	kyouto

Table 6.3: Transcription rates for phrases that include unknown words.

Without syllable trigrams	*With syllable trigrams*
66.1%	95.3%

Table 6.4: Phrase recognition rates (with syllable trigrams).

Rank	*Phrase Recognition Rate (%)*		
	Task grammar	*Task grammar + Phonetic grammar*	
		Without a penalty	*With a penalty*
1	84.2	75.3	81.7
2	94.3	86.0	86.4
3	95.7	87.8	87.5
4	95.7	88.5	88.2
5	95.7	88.5	88.2

Table 6.5: Examples of recognition errors in which vocabulary words are recognized as unknown words.

Phrase No.	*Correct*	*Result*
10	syoteino	syoteno
51	syoteino	syoteno
68	baainiwa	baainiga
69	yokousyuudai	yokoNsyudai
80	yokousyuudaio	ryokoushinaiyouo
95	yoroshiinodesyouka	yoroshiiNdesyouka
103	syoteino	syoteno
113	rokuzyuuichi	bokuzyuuichi
152	touridesu	tooridesu
157	ikuniwa	ikiNga
172	riyou	gaiyou
223	deNwani	deNaini
224	kagirarerunodesyouka	kairareruNdesyouka
227	deNwadakede	deNwaidakede
232	rikai	gikai
236	hoNyakunadono	omoiyakunadono
264	hoNyakusaremasu	koNrakusaimasu
270	shiQpitsushite	shiQtatsushite
281	agemasu	aidasu

6.5 Conclusion

In this chapter, we developed a speech recognition method that can process unknown words by introducing a phonetic typewriter approach into a task-specific HMM-LR speech recognition system. The main features of the proposed method are:

- Using a grammar derived from merging a task grammar and a phonetic grammar — the task grammar is utilized for recognizing vocabulary words, while the phonetic grammar is utilized for transcribing out-of-vocabulary words;
- Imposing a penalty on a recognition result that includes an unknown word — the use of a penalty is effective in preventing a vocabulary word from being recognized as an unknown word.

If a speech input includes an unknown word, then a segment of the speech input does not match well with any word in the system's lexicon. In this case, the phonetic grammar produces a phone sequence that closely matches with the unknown word. If the speech input includes no unknown word, then the HMM-LR system uses the task grammar and outputs words in the lexicon. Moreover, in our approach it is possible to assign a part-of-speech to an unknown word.

Evaluation experiments revealed that the proposed method is promising for processing unknown words. Results also showed that the trigram model of Japanese syllables is very effective in getting phonetic transcriptions for unknown words.

Chapter 7

Conclusion

7.1 Summary

This thesis has been concerned mainly with language modeling for automatic speech recognition. We have proposed efficient speech recognition algorithms that take full advantage of the grammatical constraints supplied by syntactic language models. We have also discussed stochastic langauge models and successfully applied them to improve speech recognition accuracy.

In Chapter 2, we first proposed predictive LR parsing, which is an extension of generalized LR parsing. Predictive LR parsing makes it possible to predict next input symbols according to a context-free grammar. We then developed the HMM-LR speech recognition algorithm by combining Hidden Markov Models and predictive LR parsing. This algorithm allows the use of a general context-free grammar to guide the search of an HMM-based speech recognizer.

In Chapter 3, we described two implementations of the HMM-LR speech recognition system, and evaluated the system through Japanese phrase recognition experiments. The baseline version uses only LPC cepstral parameters in one codebook. It attained a phrase recognition rate of 72.0% using the context-free grammar including 1,035 words and having a phone perplexity of 5.9. We then improved the system by using multiple codebooks, accurate HMM state duration control, and fuzzy vector quantization. A speaker adaptation algorithm based on codebook mapping was also incorporated. The improved version attained an average phrase recognition rate of 89.5% and 99.2% for the top five choices in the speaker-dependent condition. In the speaker-adapted condition, rates were 81.6% and 98.0%, respectively. We also showed the feasibility of the HMM-LR in very large vocabulary speech recognition.

In Chapter 4, we investigated three stochastic language models: (1) a trigram model of Japanese syllables, (2) stochastic LR parsing, and (3) a trigram model of context-free rewriting rules. These models were incorporated into the HMM-LR speech recognition system, and tested by phrase recognition experiments for one male speaker. By utilizing each of these models, the phrase recognition rate was improved from 88.2% to 92.5%, 92.1% and 91.4%, respectively. By utilizing all three models, a rate of 93.2% was achieved.

In Chapter 5, we proposed a sentence speech recognition algorithm using two-level LR parsing. In Japanese, a sentence is composed of several phrases, thus using two kinds of grammars, namely an intra-phrase grammar and an inter-phrase grammar, is sufficient for recognizing sentences. Two-level LR parsing provides a mechanism to use these two grammars stepwise during speech recognition. The proposed algorithm was evaluated through Japanese sentence recognition experiments where sentences were uttered phrase by phrase. The system attained a word accuracy of 97.5% and a sentence accuracy of 91.2%.

In Chapter 6, we discussed unknown word processing in speech recognition. We proposed a speech recognition method that gives a phonetic transcription for an unknown word. This method was tested by using the HMM-LR speech recognition system. The system achieved a high transcription rate for unknown words.

7.2 Future Work

In this thesis, we have not considered the use of higher level linguistic knowledge in speech recognition, such as semantics, pragmatics or dialogue. A practical approach to using higher level knowledge is to attach additional constraints or operations to individual rewriting rules, like *Augmented Transition Networks* (ATN) [Woods 70] or *Definite Clause Grammars* (DCG) [Pereira 80]. Recent grammar formalism called *unification-based formalism* [Shieber 86] permits integrated descriptions of syntactic, semantic and pragmatic constraints in terms of feature structures. Some attempts have been made to use unification-based grammars in speech recognition [Hemphill 89, Chow 89]. It is not overly difficult to extend the HMM-LR speech recognition system to handle unification-based grammars. However, unification is computationally very expensive, and thus would degrade speech recognition performance. Alternatively, we can adopt the *N-best search paradigm* [Schwartz 90], in which the knowledge sources are applied progressively. That is, first the inexpensive knowledge source is used to obtain a list of the most likely recognition hypotheses, which are then filtered by the remaining knowledge sources. The HMM-LR speech recognition system can output the N best hypotheses, thus the N-best search paradigm is easily incorporated into the system.

Another problem is the use of context-dependent phone models (allophonic models). In continuous speech recognition, phone pattern variations are highly dependent on their environment including their phonetic contexts. Thus, it is advantageous to use context-dependent phone models to improve speech recognition accuracy [Schwartz 85, Lee 89]. Sagayama [Sagayama 89] proposed a *phoneme environment clustering algorithm* (PEC) which is a scheme for systematically analyzing the allophonic variations in speech. The HMM-LR algorithm proposed in this thesis has been supposed to drive context-independent phone models. We are currently developing an extended HMM-LR algorithm that can handle context-dependent phone models derived from the PEC algorithm. Preliminary experiments show promising results [Nagai 91].

Another problem concerns the robustness of left-to-right speech recognition strategy, which is adopted by the HMM-LR speech recognition system. As the name implies, in this strategy, the system scans the input from the beginning (left) to the end (right) pruning low-likelihood hypotheses. In the left-to-right strategy, a problem arises when one word is missing or its likelihood score is very low. In this case, the system no longer extends the correct path. The simplest way to remedy this weakness is to assume a dummy word or tag in the input. For example, Saito et al. [Saito 88b] proposed a method of parsing phoneme sequences that might include errors, in which missing phonemes are handled by inserting possible phonemes between two real phonemes. Saito [Saito 90] also proposed *gap-filling LR parsing* for noisy input speech. In gap-filling LR parsing, the system assumes fake nonterminals at the positions of the missing words. However, assuming dummy words or tags reduces the efficiency of the recognizer because it produces a large number of hypotheses. Another solution is indicated in [Kita 91d]. In this method, when there is a problem, the system switches the grammar to a looser one. This idea is similar to the *relaxation techniques* [Kwasny 81] in the area of natural language processing, in that it relaxes the constraints of the grammar. Another possibility is to use the A^* *search algorithm* [Nilsson 80], which finds the optimal solution. [1] The A^* search algorithm uses an evaluation function defined as follows:

$$f^*(x) = g(x) + h^*(x) \tag{7.1}$$

Here, $f^*(x)$ is the estimated cost of the best path through x, $g(x)$ is the cost of getting from the starting node to x, and $h^*(x)$ is the estimated cost of getting from x to the goal node. In speech recognition, $g(x)$ is given from the matching score from the beginning of the utterance to the current word. Most speech recognition systems use only $g(x)$ and ignore $h^*(x)$. If we can define an appropriate function $h^*(x)$, then we could enhance the robustness of left-to-right strategy.

[1] Soong et al. [Soong 90] or Zue et al. [Zue 90] used the A^* search to find the N-best sentence hypotheses.

Another interesting direction for future work is to combine the HMM-LR algorithm with a dialogue model. Yamaoka et al. [Yamaoka 90] proposed a plan-based dialogue model which has the capability to predict the next utterance. Robinson [Robinson 82] proposed a grammar for dialogue. In Chapter 5, we proposed an extended LR parsing algorithm, the LR-CRT algorithm, which allows multiple start symbols in a grammar. Combining the LR-CRT algorithm with these dialogue models makes it possible to exclusively recognize those utterances that are consistent with the current plan.

7.3 Final Remarks

This thesis introduced a number of new techniques concerning language modeling for automatic speech recognition. With these techniques, we developed the high performance speech recognition system, HMM-LR, which can be considered as one of the best Japanese speech recognition systems. The HMM-LR speech recognition system is used as the front-end of SL-TRANS [Morimoto 90], an experimental system for translating Japanese conversation into English.

ATR Interpreting Telephony Research Laboratories is currently undertaking extensive efforts to perfect technologies necessary for human-to-human communication unbounded by language barriers. It is our hope that our research will contribute to the realization of an automatic telephone interpretation system, and will help to point the direction for future research.

Appendix A

Phrase Recognition Results

A.1 The Grammar

We enumerate the grammar rules used in the phrase recognition experiments. This grammar is refered to as the *general grammar* in Chapter 3. Each rule has the form $(LHS$ <--> $(RHS_1, \ldots, RHS_n))$ where LHS indicates the left side symbol and $RHS_1, \ldots, RHS_n$ are the right side symbols. The grammar symbols quoted by '<' and '>' indicate nonterminal symbols. The start symbol is indicated by *<start>*.

```
 1: (<start> <--> (Q1 <s> Q2))
 2: (<s> <--> (<conj-s>))
 3: (<s> <--> (<vp>))
 4: (<s> <--> (<np> <cop>))
 5: (<s> <--> (<np>))
 6: (<s> <--> (<xp>))
 7: (<s> <--> (<adv>))
 8: (<np> <--> (<det> <nn>))
 9: (<np> <--> (<nn>))
10: (<nn> <--> (<n-no>))
11: (<nn> <--> (<whn>))
12: (<n-no> <--> (<n>))
13: (<xp> <--> (<np> <p>))
14: (<xp> <--> (<whn> <p>))
15: (<xp> <--> (<pron>))
16: (<vp> <--> (<v>))
17: (<v> <--> (<kdo> <cop>))
18: (<cop> <--> (<cop-dec>))
19: (<cop-dec> <--> (d e s Uu))
20: (<cop-dec> <--> (d e s Uu n e))
21: (<cop-dec> <--> (d e sy ou k a))
22: (<cop-dec> <--> (d e s Uu g a))
23: (<cop-dec> <--> (n a N d e s Uu))
24: (<cop-dec> <--> (<cop-past1>))
25: (<cop-dec> <--> (<cop-past2>))
26: (<cop-dec> <--> (<cop-nonpast>))
```

```
27: (<cop-dec> <--> (<cop-past1> d e s Uu))
28: (<cop-dec> <--> (<cop-past2> d e s Uu))
29: (<cop-dec> <--> (<cop-nonpast> d e s Uu))
30: (<cop-dec> <--> (<cop-n> d e s Uu))
31: (<cop-dec> <--> (d e sh Ui t a))
32: (<cop-dec> <--> (<cop-neg1> d e sh Ui t a))
33: (<cop-dec> <--> (<cop-neg1>))
34: (<cop-n> <--> (<cop-nonpast> N))
35: (<cop-n> <--> (<cop-past1> N))
36: (<cop-n> <--> (<cop-past2> N))
37: (<cop-past1> <--> (d a Q t a))
38: (<cop-nonpast> <--> (<cop-neg2> i))
39: (<cop-past2> <--> (<cop-neg2> k a Q t a))
40: (<cop-neg1> <--> (zy a a r i m a s e N))
41: (<cop-neg1> <--> (d e w a a r i m a s e N))
42: (<cop-neg2> <--> (zy a n a))
43: (<cop-neg2> <--> (d e w a n a))
44: (<cop> <--> (<cop-dec> k a))
45: (<cop-dec> <--> (d a))
46: (<nm> <--> (n o))
47: (<nm> <--> (k o t o))
48: (<comp> <--> (k a))
49: (<comp> <--> (t o))
50: (<comp> <--> (t o k Ui))
51: (<comp> <--> (m a e))
52: (<comp> <--> (a t o))
53: (<p> <--> (g a))
54: (<p> <--> (o))
55: (<p> <--> (n i))
56: (<p> <--> (w a))
57: (<p> <--> (n i w a))
58: (<p> <--> (n o))
59: (<p> <--> (k a r a))
60: (<p> <--> (m a d e))
61: (<p> <--> (m a e))
62: (<p> <--> (a t o))
63: (<p> <--> (t o))
64: (<p> <--> (t o i Q sy o))
65: (<p> <--> (t o i Q sy o n i))
66: (<p> <--> (d e))
67: (<p> <--> (e))
68: (<p> <--> (m o))
69: (<p> <--> (t a m e n i))
70: (<p> <--> (t a m e))
71: (<p> <--> (g o r o))
72: (<p> <--> (g u r a i))
73: (<p> <--> (g o r o n i))
74: (<p> <--> (g u r a i n i))
75: (<p> <--> (m a d e n i))
76: (<p> <--> (n i y o Q t e))
77: (<p> <--> (n i t a i s Uu r u))
78: (<p> <--> (k a))
79: (<p> <--> (b a k a r i))
80: (<p> <--> (d a k e))
81: (<p> <--> (sh Ui k a))
82: (<p> <--> (d a n o))
83: (<p> <--> (d e m o))
84: (<p> <--> (h o d o))
85: (<p> <--> (k Uu r a i))
86: (<p> <--> (k o s o))
87: (<p> <--> (n a d o))
88: (<p> <--> (n a g a r a))
89: (<p> <--> (n e))
90: (<p> <--> (s a))
91: (<p> <--> (s a e))
```

```
 92: (<p> <--> (y a))
 93: (<p> <--> (y a r a))
 94: (<p> <--> (y o))
 95: (<p> <--> (y o r i))
 96: (<p> <--> (n o m i n o))
 97: (<p> <--> (n a d o n o))
 98: (<p> <--> (e n o))
 99: (<p> <--> (m a d e n o))
100: (<s> <--> (k Uu r a i))
101: (<s> <--> (ts Uu i t e w a))
102: (<s> <--> (y ou n a))
103: (<s> <--> (<n> d a k e d e))
104: (<s> <--> (<n> d a k e d a))
105: (<s> <--> (n a k Uu))
106: (<s> <--> (m ou sh Ui))
107: (<s> <--> (k ou h a N n a))
108: (<s> <--> (<v-rentai-nonpast> n i w a))
109: (<v> <--> (<v-te> k Uu d a s a i))
110: (<v> <--> (o <v-5dan-i> k Uu d a s a i))
111: (<v> <--> (<v-te> k Uu d a s a i m a s e N k a))
112: (<v> <--> (<v-te> k Uu r e m a s e N k a))
113: (<v> <--> (<v-rentai-nonpast>))
114: (<v> <--> (<v-rentai-past>))
115: (<v> <--> (<kyo> i))
116: (<v> <--> (<kdo> n a))
117: (<v> <--> (<kdo>))
118: (<v> <--> (<v-neg1>))
119: (<v-neg1> <--> (<v-negpres>))
120: (<v-neg1> <--> (<v-negpast>))
121: (<v-negpast> <--> (<v-negpres> d e sh Ui t a))
122: (<v-negpres> <--> (<v-ger> a r i m a s e N))
123: (<v-negpres> <--> (<v-masu> s e N))
124: (<kyo> <--> (n a))
125: (<kyo> <--> (<v-mizen1> n a))
126: (<kyo> <--> (<v-ger> n a))
127: (<v-ger> <--> (<kyo> k Uu))
128: (<v-5dan-r> <--> (<v-ger> n a))
129: (<v-sahen> <--> (<v-ger>))
130: (<v-ta> <--> (<v-renyo2-t> t a))
131: (<v-ta> <--> (<v-renyo2-d> d a))
132: (<v-ta> <--> (<kyo> k a Q t a))
133: (<v-rentai-past> <--> (<v-ta>))
134: (<v> <--> (<v-rentai-nonpast> g a))
135: (<v> <--> (<v-rentai-nonpast> k a))
136: (<v> <--> (<v-rentai-nonpast> n o d e sy ou k a))
137: (<v> <--> (<v-rentai-nonpast> d e sy ou k a))
138: (<v> <--> (<v-rentai-past> k a))
139: (<v> <--> (<kyo> i d e sy ou k a))
140: (<v> <--> (<kyo> i N d e sy ou k a))
141: (<v> <--> (<v-n> d e s Uu k a))
142: (<v> <--> (<v-cop> d e s Uu k a))
143: (<v> <--> (<v-rentai-nonpast> d e s Uu k a))
144: (<v> <--> (<v-rentai-past> d e s Uu k a))
145: (<v> <--> (<v-neg1> k a))
146: (<v> <--> (<v-te>))
147: (<v> <--> (<v-ger> t e))
148: (<v> <--> (<v-rentai-nonpast> t o))
149: (<v> <--> (<kyo> t o))
150: (<v> <--> (<v-katei> b a))
151: (<v> <--> (<v-katei>))
152: (<v> <--> (<kyo> k e r e b a))
153: (<v> <--> (<kdo> n a r a))
154: (<v> <--> (<kdo> n a r a b a))
155: (<v> <--> (<kdo> <cop-dec>))
156: (<v> <--> (<v-ta> r a))
```

```
157: (<v-rentai-nonpast> <--> (<v-rentai>))
158: (<v-rentai-time> <--> (<v-rentai>))
159: (<v> <--> (<v-cop>))
160: (<v-cop> <--> (<kyo> i))
161: (<v-n> <--> (<v-cop> N))
162: (<v-n> <--> (<v-rentai-nonpast> N))
163: (<v-n> <--> (<v-rentai-past> N))
164: (<v-n> <--> (<kdo> n a N))
165: (<v> <--> (<v-n> d e s Uu))
166: (<v> <--> (<v-n> d e s Uu n e))
167: (<v> <--> (<v-cop> d e s Uu))
168: (<v> <--> (<v-n> d e sh Ui t a))
169: (<v> <--> (<v-cop> d e sh Ui t a))
170: (<v> <--> (<kyo> i n o d e s Uu))
171: (<v> <--> (<v-rentai> n o d e w a))
172: (<v> <--> (<v-rentai> d a k e d a t o))
173: (<v> <--> (<kyo> i N d e s Uu g a))
174: (<v> <--> (y o r o sh ii n o d e sy ou k a))
175: (<kyo> <--> (<v-mizen1> n a k e r e b a n a r a n a))
176: (<v-masu> <--> (<v-mizen1> n a k e r e b a n a r i))
177: (<v-1dan> <--> (<v-mizen3> s e))
178: (<v-1dan> <--> (<v-mizen4> r e))
179: (<kyo> <--> (<v-renyo1> t a))
180: (<v-1dan> <--> (<v-5dan-e>))
181: (<v-1dan> <--> (<v-1dan> r e))
182: (<v-1dan> <--> (<v-1dan> r a r e))
183: (<v-1dan> <--> (k o r e))
184: (<v-1dan> <--> (<v-sahen> d e k Ui))
185: (<v-1dan> <--> (<v-fsahen> s e))
186: (<v-1dan> <--> (<v-fzahen> z i r e))
187: (<v-1dan> <--> (<v-rentai> k o t o g a d e k Ui))
188: (<kyo> <--> (<v-rentai> r a sh Ui))
189: (<kyo> <--> (<v-ta> r a sh Ui))
190: (<kyo> <--> (<kyo> i r a sh Ui))
191: (<kyo> <--> (<kdo> r a sh Ui))
192: (<kdo> <--> (<v-rentai> y ou))
193: (<kdo> <--> (<v-ta> y ou))
194: (<kdo> <--> (<kyo> i y ou))
195: (<kdo> <--> (<kdo> n a y ou))
196: (<kdo> <--> (<v-renyo1> s ou))
197: (<kdo> <--> (<kyo> s ou))
198: (<kdo> <--> (<kdo> s ou))
199: (<kdo> <--> (<v-rentai> s ou))
200: (<kdo> <--> (<v-ta> s ou))
201: (<kdo> <--> (<kyo> i s ou))
202: (<kdo> <--> (<kdo> d a s ou))
203: (<v-masu> <--> (<v-renyo1> m a))
204: (<v-mizen2> <--> (<v-masu> sy o))
205: (<v-mizen5> <--> (<v-masu> s e))
206: (<v-renyo2-t> <--> (<v-masu> sh Ui))
207: (<v-rentai> <--> (<v-masu> s Uu))
208: (<v-katei> <--> (<v-masu> s Uu r e))
209: (<v-te> <--> (<v-renyo2-t> t e))
210: (<v-te> <--> (<v-renyo2-d> d e))
211: (<v-te> <--> (<v-mizen1> n a i d e))
212: (<v-cop> <--> (<v-te> m o i))
213: (<v-cop> <--> (<v-te> m o y o r o sh ii))
214: (<v-cop> <--> (<v-ger> t e m o i))
215: (<v-cop> <--> (<v-ger> t e m o y o r o sh ii))
216: (<v-cop> <--> (<v-te> w a i k e n a i))
217: (<v-cop> <--> (<v-te> w a i k e m a s e N))
218: (<v-cop> <--> (<v-ger> t e w a i k e n a i))
219: (<v-cop> <--> (<v-ger> t e w a i k e m a s e N))
220: (<v-1dan> <--> (<v-te> m i))
221: (<v-1dan> <--> (<v-te> i))
```

```
222: (<v> <--> (<v-te> i r u d e sy ou))
223: (<v> <--> (<v-te> i r u d e sy ou k a))
224: (<v-aru> <--> (<v-te> a))
225: (<v-5dan-kt> <--> (<v-te> i))
226: (<v-kahen> <--> (<v-te>))
227: (<v-5dan-kt> <--> (<v-te> o))
228: (<v-5dan-w> <--> (<v-te> sh Ui m a))
229: (<v-1dan> <--> (<v-te> a g e))
230: (<v-1dan> <--> (<v-te> s a sh Ui a g e))
231: (<v-5dan-ki> <--> (<v-te> i t a d a))
232: (<v-5dan-ki> <--> (o <v-5dan-i> t a d a))
233: (<v-5dan-ki> <--> (<v-mizen3> s e t e i t a d a))
234: (<v> <--> (<v-te> i t a d a k Ui t a i n o d e s Uu))
235: (<v> <--> (<v-te> i t a d a k Ui t a i n o d e s Uu g a))
236: (<v> <--> (i t a d a k Ui t a i n o d e s Uu g a))
237: (<v-5dan-a> <--> (<v-5dan-w> w a))
238: (<v-5dan-a> <--> (<v-5dan-kt> k a))
239: (<v-5dan-a> <--> (<v-5dan-ki> k a))
240: (<v-5dan-a> <--> (<v-5dan-s> s a))
241: (<v-5dan-a> <--> (<v-5dan-t> t a))
242: (<v-5dan-a> <--> (<v-5dan-n> n a))
243: (<v-5dan-a> <--> (<v-5dan-m> m a))
244: (<v-5dan-a> <--> (<v-5dan-r> r a))
245: (<v-5dan-a> <--> (<v-5dan-g> g a))
246: (<v-5dan-a> <--> (<v-5dan-b> b a))
247: (<v-5dan-i> <--> (<v-5dan-w> i))
248: (<v-5dan-i> <--> (<v-5dan-kt> k Ui))
249: (<v-5dan-i> <--> (<v-5dan-ki> k Ui))
250: (<v-5dan-i> <--> (<v-5dan-s> sh Ui))
251: (<v-5dan-i> <--> (<v-5dan-t> ch Ui))
252: (<v-5dan-i> <--> (<v-5dan-n> n i))
253: (<v-5dan-i> <--> (<v-5dan-m> m i))
254: (<v-5dan-i> <--> (<v-5dan-r> r i))
255: (<v-5dan-i> <--> (<v-5dan-g> g i))
256: (<v-5dan-i> <--> (<v-5dan-b> b i))
257: (<v-5dan-i> <--> (<v-aru> r i))
258: (<v-5dan-u> <--> (<v-5dan-w> u))
259: (<v-5dan-u> <--> (<v-5dan-kt> k Uu))
260: (<v-5dan-u> <--> (<v-5dan-ki> k Uu))
261: (<v-5dan-u> <--> (<v-5dan-s> s Uu))
262: (<v-5dan-u> <--> (<v-5dan-t> ts Uu))
263: (<v-5dan-u> <--> (<v-5dan-n> n u))
264: (<v-5dan-u> <--> (<v-5dan-m> m u))
265: (<v-5dan-u> <--> (<v-5dan-r> r u))
266: (<v-5dan-u> <--> (<v-5dan-g> g u))
267: (<v-5dan-u> <--> (<v-5dan-b> b u))
268: (<v-5dan-u> <--> (<v-aru> r u))
269: (<v-5dan-e> <--> (<v-5dan-w> e))
270: (<v-5dan-e> <--> (<v-5dan-kt> k e))
271: (<v-5dan-e> <--> (<v-5dan-ki> k e))
272: (<v-5dan-e> <--> (<v-5dan-s> s e))
273: (<v-5dan-e> <--> (<v-5dan-t> t e))
274: (<v-5dan-e> <--> (<v-5dan-n> n e))
275: (<v-5dan-e> <--> (<v-5dan-m> m e))
276: (<v-5dan-e> <--> (<v-5dan-r> r e))
277: (<v-5dan-e> <--> (<v-5dan-g> g e))
278: (<v-5dan-e> <--> (<v-5dan-b> b e))
279: (<v-5dan-e> <--> (<v-aru> r e))
280: (<v-5dan-o> <--> (<v-5dan-w> o))
281: (<v-5dan-o> <--> (<v-5dan-kt> k o))
282: (<v-5dan-o> <--> (<v-5dan-ki> k o))
283: (<v-5dan-o> <--> (<v-5dan-s> s o))
284: (<v-5dan-o> <--> (<v-5dan-t> t o))
285: (<v-5dan-o> <--> (<v-5dan-n> n o))
286: (<v-5dan-o> <--> (<v-5dan-m> m o))
```

```
287: (<v-5dan-o> <--> (<v-5dan-r> r o))
288: (<v-5dan-o> <--> (<v-5dan-g> g o))
289: (<v-5dan-o> <--> (<v-5dan-b> b o))
290: (<v-mizen1> <--> (<v-5dan-a>))
291: (<v-mizen1> <--> (<v-1dan>))
292: (<v-mizen1> <--> (k o))
293: (<v-mizen1> <--> (<v-sahen> sh Ui))
294: (<v-mizen1> <--> (sh Ui))
295: (<v-mizen1> <--> (<v-fsahen> s a))
296: (<v-mizen1> <--> (<v-fzahen> z i))
297: (<v-mizen2> <--> (<v-5dan-o>))
298: (<v-mizen2> <--> (<v-1dan> y o))
299: (<v-mizen2> <--> (k o y o))
300: (<v-mizen2> <--> (<v-sahen> sh Ui y o))
301: (<v-mizen2> <--> (sh Ui y o))
302: (<v-mizen2> <--> (<v-fsahen> s o))
303: (<v-mizen2> <--> (<v-fzahen> z i y o))
304: (<v-mizen3> <--> (<v-5dan-a>))
305: (<v-mizen3> <--> (<v-1dan> s a))
306: (<v-mizen3> <--> (k o s a))
307: (<v-mizen3> <--> (<v-sahen> s a))
308: (<v-mizen3> <--> (s a))
309: (<v-mizen3> <--> (<v-fsahen> s a))
310: (<v-mizen3> <--> (<v-fzahen> z i s a))
311: (<v-mizen4> <--> (<v-5dan-a>))
312: (<v-mizen4> <--> (<v-1dan> r a))
313: (<v-mizen4> <--> (k o r a))
314: (<v-mizen4> <--> (<v-sahen> s a))
315: (<v-mizen4> <--> (s a))
316: (<v-mizen4> <--> (<v-fsahen> s a))
317: (<v-mizen4> <--> (<v-fzahen> z i r a))
318: (<v-mizen5> <--> (<v-5dan-a>))
319: (<v-mizen5> <--> (<v-1dan>))
320: (<v-mizen5> <--> (k o))
321: (<v-mizen5> <--> (<v-sahen> s e))
322: (<v-mizen5> <--> (s e))
323: (<v-mizen5> <--> (<v-fsahen> s a))
324: (<v-mizen5> <--> (<v-fzahen> z i))
325: (<v-renyo1> <--> (<v-5dan-i>))
326: (<v-renyo1> <--> (<v-1dan>))
327: (<v-renyo1> <--> (k Ui))
328: (<v-renyo1> <--> (<v-sahen> sh Ui))
329: (<v-renyo1> <--> (sh Ui))
330: (<v-renyo1> <--> (<v-fsahen> sh Ui))
331: (<v-renyo1> <--> (<v-fzahen> z i))
332: (<v-renyo2-t> <--> (<v-5dan-w> Q))
333: (<v-renyo2-t> <--> (<v-5dan-kt> Q))
334: (<v-renyo2-t> <--> (<v-5dan-ki> i))
335: (<v-renyo2-t> <--> (<v-5dan-s> sh Ui))
336: (<v-renyo2-t> <--> (<v-5dan-t> Q))
337: (<v-renyo2-d> <--> (<v-5dan-n> N))
338: (<v-renyo2-d> <--> (<v-5dan-m> N))
339: (<v-renyo2-t> <--> (<v-5dan-r> Q))
340: (<v-renyo2-d> <--> (<v-5dan-g> i))
341: (<v-renyo2-d> <--> (<v-5dan-b> N))
342: (<v-renyo2-t> <--> (<v-1dan>))
343: (<v-renyo2-t> <--> (k Ui))
344: (<v-renyo2-t> <--> (<v-sahen> sh Ui))
345: (<v-renyo2-t> <--> (sh Ui))
346: (<v-renyo2-t> <--> (<v-fsahen> sh Ui))
347: (<v-renyo2-t> <--> (<v-fzahen> z i))
348: (<v-rentai> <--> (<v-5dan-u>))
349: (<v-rentai> <--> (<v-1dan> r u))
350: (<v-rentai> <--> (k Uu r u))
351: (<v-rentai> <--> (<v-sahen> s Uu r u))
```

```
352: (<v-rentai> <--> (s Uu r u))
353: (<v-rentai> <--> (<v-fsahen> s Uu r u))
354: (<v-rentai> <--> (<v-fzahen> z u r u))
355: (<v-katei> <--> (<v-5dan-e>))
356: (<v-katei> <--> (<v-1dan> r e))
357: (<v-katei> <--> (k Uu r e))
358: (<v-katei> <--> (<v-sahen> s Uu r e))
359: (<v-katei> <--> (s Uu r e))
360: (<v-katei> <--> (<v-fsahen> s Uu r e))
361: (<v-katei> <--> (<v-fzahen> z u r e))
362: (<n> <--> (<number-word-time>))
363: (<number-word-time> <--> (<number-word-9gatu>))
364: (<number-word-time> <--> (<number-word-4gatu>))
365: (<number-word-time> <--> (<number-word>))
366: (<calendar> <--> (<month-word> <date-word>))
367: (<calendar> <--> (<month-word> <date-word> <day-of-week>))
368: (<calendar> <--> (<date-word>))
369: (<calendar> <--> (<date-word> <day-of-week>))
370: (<calendar> <--> (<month-word>))
371: (<calendar> <--> (<year>))
372: (<calendar> <--> (<year> <month>))
373: (<calendar> <--> (<year> <month> <date>))
374: (<calendar> <--> (<year> <month> <date> <day-of-week>))
375: (<calendar> <--> (<month> <date>))
376: (<calendar> <--> (<month> <date> <day-of-week>))
377: (<calendar> <--> (<date>))
378: (<calendar> <--> (<date> <day-of-week>))
379: (<calendar> <--> (<day-of-week>))
380: (<calendar> <--> (<month>))
381: (<date-word> <--> (<special-date-word>))
382: (<special-date-word> <--> (ts Uu i t a ch Ui))
383: (<date-word> <--> (<date-w> <day-word-ka>))
384: (<date-word> <--> (<date-w> <day-word-niti>))
385: (<date-w> <--> (h Uu ts Uu))
386: (<date-w> <--> (m i Q))
387: (<date-w> <--> (y o Q))
388: (<date-w> <--> (i ts Uu))
389: (<date-w> <--> (m u i))
390: (<date-w> <--> (n a n o))
391: (<date-w> <--> (y ou))
392: (<date-w> <--> (k o k o n o))
393: (<date-w> <--> (t ou))
394: (<date-w> <--> (zy uu i ch Ui))
395: (<date-w> <--> (zy uu n i))
396: (<date-w> <--> (zy uu s a N))
397: (<date-w> <--> (zy uu y o Q))
398: (<date-w> <--> (zy uu g o))
399: (<date-w> <--> (zy uu r o k Uu))
400: (<date-w> <--> (zy uu sh Ui ch Ui))
401: (<date-w> <--> (zy uu n a n a))
402: (<date-w> <--> (zy uu h a ch Ui))
403: (<date-w> <--> (zy uu k Uu))
404: (<date-w> <--> (zy uu ky u))
405: (<date-w> <--> (h a ts Uu k a))
406: (<date-w> <--> (n i zy uu i ch Ui))
407: (<date-w> <--> (n i zy uu n i))
408: (<date-w> <--> (n i zy uu s a N))
409: (<date-w> <--> (n i zy uu y o Q))
410: (<date-w> <--> (n i zy uu g o))
411: (<date-w> <--> (n i zy uu r o k Uu))
412: (<date-w> <--> (n i zy uu sh Ui ch Ui))
413: (<date-w> <--> (n i zy uu n a n a))
414: (<date-w> <--> (n i zy uu h a ch Ui))
415: (<date-w> <--> (n i zy uu k Uu))
416: (<date-w> <--> (n i zy uu ky uu))
```

```
417: (<date-w> <--> (s a N zy uu))
418: (<date-w> <--> (s a N zy uu i ch Ui))
419: (<date-w> <--> (r o k Uu zy uu i ch Ui))
420: (<day-word-niti> <--> (n i ch Ui))
421: (<day-word-ka> <--> (k a))
422: (<month> <--> (<num-month> g a ts Uu))
423: (<month-word> <--> (<number-word> g a ts Uu))
424: (<month-word> <--> (<number-word-9gatu> g a ts Uu))
425: (<year> <--> (<num-year> n e N))
426: (<day-of-week> <--> (<youbi> b i))
427: (<day-of-week> <--> (<youbi>))
428: (<youbi> <--> (g e ts Uu y ou))
429: (<youbi> <--> (k a y ou))
430: (<youbi> <--> (s Uu i y ou))
431: (<youbi> <--> (m o k Uu y ou))
432: (<youbi> <--> (k Ui N y ou))
433: (<youbi> <--> (d o y ou))
434: (<youbi> <--> (n i ch Ui y ou))
435: (<relative-time> <--> (<duration> <time-direction>))
436: (<time-direction> <--> (m a e))
437: (<time-direction> <--> (g o))
438: (<time-direction> <--> (a t o))
439: (<time-direction> <--> (n o ch Ui))
440: (<oki> <--> (o k Ui))
441: (<goto> <--> (g o t o))
442: (<frequency> <--> (<freq-with-times> <times>))
443: (<frequency> <--> (<freq-without-times>))
444: (<freq-with-times> <--> (<freq-spec> n i))
445: (<freq-with-times> <--> (<duration> n i))
446: (<freq-with-times> <--> (<duration> d e))
447: (<freq-with-times> <--> (<duration> n o a i d a n i))
448: (<freq-with-times> <--> (<freq-without-particle))
449: (<freq-without-times> <--> (<freq-spec> n i))
450: (<freq-without-times> <--> (<freq-without-particle>))
451: (<freq-spec> <--> (<duration> <goto>))
452: (<freq-spec> <--> (<step-duration>))
453: (<freq-without-particle> <--> (<unit-duration>))
454: (<duration> <--> (<time-unit>))
455: (<duration> <--> (<time-unit> n i))
456: (<time-unit> <--> (<prim-time-unit>))
457: (<time-unit> <--> (<temp-appro-prefix> <prim-time-unit>))
458: (<time-unit> <--> (<prim-time-unit> <temp-suffix>))
459: (<time-unit> <--> (<temp-appro-prefix> <prim-time-unit> <temp-rel-suffix>))
460: (<time-unit> <--> (<temp-appro-prefix> <prim-time-unit> <temp-appro-suffix>))
461: (<temp-appro-prefix> <--> (y a k Uu))
462: (<temp-appro-prefix> <--> (o y o s o))
463: (<temp-suffix> <--> (<temp-rel-suffix>))
464: (<temp-suffix> <--> (<temp-appro-suffix>))
465: (<temp-rel-suffix> <--> (i zy ou))
466: (<temp-rel-suffix> <--> (i n a i))
467: (<temp-rel-suffix> <--> (m i m a N))
468: (<temp-appro-suffix> <--> (t ei d o))
469: (<temp-appro-suffix> <--> (h o d o))
470: (<prim-time-unit> <--> (<hour-unit>))
471: (<prim-time-unit> <--> (<min-unit>))
472: (<prim-time-unit> <--> (<min-unit> k a N))
473: (<prim-time-unit> <--> (<hour-unit> <min-unit>))
474: (<prim-time-unit> <--> (<hour-unit> <min-unit> <sec-unit>))
475: (<prim-time-unit> <--> (<min-unit> <sec-unit>))
476: (<prim-time-unit> <--> (<hour-unit> <min-unit> k a N))
477: (<hour-unit> <--> (<time-numeral> <hour-unit-word>))
478: (<hour-unit-word> <--> (z i k a N))
479: (<min-unit> <--> (<time-numeral> <time-min-word-hun>))
480: (<min-unit> <--> (<time-numeral> <time-min-word-pun>))
481: (<min-unit-word> <--> (h Uu N))
```

```
482: (<prim-time-unit> <--> (<time-numeral> <day-word-ka>))
483: (<prim-time-unit> <--> (<time-numeral> <day-word-niti>))
484: (<prim-time-unit> <--> (<date-word>))
485: (<prim-time-unit> <--> (<date-word> k a N))
486: (<day-unit-word> <--> (n i ch Ui))
487: (<prim-time-unit> <--> (<time-numeral> <week-unit-word>))
488: (<prim-time-unit> <--> (<time-numeral> <week-unit-word> k a N))
489: (<week-unit-word> <--> (sy uu))
490: (<prim-time-unit> <--> (<time-numeral> <kagetu>))
491: (<prim-time-unit> <--> (<time-numeral> <kagetu> k a N))
492: (<kagetu> <--> (k a g e ts Uu))
493: (<month-unit-word> <--> (ts Uu k Ui))
494: (<prim-time-unit> <--> (<time-numeral> <year-unit-word>))
495: (<prim-time-unit> <--> (<time-numeral> <year-unit-word> k a N))
496: (<year-unit-word> <--> (n e N))
497: (<time-numeral> <--> (<number-word>))
498: (<relative-time> <--> (<duration-present>))
499: (<relative-time> <--> (<duration-past>))
500: (<relative-time> <--> (<duration-future>))
501: (<duration-present> <--> (<day-exp-today>))
502: (<duration-past> <--> (<day-exp-yesterday>))
503: (<duration-future> <--> (<day-exp-tomorrow>))
504: (<day-exp-today> <--> (<today>))
505: (<today> <--> (ky ou))
506: (<today> <--> (h o N z i ts Uu))
507: (<day-exp-yesterday> <--> (<yesterday>))
508: (<yesterday> <--> (k Ui n ou))
509: (<yesterday> <--> (s a k Uu z i ts Uu))
510: (<day-exp-tomorrow> <--> (<tomorrow>))
511: (<tomorrow> <--> (a sh Ui t a))
512: (<tomorrow> <--> (my ou n i ch Ui))
513: (<relative-time> <--> (<duration-day-segment-present>))
514: (<duration-day-segment-present> <--> (<duration-present> n o <day-segment>))
515: (<relative-time> <--> (<duration-day-segment-past>))
516: (<duration-day-segment-past> <--> (<duration-past> n o <day-segment>))
517: (<relative-time> <--> (<duration-day-segment-future>))
518: (<duration-day-segment-future> <--> (<duration-future> n o <day-segment>))
519: (<relative-time> <--> (<day-segment-present>))
520: (<day-segment-present> <--> (<day-segment>))
521: (<day-segment> <--> (a s a))
522: (<day-segment> <--> (h Ui r u))
523: (<day-segment> <--> (b a N))
524: (<day-segment> <--> (y o r u))
525: (<day-segment> <--> (g o z e N))
526: (<day-segment> <--> (g o g o))
527: (<this-morning> <--> (k e s a))
528: (<this-morning-2> <--> (<this-morning>))
529: (<relative-time> <--> (<this-morning-2>))
530: (<v-sahen> <--> (t ou k ou))
531: (<v-sahen> <--> (y ou i))
532: (<v-sahen> <--> (sh Ui Q p Ui ts Uu))
533: (<v-sahen> <--> (t e N p Uu))
534: (<v-sahen> <--> (s ei g e N))
535: (<v-sahen> <--> (sh Ui y ou))
536: (<v-sahen> <--> (r i y ou))
537: (<v-sahen> <--> (y o t ei))
538: (<v-sahen> <--> (h o N y a k Uu))
539: (<v-sahen> <--> (z i d ou h o N y a k Uu))
540: (<v-sahen> <--> (cy ou k ou))
541: (<v-sahen> <--> (t ou r o k Uu))
542: (<v-sahen> <--> (s a N k a))
543: (<v-sahen> <--> (h a Q py ou))
544: (<v-sahen> <--> (m ei k Ui))
545: (<v-sahen> <--> (k Ui b ou))
546: (<v-sahen> <--> (g o k Ui b ou))
```

547: (<v-sahen> <--> (k a k Uu n i N))
548: (<v-sahen> <--> (o k Uu r i))
549: (<v-sahen> <--> (o n e g a i))
550: (<v-aru> <--> (a))
551: (<kyo> <--> (o k Ui))
552: (<kdo> <--> (h Ui ts Uu y ou))
553: (<det> <--> (k o n o))
554: (<det> <--> (s o n o))
555: (<det> <--> (a n o))
556: (<whn> <--> (d o k o))
557: (<whn> <--> (i ts Uu))
558: (<whadv> <--> (i ts Uu))
559: (<whn> <--> (n a N z i))
560: (<whadv> <--> (n a N z i n i))
561: (<whadv> <--> (n a N <pun> n i))
562: (<whadv> <--> (n a N z i n a N <pun> n i))
563: (<whadv> <--> (n a N n i ch Ui n i))
564: (<whn> <--> (n a N))
565: (<whn> <--> (d o n o g u r a i))
566: (<whadv> <--> (d o n o g u r a i))
567: (<whn> <--> (d a r e))
568: (<whadv> <--> (d ou sh Ui t e))
569: (<whadv> <--> (n a z e))
570: (<whadv> <--> (d ou))
571: (<whn> <--> (d o r e))
572: (<whadv> <--> (n a N k a i))
573: (<whadv> <--> (n a N d o))
574: (<whadv> <--> (d o r e d a k e))
575: (<whadv> <--> (s o n o))
576: (<conj-s> <--> (s ou d e s Uu k a))
577: (<conj-s> <--> (w a k a r i m a sh Ui t a))
578: (<n> <--> (<lname>))
579: (<n> <--> (<fname> <lname>))
580: (<n> <--> (<fname>))
581: (<n> <--> (<name>))
582: (<name> <--> (<name> s a N))
583: (<name> <--> (<lname> s a m a))
584: (<fname> <--> (i ch Ui t a r ou))
585: (<lname> <--> (t a k a r a))
586: (<pron> <--> (w a t a sh Ui))
587: (<pron> <--> (w a t a sh Ui t a ch Ui))
588: (<n> <--> (k o ch Ui r a))
589: (<n> <--> (n a N k a))
590: (<n> <--> (k o r e))
591: (<n> <--> (s o r e))
592: (<n> <--> (a r e))
593: (<n> <--> (<relative-time>))
594: (<n> <--> (<calendar>))
595: (<n> <--> (<time-of-day>))
596: (<adv> <--> (<frequency>))
597: (<adv> <--> (<relative-time>))
598: (<adv> <--> (<duration>))
599: (<adv> <--> (<det> <conj> n i))
600: (<adv> <--> (<det> <conj> w a))
601: (<adv> <--> (<det> <conj> n i w a))
602: (<adv> <--> (<det> <conj>))
603: (<adv> <--> (<count>))
604: (<n> <--> (<count>))
605: (<count> <--> (h Ui t o ts Uu))
606: (<count> <--> (h Uu t a ts Uu))
607: (<count> <--> (m i Q ts Uu))
608: (<adv> <--> (<count-hai>))
609: (<n> <--> (<count-hai>))
610: (<number-word> <--> (<number-word-x>))
611: (<number-word> <--> (<number-word-g>))

```
612: (<number-word> <--> (<number-word-3>))
613: (<number-word-x> <--> (i ch Ui))
614: (<number-word-g> <--> (n i))
615: (<number-word-3> <--> (s a N))
616: (<number-word-4gatu> <--> (sh Ui))
617: (<number-word-g> <--> (y o N))
618: (<number-word-g> <--> (g o))
619: (<number-word-x> <--> (r o k Uu))
620: (<number-word-g> <--> (sh Ui ch Ui))
621: (<number-word-g> <--> (n a n a))
622: (<number-word-x> <--> (h a ch Ui))
623: (<number-word-9gatu> <--> (k Uu))
624: (<number-word-g> <--> (ky uu))
625: (<number-word> <--> (zy uu))
626: (<number-word> <--> (zy uu i ch Ui))
627: (<number-word> <--> (zy uu n i))
628: (<number-word-p> <--> (i Q))
629: (<number-word-p> <--> (r o Q))
630: (<number-word-p> <--> (h a Q))
631: (<number-word-p> <--> (zy u Q))
632: (<number-word-k> <--> (i Q))
633: (<number-word-k> <--> (r o Q))
634: (<number-word-k> <--> (h a Q))
635: (<number-word-k> <--> (zy u Q))
636: (<adv> <--> (<times>))
637: (<times> <--> (<number-word-k> k a i))
638: (<times> <--> (<number-word-g> k a i))
639: (<times> <--> (<number-word-3> k a i))
640: (<times> <--> (<number-word> d o))
641: (suzuki <--> (s Uu z u k Ui))
642: (yamamoto <--> (y a m a m o t o))
643: (<kyo> <--> (t a k a))
644: (<kyo> <--> (n i k Uu))
645: (<kyo> <--> (m u ts Uu k a sh Ui))
646: (<kyo> <--> (o s o))
647: (<kyo> <--> (i s o g a sh Ui))
648: (<kyo> <--> (a t a r a sh Ui))
649: (<kyo> <--> (t a n o sh Ui))
650: (<kyo> <--> (s a m u))
651: (<kyo> <--> (o m o sh Ui r o))
652: (<kyo> <--> (y a s Uu))
653: (<kyo> <--> (t oo))
654: (<kyo> <--> (n a g a))
655: (<kyo> <--> (h a y a))
656: (<kyo> <--> (ch Ui k a))
657: (<kyo> <--> (o))
658: (<kyo> <--> (y o r o sh Ui))
659: (<kyo> <--> (k Uu w a sh Ui))
660: (<kyo> <--> (y o))
661: (<adv> <--> (ts Uu m a r i))
662: (<adv> <--> (t a d a i m a))
663: (<adv> <--> (y o k Uu))
664: (<adv> <--> (g o t o n i))
665: (<adv> <--> (a r u i h a))
666: (<adv> <--> (m a N i ch Ui))
667: (<adv> <--> (d a i zy ou b u))
668: (<adv> <--> (s Uu k Uu n a k Uu t o m o))
669: (<adv> <--> (m a Q t a k Uu))
670: (<adv> <--> (i Q k ou))
671: (<adv> <--> (t a t o e b a))
672: (<adv> <--> (o k Uu))
673: (<adv> <--> (n a r u h o d o))
674: (<adv> <--> (d ou sh Ui t e m o))
675: (<adv> <--> (t a i h e N))
676: (<adv> <--> (t a i t ei))
```

677: (<adv> <--> (n a n i k a))
678: (<adv> <--> (n a N k a))
679: (<adv> <--> (i ts Uu m a d e))
680: (<adv> <--> (h o N t ou n i))
681: (<adv> <--> (z e h Ui t o m o))
682: (<adv> <--> (d a i b u))
683: (<adv> <--> (h Ui t o m a z u))
684: (<adv> <--> (i k a g a))
685: (<adv> <--> (y a h a r i))
686: (<adv> <--> (m a t a))
687: (<adv> <--> (t a sy ou))
688: (<adv> <--> (m a e m o Q t e))
689: (<adv> <--> (o s o r a k Uu))
690: (<adv> <--> (m a d e))
691: (<adv> <--> (k Ui Q t o))
692: (<adv> <--> (k a n a r a z u))
693: (<adv> <--> (cy ou d o))
694: (<adv> <--> (n a n i m o))
695: (<adv> <--> (b e ts Uu n i))
696: (<adv> <--> (s o r e z o r e))
697: (<adv> <--> (s o r e s o r e))
698: (<adv> <--> (y o k ei))
699: (<adv> <--> (i ch Ui b a N))
700: (<adv> <--> (d a i t a i))
701: (<adv> <--> (t o k Uu n i))
702: (<adv> <--> (z e N z e N))
703: (<adv> <--> (cy o Q t o))
704: (<adv> <--> (ts Uu g i n i))
705: (<adv> <--> (s Uu g u))
706: (<adv> <--> (y a k Uu))
707: (<adv> <--> (t o m o n i))
708: (<adv> <--> (h a Q k Ui r i))
709: (<adv> <--> (d ou k a))
710: (<adv> <--> (n a r u b e k Uu))
711: (<adv> <--> (k a w a r i n i))
712: (<adv> <--> (i Q t a N))
713: (<adv> <--> (m o ch Ui r o N))
714: (<adv> <--> (m a))
715: (<adv> <--> (s Uu b e t e))
716: (<adv> <--> (d ou z o))
717: (<adv> <--> (s Uu k o sh Ui))
718: (<adv> <--> (cy o k Uu s e ts Uu))
719: (<adv> <--> (z e h Ui))
720: (<adv> <--> (t a k Uu s a N))
721: (<adv> <--> (t a b u N))
722: (<adv> <--> (i k Uu r a))
723: (<adv> <--> (s a Q s o k Uu))
724: (<adv> <--> (m ou))
725: (<adv> <--> (m a z u))
726: (<adv> <--> (d ou))
727: (<adv> <--> (d ou m o))
728: (<adv> <--> (s a k Ui h o d o))
729: (<adv> <--> (k a n a r i))
730: (<adv> <--> (a m a r i))
731: (<adv> <--> (i Q p a i))
732: (<adv> <--> (y o r o sh Ui k Uu))
733: (<adv> <--> (m o sh Ui))
734: (<adv> <--> (m a d a))
735: (<adv> <--> (s Uu d e n i))
736: (<adv> <--> (s ou))
737: (<adv> <--> (d ou i u))
738: (<adv> <--> (k o N n a))
739: (<adv> <--> (a r a y u r u))
740: (<adv> <--> (d o N n a))
741: (<adv> <--> (s ou i u))

```
742: (<adv> <--> (s o n o))
743: (<adv> <--> (s o r e))
744: (<adv> <--> (k o n o))
745: (<adv> <--> (d o n o))
746: (<adv> <--> (y ou n a))
747: (<adv> <--> (y ou n i))
748: (<adv> <--> (d o n o y ou n a))
749: (<adv> <--> (m o sh Ui k Uu h a))
750: (<adv> <--> (sh Ui k a sh Ui))
751: (<adv> <--> (s Uu n a w a ch Ui))
752: (<adv> <--> (t e))
753: (<adv> <--> (d e m o))
754: (<adv> <--> (zy a))
755: (<adv> <--> (s Uu r u t o))
756: (<adv> <--> (s o r e d e))
757: (<adv> <--> (i m a))
758: (<adv> <--> (z i ts Uu h a))
759: (<adv> <--> (s a t e))
760: (<adv> <--> (s o r e d e m o))
761: (<adv> <--> (s o k o d e))
762: (<adv> <--> (m a t a w a))
763: (<adv> <--> (s o r e k a r a))
764: (<adv> <--> (t a d a))
765: (<adv> <--> (t a d a sh Ui))
766: (<adv> <--> (t o k o r o d e))
767: (<adv> <--> (o y o b i))
768: (<adv> <--> (s o r e t o m o))
769: (<adv> <--> (k e r e d o m o))
770: (<adv> <--> (s o sh Ui t e))
771: (<adv> <--> (d e w a))
772: (<adv> <--> (s o r e d e w a))
773: (<adv> <--> (o h a y ou))
774: (<adv> <--> (s Uu m i m a s e M))
775: (<adv> <--> (i e))
776: (<adv> <--> (s a y o n a r a))
777: (<adv> <--> (s a y ou n a r a))
778: (<adv> <--> (a r i g a t ou))
779: (<adv> <--> (h a i))
780: (<adv> <--> (m o sh Ui m o sh Ui))
781: (<v-5dan-s> <--> (k Uu r i k a e))
782: (<v-1dan> <--> (sh Ui r e))
783: (<v-1dan> <--> (k a e r e))
784: (<v-5dan-s> <--> (s Uu g o))
785: (<v-5dan-r> <--> (k Ui))
786: (<v-1dan> <--> (k a m a e))
787: (<v-5dan-g> <--> (sh Ui n o))
788: (<v-1dan> <--> (u k e))
789: (<v-5dan-r> <--> (a t a))
790: (<v-5dan-w> <--> (h a n a sh Ui a))
791: (<v-5dan-s> <--> (h a n a))
792: (<v-1dan> <--> (k a k e))
793: (<v-1dan> <--> (k o e))
794: (<v-5dan-s> <--> (h a r a i m o d o))
795: (<v-1dan> <--> (m ou sh Ui u k e))
796: (<v-5dan-r> <--> (h a z i m a))
797: (<v-5dan-ki> <--> (t o d o))
798: (<v-5dan-s> <--> (d a))
799: (<v-1dan> <--> (a ts Uu m e))
800: (<v-5dan-b> <--> (y o r o k o))
801: (<v-5dan-m> <--> (t a n o sh Ui))
802: (<v-5dan-r> <--> (ts Uu k Uu))
803: (<v-5dan-w> <--> (t e ts Uu d a))
804: (<v-1dan> <--> (sy a b e r e))
805: (<v-5dan-r> <--> (sh Ui a g a))
806: (<v-5dan-r> <--> (s e m a))
```

807: (<v-5dan-w> <--> (sh Ui t a g a))
808: (<v-5dan-m> <--> (t a n o))
809: (<v-1dan> <--> (ou z i))
810: (<v-sahen> <--> (t a i))
811: (<v-5dan-m> <--> (y o))
812: (<v-5dan-r> <--> (sh Ui m e k Ui))
813: (<v-1dan> <--> (t o r e))
814: (<v-1dan> <--> (i e))
815: (<v-5dan-r> <--> (k a w a))
816: (<v-1dan> <--> (ts Uu t a e))
817: (<v-5dan-r> <--> (u k e t o))
818: (<v-5dan-w> <--> (t o r i a ts Uu k a))
819: (<v-5dan-ki> <--> (k Ui))
820: (<v-5dan-ki> <--> (o))
821: (<v-5dan-r> <--> (t o m a))
822: (<v-5dan-m> <--> (k o))
823: (<v-5dan-r> <--> (o w a))
824: (<v-5dan-s> <--> (a r a w a))
825: (<v-5dan-r> <--> (m a ts Uu w a))
826: (<v-1dan> <--> (i r e))
827: (<v-5dan-r> <--> (k a k a))
828: (<v-5dan-ki> <--> (o ch Ui ts Uu))
829: (<v-5dan-r> <--> (y a))
830: (<v-5dan-s> <--> (y u r u))
831: (<v-5dan-r> <--> (m a w a))
832: (<v-5dan-r> <--> (n a s a))
833: (<v-5dan-r> <--> (k Ui m a))
834: (<v-5dan-b> <--> (a s o))
835: (<v-5dan-m> <--> (s Uu))
836: (<v-5dan-ki> <--> (ts Uu))
837: (<v-1dan> <--> (n a r e))
838: (<v-1dan> <--> (n o r i k a e))
839: (<v-5dan-g> <--> (n o r i ts Uu))
840: (<v-5dan-r> <--> (k a sh Ui k o m a))
841: (<v-5dan-r> <--> (u k e t a m a w a))
842: (<v-1dan> <--> (d e))
843: (<v-1dan> <--> (m ou sh Ui a g e))
844: (<v-5dan-ki> <--> (a r u))
845: (<v-1dan> <--> (k a N g a e))
846: (<v-5dan-ki> <--> (k a))
847: (<v-5dan-m> <--> (h Uu r i k o))
848: (<v-5dan-m> <--> (h a s a))
849: (<v-5dan-s> <--> (m ou))
850: (<v-1dan> <--> (m i s e))
851: (<v-5dan-r> <--> (t o))
852: (<v-1dan> <--> (u k e ts Uu k e))
853: (<v-5dan-r> <--> (d e k Ui a g a))
854: (<v-5dan-ki> <--> (h Ui r a))
855: (<v-1dan> <--> (t a n o m e))
856: (<v-5dan-w> <--> (k a m a))
857: (<v-1dan> <--> (sh Ui r a s e))
858: (<v-1dan> <--> (s o e))
859: (<v-1dan> <--> (y o m i a g e))
860: (<v-1dan> <--> (i t a d a k e))
861: (<v-5dan-r> <--> (i r a Q sy a))
862: (<v-1dan> <--> (o r i))
863: (<v-5dan-r> <--> (n o))
864: (<v-5dan-m> <--> (m a k Ui k o))
865: (<v-1dan> <--> (m ou k e))
866: (<v-5dan-w> <--> (u k a g a))
867: (<v-5dan-s> <--> (i t a))
868: (<v-5dan-r> <--> (w a k a))
869: (<v-5dan-ki> <--> (ch Ui k a d u))
870: (<v-5dan-ki> <--> (i))
871: (<v-5dan-r> <--> (o Q sy a))

```
872: (<v-5dan-s> <--> (k a e))
873: (<v-1dan> <--> (h a r a e))
874: (<v-5dan-r> <--> (y o))
875: (<v-5dan-ki> <--> (ts Uu d u))
876: (<v-1dan> <--> (o k o n a w a r e))
877: (<v-5dan-r> <--> (sh Ui))
878: (<v-5dan-r> <--> (t a z u s a w a))
879: (<v-5dan-r> <--> (h a i))
880: (<v-1dan> <--> (h Uu k Uu m a r e))
881: (<v-5dan-w> <--> (sh Ui h a r a))
882: (<v-5dan-ki> <--> (i t a d a))
883: (<v-1dan> <--> (m a t o m e))
884: (<v-1dan> <--> (o sh Ui e))
885: (<v-5dan-g> <--> (i s o))
886: (<v-5dan-t> <--> (m a))
887: (<v-1dan> <--> (m i e))
888: (<v-5dan-s> <--> (w a t a))
889: (<v-1dan> <--> (m o r a e))
890: (<v-1dan> <--> (k Ui k e))
891: (<v-1dan> <--> (i k e))
892: (<v-1dan> <--> (y u k e))
893: (<v-1dan> <--> (t a z u n e))
894: (<v-5dan-w> <--> (o k o n a))
895: (<v-5dan-w> <--> (i))
896: (<v-1dan> <--> (k a N s Uu))
897: (<v-1dan> <--> (ts Uu r e))
898: (<v-5dan-w> <--> (o m o))
899: (<v-1dan> <--> (d e k Ui))
900: (<v-5dan-r> <--> (o))
901: (<v-5dan-w> <--> (ts Uu k a))
902: (<v-5dan-r> <--> (n a))
903: (<v-5dan-w> <--> (h a r a))
904: (<v-1dan> <--> (i))
905: (<v-5dan-w> <--> (n e g a))
906: (<v-1dan> <--> (k Ui k a s e))
907: (<v-5dan-r> <--> (o k Uu))
908: (<v-5dan-t> <--> (m o))
909: (<v-5dan-m> <--> (m ou sh Ui k o))
910: (<v-1dan> <--> (m o ch Ui))
911: (<v> <--> (k Uu d a s a i))
912: (<v-1dan> <--> (h Uu k Uu m e))
913: (<v-5dan-m> <--> (h Uu k Uu))
914: (<v-1dan> <--> (a g e))
915: (<v-5dan-r> <--> (k a g i))
916: (<n> <--> (z i k o k Uu hy ou))
917: (<n> <--> (r i m u z i N))
918: (<n> <--> (d a N s ei))
919: (<n> <--> (h e N z i))
920: (<n> <--> (s a i sh Ui N))
921: (<n> <--> (ch ii m u))
922: (<n> <--> (t a i sy ou))
923: (<n> <--> (g oo k ei))
924: (<n> <--> (g a k Uu))
925: (<n> <--> (o N s ei))
926: (<n> <--> (r i k a i))
927: (<n> <--> (o N s ei r i k a i))
928: (<n> <--> (o N s ei n i N sh Ui k Ui))
929: (<n> <--> (n i N sh Ui k Ui))
930: (<n> <--> (sy u Q s e k Ui))
931: (<n> <--> (m e k a))
932: (<n> <--> (g i d a i))
933: (<n> <--> (s e Q sy o N))
934: (<n> <--> (k ou s ei))
935: (<n> <--> (k e s a))
936: (<n> <--> (d ou h a N sy a))
```

```
 937: (<n> <--> (n i N z u))
 938: (<n> <--> (zy ou z i))
 939: (<n> <--> (zy uu n i n i ch Ui))
 940: (<n> <--> (zy uu g a ts Uu))
 941: (<n> <--> (i k a))
 942: (<n> <--> (sy o k Uu b a))
 943: (<n> <--> (n i Q s Uu))
 944: (<n> <--> (h a z i m e t e))
 945: (<n> <--> (n o r i b a))
 946: (<n> <--> (d e g u ch Ui))
 947: (<n> <--> (e a p oo t o))
 948: (<n> <--> (k o t o))
 949: (<n> <--> (n e N))
 950: (<n> <--> (cy o k Uu z e N))
 951: (<n> <--> (ts Uu m o r i))
 952: (<n> <--> (m a m a))
 953: (<n> <--> (m o n o))
 954: (<n> <--> (s a i sy o))
 955: (<n> <--> (h e N t ou))
 956: (<n> <--> (n e N d o))
 957: (<n> <--> (s e N ky uu hy a k Uu ky uu zy uu <number-word> n e N))
 958: (<n> <--> (n e N g ou))
 959: (<n> <--> (t a b i t a b i))
 960: (<n> <--> (r a i n e N))
 961: (<n> <--> (s e w a))
 962: (<n> <--> (k Uu n i))
 963: (<n> <--> (k o n e))
 964: (<n> <--> (n e g a i))
 965: (<n> <--> (h Ui t o t o k Ui))
 966: (<n> <--> (m a i k o))
 967: (<n> <--> (g ei sy a))
 968: (<n> <--> (k a N k ei))
 969: (<n> <--> (z i m u sy o))
 970: (<n> <--> (a N n a i))
 971: (<n> <--> (k ei s ou))
 972: (<n> <--> (h Uu k Uu s ou))
 973: (<n> <--> (h Uu k Uu))
 974: (<n> <--> (h a z u))
 975: (<n> <--> (b e N r i))
 976: (<n> <--> (k ou ts uu))
 977: (<n> <--> (k ou ts uu r o))
 978: (<n> <--> (k ou ts uu sy u d a N))
 979: (<n> <--> (sy u d a N))
 980: (<n> <--> (n a N n i N))
 981: (<n> <--> (n a i))
 982: (<n> <--> (n o z o m i))
 983: (<n> <--> (m a ch Ui g a i))
 984: (<n> <--> (k o N t a k Uu t o))
 985: (<n> <--> (sy u w a))
 986: (<n> <--> (k Ui g ou))
 987: (<n> <--> (h a n a sh Ui a i))
 988: (<n> <--> (m o d e r u))
 989: (<n> <--> (k o N py uu t a))
 990: (<n> <--> (h a Q s ei))
 991: (<n> <--> (b u N s e k Ui))
 992: (<n> <--> (k a k Ui k o t o b a))
 993: (<n> <--> (h a n a sh Ui k o t o b a))
 994: (<n> <--> (p a N h Uu r e Q t o))
 995: (<n> <--> (k o k o r o))
 996: (<n> <--> (i k Ui))
 997: (<n> <--> (a t a r i))
 998: (<n> <--> (k a k Uu sh Ui k Ui))
 999: (<n> <--> (i ch Ui g e N))
1000: (<n> <--> (s o b a))
1001: (<n> <--> (n i h o N sh Ui k Ui))
```

```
1002: (<n> <--> (ry o k a N))
1003: (<n> <--> (i N k a i))
1004: (<n> <--> (s o sh Ui k Ui))
1005: (<n> <--> (d a i s a N sy a))
1006: (<n> <--> (t o Q p Ui k Uu s Uu))
1007: (<n> <--> (h a r a i k o m i))
1008: (<n> <--> (n i Q t ei))
1009: (<n> <--> (a t e))
1010: (<n> <--> (y ou y a k Uu))
1011: (<n> <--> (ou y ou))
1012: (<n> <--> (t e k Ui y ou))
1013: (<n> <--> (r i r o N))
1014: (<n> <--> (k a t ei))
1015: (<n> <--> (k a k Uu t o k Uu))
1016: (<n> <--> (t e s Uu))
1017: (<n> <--> (z e N g a k Uu))
1018: (<n> <--> (d e k Ui))
1019: (<n> <--> (sh ii i N g u))
1020: (<n> <--> (s a i t o))
1021: (<n> <--> (sy a k ou))
1022: (<n> <--> (y u g a t a))
1023: (<n> <--> (m u n e))
1024: (<n> <--> (g e N s o k Uu))
1025: (<n> <--> (k o r e k a r a))
1026: (<n> <--> (sh Ui d a i))
1027: (<n> <--> (d a N k a i))
1028: (<n> <--> (g e N z a i))
1029: (<n> <--> (s o n o t a))
1030: (<n> <--> (m e N))
1031: (<n> <--> (g i zy u ts Uu))
1032: (<n> <--> (n a g a r a))
1033: (<n> <--> (z a N n e N))
1034: (<n> <--> (k Ui r o k Uu))
1035: (<n> <--> (r o k Uu))
1036: (<n> <--> (z e N t a i))
1037: (<n> <--> (y ou i))
1038: (<n> <--> (i g a i))
1039: (<n> <--> (i N t a r e s Uu t o))
1040: (<n> <--> (s ei y a k Uu))
1041: (<n> <--> (s Uu p ii ch Ui))
1042: (<n> <--> (o k Uu r i s a k Ui))
1043: (<n> <--> (a p Uu r i k ee sy o N))
1044: (<n> <--> (s a m a r ii))
1045: (<n> <--> (h o N sy a))
1046: (<n> <--> (h Ui sy o))
1047: (<n> <--> (sh Ui sy a))
1048: (<n> <--> (m o N d a i))
1049: (<n> <--> (ts Uu g ou))
1050: (<n> <--> (zy u N b i))
1051: (<n> <--> (g i r i g i r i))
1052: (<n> <--> (t e g a m i))
1053: (<n> <--> (k a k Uu n i N))
1054: (<n> <--> (g e N k ou))
1055: (<n> <--> (s o n o h e N))
1056: (<n> <--> (k o N sy uu))
1057: (<n> <--> (h a y a m e))
1058: (<n> <--> (y ou sh Ui k Ui))
1059: (<n> <--> (s o r o e))
1060: (<n> <--> (k ei h Ui))
1061: (<n> <--> (k a t a ch Ui))
1062: (<n> <--> (s e N s ei))
1063: (<n> <--> (k ou ry o))
1064: (<n> <--> (g a Q k a))
1065: (<n> <--> (z i N m o N))
1066: (<n> <--> (z i k a i))
```

1067: (<n> <--> (gy ou sy a))
1068: (<n> <--> (y o y uu))
1069: (<n> <--> (k Ui z i ts Uu))
1070: (<n> <--> (ts Uu i k a))
1071: (<n> <--> (r e s e p Uu sy o N))
1072: (<n> <--> (z e N i N))
1073: (<n> <--> (sh Ui t a))
1074: (<n> <--> (g e N k Ui N))
1075: (<n> <--> (h Uu r i))
1076: (<n> <--> (r e z i s Uu t o r ee sy o N))
1077: (<n> <--> (s e ts Uu m ei))
1078: (<n> <--> (p ee p aa))
1079: (<n> <--> (h o))
1080: (<n> <--> (k o N d o))
1081: (<n> <--> (sh Ui t e N))
1082: (<n> <--> (t o r i h Ui k Ui))
1083: (<n> <--> (t o))
1084: (<n> <--> (k Ui k a i))
1085: (<n> <--> (k ou k a N))
1086: (<n> <--> (i k e N))
1087: (<n> <--> (k a i s a i))
1088: (<n> <--> (cy ou sy u))
1089: (<n> <--> (ky o n e N))
1090: (<n> <--> (m a t a))
1091: (<n> <--> (z i t e N))
1092: (<n> <--> (n i N))
1093: (<n> <--> (z i N))
1094: (<n> <--> (h Ui t o))
1095: (<n> <--> (s Uu p Ui k aa))
1096: (<n> <--> (y a r i k a t a))
1097: (<n> <--> (s e N m o N))
1098: (<n> <--> (s e N k ou))
1099: (<n> <--> (s e N z i ts Uu))
1100: (<n> <--> (g o z o N z i))
1101: (<n> <--> (z e N g o))
1102: (<n> <--> (d ou r o))
1103: (<n> <--> (t e))
1104: (<n> <--> (ky o r i))
1105: (<n> <--> (n e d a N))
1106: (<n> <--> (ky ou))
1107: (<n> <--> (g u a i))
1108: (<n> <--> (k Uu r u m a))
1109: (<n> <--> (s a N zy u n i ch Ui))
1110: (<n> <--> (k a w a s e))
1111: (<n> <--> (e N ry o))
1112: (<n> <--> (h Uu m ei))
1113: (<n> <--> (ry o k ou))
1114: (<n> <--> (ts Uu g i))
1115: (<n> <--> (g o r a N))
1116: (<n> <--> (m i h o N i ch Ui))
1117: (<n> <--> (h Ui N))
1118: (<n> <--> (y u ny u))
1119: (<n> <--> (d e m i s e))
1120: (<n> <--> (k ou e N))
1121: (<n> <--> (h e N s e N))
1122: (<n> <--> (b u N k a z a i))
1123: (<n> <--> (k a z u k a z u))
1124: (<n> <--> (k e))
1125: (<n> <--> (i h Ui N))
1126: (<n> <--> (k e N b u ts Uu))
1127: (<n> <--> (t ou cy a k Uu))
1128: (<n> <--> (d a i hy ou))
1129: (<n> <--> (s Uu d o m a r i))
1130: (<n> <--> (s a i k Ui N))
1131: (<n> <--> (y o s a))

```
1132: (<n> <--> (t o sh Ui N))
1133: (<n> <--> (m a w a r i))
1134: (<n> <--> (a sh Ui))
1135: (<n> <--> (t o m o d a ch Ui))
1136: (<n> <--> (k Ui N p e N))
1137: (<n> <--> (r a i sy u))
1138: (<n> <--> (ch Ui z u))
1139: (<n> <--> (k a k a r i))
1140: (<n> <--> (t ou sy a))
1141: (<n> <--> (k e N))
1142: (<n> <--> (k uu p o N))
1143: (<n> <--> (sh Ui h a r a i))
1144: (<n> <--> (sy u t e N))
1145: (<n> <--> (s e N))
1146: (<n> <--> (t o k Uu))
1147: (<n> <--> (t o k Uu t e N))
1148: (<n> <--> (cy ou sy o k Uu))
1149: (<n> <--> (p Uu r a N))
1150: (<n> <--> (ky a k Uu))
1151: (<n> <--> (zy o s ei))
1152: (<n> <--> (sy o k Uu z i))
1153: (<n> <--> (t ei d o))
1154: (<n> <--> (t ee d o))
1155: (<n> <--> (k o m i))
1156: (<n> <--> (z ei k Ui N))
1157: (<n> <--> (y o s a N))
1158: (<n> <--> (s e k Ui))
1159: (<n> <--> (t a ch Ui m i))
1160: (<n> <--> (ch Ui k e Q t o))
1161: (<n> <--> (k o N s aa t o))
1162: (<n> <--> (t o n a r i))
1163: (<n> <--> (h a n a m a ts Uu r i))
1164: (<n> <--> (h a r u y a s Uu m i))
1165: (<n> <--> (i b e N t o))
1166: (<n> <--> (k o d o m o))
1167: (<n> <--> (o t o n a))
1168: (<n> <--> (ny uu zy ou))
1169: (<n> <--> (m o y o r i))
1170: (<n> <--> (n o r i k a e))
1171: (<n> <--> (sy o y ou))
1172: (<n> <--> (k a N zy ou s e N))
1173: (<n> <--> (k uu k ou))
1174: (<n> <--> (h Ui k ou k Ui))
1175: (<n> <--> (sy o Q p Ui N g u))
1176: (<n> <--> (sy uu h e N))
1177: (<n> <--> (cy uu sh Ui N))
1178: (<n> <--> (ts uu ch Ui))
1179: (<n> <--> (zy ou h ou))
1180: (<n> <--> (h o k a))
1181: (<n> <--> (b i))
1182: (<n> <--> (n i ch Ui))
1183: (<n> <--> (h Ui))
1184: (<n> <--> (s a i sy uu))
1185: (<n> <--> (b a N))
1186: (<n> <--> (z e N z i ts Uu))
1187: (<n> <--> (ky ou m i))
1188: (<n> <--> (sh Ui s e ts Uu))
1189: (<n> <--> (k Ui N zy o))
1190: (<n> <--> (k ou sh Ui k Ui))
1191: (<n> <--> (k ou sh Ui k Ui g e N g o))
1192: (<n> <--> (k a i g a i))
1193: (<n> <--> (s o ts Uu gy ou))
1194: (<n> <--> (s ei g e N))
1195: (<n> <--> (k a z u))
1196: (<n> <--> (z i))
```

```
1197: (<n> <--> (y uu s ou))
1198: (<n> <--> (z i b u N))
1199: (<n> <--> (y uu b i N))
1200: (<n> <--> (r a i g e ts Uu))
1201: (<n> <--> (a N sh Ui N))
1202: (<n> <--> (k Ui b ou))
1203: (<n> <--> (s a N k ou))
1204: (<n> <--> (r i r e k Ui sy o))
1205: (<n> <--> (b o sy uu))
1206: (<n> <--> (h o N y a k Uu))
1207: (<n> <--> (sh Ui k a k Uu))
1208: (<n> <--> (ky uu k ei))
1209: (<n> <--> (b u))
1210: (<n> <--> (y ou k e N))
1211: (<n> <--> (sy a sh Ui N))
1212: (<n> <--> (y ou s Uu))
1213: (<n> <--> (sh Ui ts Uu m o N))
1214: (<n> <--> (y o r u))
1215: (<n> <--> (t o m a r i))
1216: (<n> <--> (h e y a))
1217: (<n> <--> (ts Uu i N))
1218: (<n> <--> (d ou sh Ui ts Uu))
1219: (<n> <--> (m u s Uu m e))
1220: (<n> <--> (k a N k ou))
1221: (<n> <--> (sh Ui n a i))
1222: (<n> <--> (ch Ui k a k Uu))
1223: (<n> <--> (d a b u r u))
1224: (<n> <--> (sh Ui N g u r u))
1225: (<n> <--> (m ei k Ui))
1226: (<n> <--> (r e N r a k Uu))
1227: (<n> <--> (k ou z a))
1228: (<n> <--> (y ou k ou))
1229: (<n> <--> (b a N g ou))
1230: (<n> <--> (m ei sy ou))
1231: (<n> <--> (a t e s a k Ui))
1232: (<n> <--> (k a i zy ou))
1233: (<n> <--> (k a i zy ou h Uu k Ui N))
1234: (<n> <--> (h Uu k Ui N))
1235: (<n> <--> (k o z i N t e k Ui))
1236: (<n> <--> (s aa b i s Uu))
1237: (<n> <--> (h a r a i m o d o sh Ui))
1238: (<n> <--> (sy o k Uu h Ui))
1239: (<n> <--> (t ou r o N))
1240: (<n> <--> (g e sy a))
1241: (<n> <--> (k o k Uu t e ts Uu))
1242: (<n> <--> (i Q sy o))
1243: (<n> <--> (t e h a i))
1244: (<n> <--> (b u N))
1245: (<n> <--> (p Uu N))
1246: (<n> <--> (d a i <number-word-k> k a i))
1247: (<n> <--> (g o h a N))
1248: (<n> <--> (h Ui r u))
1249: (<n> <--> (o k Uu r i))
1250: (<n> <--> (m a ts Uu z i ts Uu))
1251: (<n> <--> (u ts Uu sh Ui))
1252: (<n> <--> (g a k Uu s ei sy ou))
1253: (<n> <--> (g a k Uu w a r i))
1254: (<n> <--> (u e))
1255: (<n> <--> (g e N g o g a k Uu))
1256: (<n> <--> (k a N r e N))
1257: (<n> <--> (g e N zy ou))
1258: (<n> <--> (d e N w a))
1259: (<n> <--> (m o k Uu t e k Ui))
1260: (<n> <--> (sh Ui ts Uu r ei))
1261: (<n> <--> (g a k Uu cy ou))
```

```
1262: (<n> <--> (ky o k Uu cy ou))
1263: (<n> <--> (z i m u))
1264: (<n> <--> (m e Q s ee z i))
1265: (<n> <--> (h o N t ou))
1266: (<n> <--> (sy u sy ou))
1267: (<n> <--> (sy o n i ch Ui))
1268: (<n> <--> (k Ui ny uu))
1269: (<n> <--> (t e ts Uu z u k Ui))
1270: (<n> <--> (g o g o))
1271: (<n> <--> (k a i k a N))
1272: (<n> <--> (r i y ou))
1273: (<n> <--> (b a s Uu))
1274: (<n> <--> (ch Ui k a t e ts Uu))
1275: (<n> <--> (o s o r e))
1276: (<n> <--> (zy uu t a i))
1277: (<n> <--> (a s a))
1278: (<n> <--> (z i k a N))
1279: (<n> <--> (h ou h ou))
1280: (<n> <--> (t a k Uu sh ii))
1281: (<n> <--> (e k Ui))
1282: (<n> <--> (b a sy o))
1283: (<n> <--> (k a N ry ou))
1284: (<n> <--> (u k e ts Uu k e))
1285: (<n> <--> (y uu z i N))
1286: (<n> <--> (t o k o r o))
1287: (<n> <--> (k ou r i ts Uu))
1288: (<n> <--> (k o k Uu r i ts Uu))
1289: (<n> <--> (z i z e N))
1290: (<n> <--> (k a t a g a t a))
1291: (<n> <--> (d ou gy ou sy a))
1292: (<n> <--> (a i d a))
1293: (<n> <--> (k a N))
1294: (<n> <--> (i Q p a N))
1295: (<n> <--> (ts Uu aa))
1296: (<n> <--> (s a i t o sh ii i N g u))
1297: (<n> <--> (h a N))
1298: (<n> <--> (k a w a r i))
1299: (<n> <--> (t o k Ui))
1300: (<n> <--> (cy uu sy o k Uu))
1301: (<n> <--> (k o g i Q t e))
1302: (<n> <--> (k Uu r e z i Q t o k aa d o))
1303: (<n> <--> (k a n e))
1304: (<n> <--> (m ou sh Ui w a k e))
1305: (<n> <--> (ky a N s e r u))
1306: (<n> <--> (k Ui N g a k Uu))
1307: (<n> <--> (t o r i))
1308: (<n> <--> (<number-word> m a N e N))
1309: (<n> <--> (<number-word> m a N))
1310: (<n> <--> (<number-word> s e N e N))
1311: (<n> <--> (<number-word> m a N <number-word> s e N e N))
1312: (<n> <--> (sh Ui ry ou))
1313: (<n> <--> (s a i t ei))
1314: (<n> <--> (k Ui k a N))
1315: (<n> <--> (z e N))
1316: (<n> <--> (d ou z i))
1317: (<n> <--> (k Ui n e N))
1318: (<n> <--> (k ei sh Ui k Ui))
1319: (<n> <--> (w a k e))
1320: (<n> <--> (zy ou ky ou))
1321: (<n> <--> (sh Ui g o t o))
1322: (<n> <--> (b u N y a))
1323: (<n> <--> (k o k Uu n a i))
1324: (<n> <--> (k a i sy a))
1325: (<n> <--> (k a i h a ts Uu))
1326: (<n> <--> (k Ui))
```

1327: (<n> <--> (ts uu y a k Uu d e N w a))
1328: (<n> <--> (ts uu y a k Uu))
1329: (<n> <--> (z i d ou))
1330: (<n> <--> (z i d ou h o N y a k Uu))
1331: (<n> <--> (z i d ou h o N y a k Uu d e N w a))
1332: (<n> <--> (z i d ou h o N y a k Uu d e N w a k e N ky uu zy o))
1333: (<n> <--> (z i d ou h o N y a k Uu d e N w a k e N ky uu sy o))
1334: (<n> <--> (sy u d a i))
1335: (<n> <--> (k o my u n i k ee sy o N))
1336: (<n> <--> (h a i t e k Uu))
1337: (<n> <--> (sy u s a i))
1338: (<n> <--> (z i m u ky o k Uu))
1339: (<n> <--> (i r o i r o))
1340: (<n> <--> (a t o))
1341: (<n> <--> (ee zy e N t o))
1342: (<n> <--> (t o r a b e r u))
1343: (<n> <--> (n a k a))
1344: (<n> <--> (zy u))
1345: (<n> <--> (s Uu r a i d o))
1346: (<n> <--> (y o y a k Uu))
1347: (<n> <--> (k a k Uu z i))
1348: (<n> <--> (sy u k Uu h a k Uu))
1349: (<n> <--> (h o t e r u))
1350: (<n> <--> (b a a i))
1351: (<n> <--> (d o r u))
1352: (<n> <--> (i m a))
1353: (<n> <--> (ky ou zy u))
1354: (<n> <--> (z e N b u))
1355: (<n> <--> (sy o r u i))
1356: (<n> <--> (t a m e))
1357: (<n> <--> (s ei))
1358: (<n> <--> (y o t ei))
1359: (<n> <--> (k e N ky uu))
1360: (<n> <--> (k e N ky uu zy o))
1361: (<n> <--> (k e N ky uu sy o))
1362: (<n> <--> (k e N ky uu b u N y a))
1363: (<n> <--> (d a i g a k Uu))
1364: (<n> <--> (k a N g ei))
1365: (<n> <--> (h Uu z i N))
1366: (<n> <--> (d ou k ou))
1367: (<n> <--> (p Uu r o g u r a m u))
1368: (<n> <--> (m u ry ou))
1369: (<n> <--> (ts Uu m a))
1370: (<n> <--> (k a N b e N))
1371: (<n> <--> (s a i g o))
1372: (<n> <--> (r o N b u N))
1373: (<n> <--> (k o p ii))
1374: (<n> <--> (b e ts Uu))
1375: (<n> <--> (i k Uu ts Uu))
1376: (<n> <--> (k a i))
1377: (<n> <--> (b u N k a))
1378: (<n> <--> (t ou z i ts Uu))
1379: (<n> <--> (ry ou h ou))
1380: (<n> <--> (h Ui ts Uu y ou))
1381: (<n> <--> (d ou h uu))
1382: (<n> <--> (h a Q py ou))
1383: (<n> <--> (g o h a Q py ou))
1384: (<n> <--> (m ou sh Ui k o m i))
1385: (<n> <--> (o m ou sh Ui k o m i))
1386: (<n> <--> (m ou sh Ui k o m i y ou sh Ui))
1387: (<n> <--> (d a N t a i))
1388: (<n> <--> (w a r i b i k Ui))
1389: (<n> <--> (ry ou k Ui N))
1390: (<n> <--> (g a k Uu s ei))
1391: (<n> <--> (sh Ui N p a i))

```
1392: (<n> <--> (h a n a sh Ui))
1393: (<n> <--> (d ou z i ts uu y a k Uu))
1394: (<n> <--> (s a i))
1395: (<n> <--> (g e N g o))
1396: (<n> <--> (sy a))
1397: (<n> <--> (g a i k o k Uu))
1398: (<n> <--> (k o k Uu s a i))
1399: (<n> <--> (k o N k a i))
1400: (<n> <--> (z i sh Ui N))
1401: (<n> <--> (ei g o))
1402: (<n> <--> (n i h o N g o))
1403: (<n> <--> (n i Q p o N g o))
1404: (<n> <--> (k o t o b a))
1405: (<n> <--> (k o t o sh Ui))
1406: (<n> <--> (k Ui g e N))
1407: (<n> <--> (g i N k ou))
1408: (<n> <--> (h Ui y ou))
1409: (<n> <--> (n a i y ou))
1410: (<n> <--> (sh Ui m e k Ui r i))
1411: (<n> <--> (s a N k a))
1412: (<n> <--> (s a N k a h Ui y ou))
1413: (<n> <--> (s a N k a t ou r o k Uu))
1414: (<n> <--> (h ou))
1415: (<n> <--> (k a t a))
1416: (<n> <--> (sh Ui ky uu))
1417: (<n> <--> (m a ch Ui))
1418: (<n> <--> (m a ch Ui z i k a N))
1419: (<n> <--> (k Uu))
1420: (<n> <--> (zy uu sy o))
1421: (<n> <--> (zy uu sy o sh Ui m ei))
1422: (<n> <--> (g o zy uu sy o))
1423: (<n> <--> (n a m a e))
1424: (<n> <--> (o n a m a e))
1425: (<n> <--> (y ou sh Ui))
1426: (<n> <--> (cy ou k ou))
1427: (<n> <--> (t ou r o k Uu))
1428: (<n> <--> (t ou r o k Uu ry ou))
1429: (<n> <--> (k a i g i))
1430: (<n> <--> (k a i g i zy ou))
1431: (<n> <--> (k a i g i zy ou y u k Ui))
1432: (<n> <--> (o t a k Uu))
1433: (<n> <--> (a n a t a))
1434: (<n> <--> (d o n a t a))
1435: (<n> <--> (d a r e))
1436: (<n> <--> (m i n a))
1437: (<n> <--> (y o k ou sy uu d a i))
1438: (<n> <--> (y o k ou sy uu))
1439: (<n> <--> (ky ou t o))
1440: (<n> <--> (ky ou t o e k Ui))
1441: (<n> <--> (ky ou t o k o k Uu s a i k a i g i zy ou))
1442: (<n> <--> (zy u r i))
1443: (<n> <--> (zy u r i ts uu ch Ui))
1444: (<n> <--> (g e N g o k a i s e k Ui))
1445: (<n> <--> (k a i s e k Ui))
1446: (<n> <--> (k a n ou))
1447: (<n> <--> (sh Ui r a s e))
1448: (<n> <--> (o sh Ui r a s e))
1449: (<n> <--> (oo s a k a sh Ui))
1450: (<n> <--> (h Ui g a sh Ui k Uu))
1451: (<n> <--> (sh Ui r o m i))
1452: (<n> <--> (s o ch Ui r a))
1453: (<n> <--> (s e N ky uu hy a k Uu ky uu zy uu <number-word>))
1454: (<n> <--> (k Ui t a o o z i e k Ui))
1455: (<n> <--> (s ou))
1456: (<n> <--> (sy o t ei))
```

```
1457: (<n> <--> (sh Ui m ei))
1458: (<n> <--> (s o k o))
1459: (<n> <--> (i m i))
1460: (<n> <--> (ry a k Uu z u))
1461: (<n> <--> (o n e g a i))
```

A.2 Results

The recognition results for the first 100 test phrases are listed below. We explain the format of results by taking the first phrase as an example. "(001)" indicates the phrase number, "MAU_SB3_01" is the name of the speech data file, "296.0" and "1246.5" are the start and end positions in the file where the phrase utterance is located, and "|daiikkai|" shows the correct phrase. For each phrase, the top five recognition results are shown, in which a correctly recognized result is marked with '>' and a word boundary is marked with '–'. Each score represents negative log likelihood, thus smaller value is more probable.

```
(001) MAU_SB3_01    296.0  1246.5   |daiikkai|
 > 1: daiiQ-kai                          (score = 11.446128)
   2: daiiQ-kai-ni                       (score = 11.792510)
   3: dairoQ-kai                         (score = 11.816109)
   4: daizyuQ-kai                        (score = 11.955602)
   5: daiiQ-kai-o                        (score = 12.059597)

(002) MAU_SB3_01   1991.0  2734.0   |tsuuyaku|
 > 1: tsuuyaku                           (score = 11.979701)
   2: tsuuyaku-o                         (score = 12.630069)
   3: seiyaku                            (score = 12.777961)
   4: tsuuyaku-no                        (score = 12.782361)
   5: tsuuyaku-mo                        (score = 12.797860)

(003) MAU_SB3_01   3078.5  3694.0   |deNwa|
 > 1: deNwa                              (score = 11.467227)
   2: doNna                              (score = 11.824109)
   3: de-ru-ga                           (score = 11.846608)
   4: buN-ga                             (score = 11.848408)
   5: buN-wa                             (score = 11.852407)

(004) MAU_SB3_01   4316.0  4971.5   |kokusai|
 > 1: kokusai                            (score = 11.567922)
   2: kokusai-ni                         (score = 11.998000)
   3: kokusai-yori                       (score = 12.524974)
   4: kokusai-o                          (score = 12.592970)
   5: yosa-ni                            (score = 12.656467)

(005) MAU_SB3_01   5373.5  5949.0   |kaigini|
 > 1: kaigi-ni                           (score = 13.158442)
   2: kaigi                              (score = 13.288036)
   3: kai-ni                             (score = 13.301835)
   4: taiheN                             (score = 13.507725)
   5: daibu                              (score = 13.561222)
```

```
(006) MAU_SB3_01   6691.0  7391.5   |saNkano|
   1: saN-dano                          (score = 11.702315)
 > 2: saNka-no                          (score = 11.742713)
   3: saNka-mo                          (score = 11.796910)
   4: saN-ka-go                         (score = 11.962902)
   5: saNka-o                           (score = 11.975201)

(007) MAU_SB3_01   7808.5  8546.5   |tourokuo|
 > 1: touroku-o                         (score = 12.079896)
   2: touroku-mo                        (score = 12.267587)
   3: touroku-no                        (score = 12.297485)
   4: doukou                            (score = 12.530473)
   5: doukou-o                          (score = 12.570172)

(008) MAU_SB3_01   9048.5 10169.0   |gokibousareru|
 > 1: gokibou-sa-re-ru                  (score = 11.921004)
   2: gokibou-sa-re-re-ru               (score = 12.127094)
   3: gokibou-suru                      (score = 12.208390)
   4: gokibou-sa-se-ru                  (score = 12.399880)
   5: gokibou-sa-re-re-re-ru            (score = 12.471776)

(009) MAU_SB3_01  10641.0 11111.5   |katawa|
 > 1: kata-wa                           (score = 12.152292)
   2: kata                              (score = 12.516974)
   3: kata-da                           (score = 12.746063)
   4: ka-i-ta-ra                        (score = 12.747263)
   5: kata-ga                           (score = 12.763762)

(010) MAU_SB3_01  12363.5 13151.5   |shoteino|
 > 1: syotei-no                         (score = 11.853207)
   2: syotei-eno                        (score = 11.875606)
   3: syotei-mo                         (score = 11.939303)
   4: syuteN-o                          (score = 12.115994)
   5: syuteN-no                         (score = 12.129294)

(011) MAU_SB3_01  13646.0 14431.5   |moushikomi|
 > 1: moushikomi                        (score = 12.233588)
   2: moushikomi-ni                     (score = 12.639168)
   3: moushiko-mu                       (score = 12.745063)
   4: omoushikomi                       (score = 12.767962)
   5: moushikomi-o                      (score = 12.823959)

(012) MAU_SB3_01  14986.0 15639.0   |youshini|
 > 1: youshi-ni                         (score = 12.677966)
   2: youshi                            (score = 12.701065)
   3: youshiki                          (score = 12.806360)
   4: youshiki-ni                       (score = 12.920054)
   5: yoroshi-i                         (score = 13.048248)

(013) MAU_SB3_01  16441.0 17119.0   |juusho|
 > 1: zyuusyo                           (score = 11.653817)
   2: zyuusyo-o                         (score = 11.744113)
   3: zyuusyo-mo                        (score = 12.132993)
   4: zimusyo                           (score = 12.179791)
   5: zimusyo-o                         (score = 12.270087)

(014) MAU_SB3_01  17511.0 18251.5   |shimeito|
 > 1: shimei-to                         (score = 11.809710)
   2: shimei-ato                        (score = 12.051197)
   3: shimeki-ru-to                     (score = 12.249088)
   4: shimekiri-to                      (score = 12.318784)
   5: shire-ru-to                       (score = 12.445678)
```

```
(015) MAU_SB3_01  19068.5 19854.0   |happyou|
 > 1: haQpyou                            (score = 10.704465)
   2: haQpyou-o                          (score = 10.932153)
   3: haQpyou-mo                         (score = 11.292835)
   4: haQpyou-no                         (score = 11.417229)
   5: haQpyou-yo                         (score = 11.468327)

(016) MAU_SB3_01  20221.0 21066.5   |choukouno|
 > 1: cyoukou-no                         (score = 11.201640)
   2: cyoukou-mo                         (score = 11.239238)
   3: cyoukou-o                          (score = 11.565622)
   4: cyoukou                            (score = 11.614719)
   5: cyoukou-goro                       (score = 11.651117)

(017) MAU_SB3_01  21476.0 22104.0   |betsuo|
 > 1: betsu-o                            (score = 11.156342)
   2: betsu-mo                           (score = 11.458627)
   3: betsu-no                           (score = 11.516524)
   4: betsu-yo                           (score = 11.634318)
   5: de-sou                             (score = 11.739813)

(018) MAU_SB3_01  22456.0 23221.5   |meikishite|
 > 1: meiki-shi-te                       (score = 12.320184)
   2: meiki-shi-te-i-ku                  (score = 12.476576)
   3: meiki-shi-te-o-ku                  (score = 12.677466)
   4: meiki-shi-ta                       (score = 12.914854)
   5: meiki-shi-te-i-ru                  (score = 12.918254)

(019) MAU_SB3_01  24246.0 24879.0   |kokusai|
 > 1: kokusai                            (score = 11.702915)
   2: kokusai-ni                         (score = 12.164192)
   3: kokusai-o                          (score = 12.727064)
   4: kokusai-yori                       (score = 12.759362)
   5: kokusai-e                          (score = 12.772261)

(020) MAU_SB3_01  25243.5 25789.0   |kaigi|
 > 1: kaigi                              (score = 12.316984)
   2: kai-ni                             (score = 12.496175)
   3: kaigi-ni                           (score = 12.755062)
   4: kai                                (score = 12.816459)
   5: kai-yori                           (score = 12.924054)

(021) MAU_SB3_01  26206.0 27141.5   |jimukyokumade|
 > 1: zimukyoku-made                     (score = 12.691465)
   2: zimukyoku-mae                      (score = 12.715064)
   3: zimukyoku-dake                     (score = 12.943053)
   4: zimukyoku-nado                     (score = 12.979651)
   5: zimukyoku-da                       (score = 13.101245)

(022) MAU_SB3_01  27586.0 28346.5   |omoushikomi|
 > 1: omoushikomi                        (score = 12.652767)
   2: omoushikomi-ni                     (score = 12.898055)
   3: moushikomi                         (score = 12.946153)
   4: moushikomi-ni                      (score = 13.191340)
   5: kono-moushikomi                    (score = 13.198840)

(023) MAU_SB3_01  28776.0 29289.0   |kudasai|
 > 1: kudasai                            (score = 12.830658)
   2: kaisai                             (score = 13.444228)
   3: nasa-ru                            (score = 13.732413)
   4: kazu-ni                            (score = 13.833408)
   5: hasa-mu                            (score = 13.926204)
```

```
(024) MAU_SB3_02    288.5   634.0   |hai|
 > 1: hai                                (score = 11.685516)
   2: kai                                (score = 11.964302)
   3: huri                               (score = 12.132993)
   4: hi                                 (score = 12.273786)
   5: kurai                              (score = 12.341783)

(025) MAU_SB3_02   1363.5  2006.5   |kochirawa|
 > 1: kochira-wa                         (score = 12.843858)
   2: kokunai-wa                         (score = 13.238138)
   3: kaisya-wa                          (score = 13.275636)
   4: hoka-wa                            (score = 13.375231)
   5: kochira-niwa                       (score = 13.451727)

(026) MAU_SB3_02   2536.0  3459.0   |daiikkai|
 > 1: daiiQ-kai                          (score = 11.631318)
   2: daiiQ-kai-ni                       (score = 11.964002)
   3: dairoQ-kai                         (score = 12.005300)
   4: daizyuQ-kai                        (score = 12.013699)
   5: daihaQ-kai                         (score = 12.140293)

(027) MAU_SB3_02   4056.0  4771.5   |tsuuyaku|
 > 1: tsuuyaku                           (score = 12.190990)
   2: tsudu-ku                           (score = 12.686266)
   3: tsuuyaku-o                         (score = 12.920154)
   4: tsuuyaku-e                         (score = 13.037348)
   5: tsudu-ke-ru                        (score = 13.057947)

(028) MAU_SB3_02   5106.0  5701.5   |deNwa|
 > 1: deNwa                              (score = 11.752912)
   2: buN-wa                             (score = 11.943303)
   3: buN-da                             (score = 12.035498)
   4: buN-ga                             (score = 12.052997)
   5: de-ru-ga                           (score = 12.060897)

(029) MAU_SB3_02   6183.5  6834.0   |kokusai|
 > 1: kokusai                            (score = 12.110794)
   2: kokusai-ni                         (score = 12.595970)
   3: kokusai-o                          (score = 12.965252)
   4: kokusai-e                          (score = 13.076846)
   5: kokusai-yori                       (score = 13.114544)

(030) MAU_SB3_02   7283.5  7789.0   |kaigi|
 > 1: kaigi                              (score = 12.833358)
   2: kai-ni                             (score = 13.023449)
   3: kai                                (score = 13.089046)
   4: kaigi-ni                           (score = 13.228639)
   5: hai                                (score = 13.294835)

(031) MAU_SB3_02   8228.5  9246.5   |jimukyokudesu|
 > 1: zimukyoku-desu                     (score = 12.916654)
   2: zyuukyu-ka-desu                    (score = 13.329734)
   3: zyuuku-ka-desu                     (score = 13.414129)
   4: zimukyoku-sa                       (score = 13.466527)
   5: zimukyoku-koso                     (score = 13.557722)

(032) MAU_SB3_03    291.0   931.5   |moshimoshi|
 > 1: moshimoshi                         (score = 12.389981)
   2: go-syuu-nochi                      (score = 13.681716)
   3: ni-syuu-nochi                      (score = 13.930803)
   4: mochii-ru-rashi-i                  (score = 13.935103)
   5: bosyuu-ni                          (score = 14.080096)
```

```
(033) MAU_SB3_03   1763.5  2494.0   |tsuuyaku|
 > 1: tsuuyaku                          (score = 11.840708)
   2: tsuuyaku-o                        (score = 12.544773)
   3: tsudu-ku                          (score = 12.545373)
   4: seiyaku                           (score = 12.735163)
   5: tsure-na-ku-na-ru                 (score = 12.748363)

(034) MAU_SB3_03   3068.5  3689.0   |deNwa|
 > 1: deNwa                             (score = 11.273636)
   2: buN-wa                            (score = 11.466227)
   3: beNri-wa                          (score = 11.588621)
   4: buN-ga                            (score = 11.714914)
   5: buN-da                            (score = 11.718814)

(035) MAU_SB3_03   4348.5  4961.5   |kokusai|
 > 1: kokusai                           (score = 11.894205)
   2: kokusai-ni                        (score = 12.336983)
   3: kokusai-o                         (score = 12.866957)
   4: kokusai-yori                      (score = 12.958052)
   5: kokusai-e                         (score = 12.995350)

(036) MAU_SB3_03   5401.0  6176.5   |kaigieno|
 > 1: kaigi-eno                         (score = 12.436578)
   2: kaigi-demo                        (score = 12.552072)
   3: kaigi-mo                          (score = 12.758462)
   4: kaigi-no                          (score = 12.773761)
   5: kaigi-yo                          (score = 12.849058)

(037) MAU_SB3_03   6731.0  7456.5   |saNkao|
 > 1: saNka-o                           (score = 12.131493)
   2: saNkou                            (score = 12.290685)
   3: saNkou-o                          (score = 12.461377)
   4: saNka-mo                          (score = 12.464777)
   5: saN-ka-go                         (score = 12.474076)

(038) MAU_SB3_03   7976.0  9181.5   |moushikomitainodesu|
 > 1: moushiko-mi-ta-inodesu            (score = 12.396480)
   2: moushiko-mi-ta-i-N-desu           (score = 12.433578)
   3: moushiko-me-ta-inodesu            (score = 12.470177)
   4: moushiko-me-ta-i-N-desu           (score = 12.507375)
   5: moushiko-me-ta-N-desu             (score = 12.566672)

(039) MAU_SB3_03   9921.0 10499.0   |keredomo|
 > 1: keredomo                          (score = 12.230888)
   2: teedo-o                           (score = 12.452277)
   3: teido-o                           (score = 12.452277)
   4: teido                             (score = 12.459477)
   5: teedo                             (score = 12.459477)

(040) MAU_SB3_03  11758.5 12316.5   |dono|
 > 1: dono                              (score = 11.093045)
   2: go-no                             (score = 11.279136)
   3: dou                               (score = 11.310334)
   4: go-mo                             (score = 11.347633)
   5: doru-o                            (score = 11.398030)

(041) MAU_SB3_03  12881.0 13484.0   |youna|
 > 1: youna                             (score = 12.188091)
   2: youi-da                           (score = 12.462077)
   3: i-u-you-na                        (score = 12.480076)
   4: youi-ga                           (score = 12.535873)
   5: i-u-you-da                        (score = 12.575471)
```

```
(042) MAU_SB3_03  14061.0 14824.0   |tetsuzukio|
   1: tetsuzuki-yo                        (score = 12.274686)
 > 2: tetsuzuki-o                         (score = 12.592070)
   3: tetsuzuki-mo                        (score = 12.640568)
   4: tetsuzuki-no                        (score = 12.774361)
   5: tetsuzuki-goro                      (score = 12.801960)

(043) MAU_SB3_03  15413.5 15996.5   |sureba|
 > 1: sure-ba                             (score = 12.034298)
   2: sore-da                             (score = 12.235488)
   3: su-me-ba                            (score = 12.366682)
   4: tsu-ke-ba                           (score = 12.375281)
   5: tsuku-re-ba                         (score = 12.391480)

(044) MAU_SB3_03  16596.0 17756.5   |yoroshiinodeshouka|
   1: yoroshi-iNdesyouka                  (score = 12.807460)
 > 2: yoroshiinodesyouka                  (score = 12.854757)
   3: yu-u-rashi-iNdesyouka               (score = 12.883156)
   4: youi-shi-na-idesyouka               (score = 12.978851)
   5: i-u-rashi-iNdesyouka                (score = 13.007250)

(045) MAU_SB3_04    296.0  1011.5   |tsuuyaku|
 > 1: tsuuyaku                            (score = 11.752212)
   2: tsuuyaku-o                          (score = 12.420879)
   3: tsuuyaku-e                          (score = 12.504475)
   4: tsuuyaku-no                         (score = 12.578771)
   5: tsuuyaku-mo                         (score = 12.585471)

(046) MAU_SB3_04   1451.0  2179.0   |deNwano|
 > 1: deNwa-no                            (score = 11.515924)
   2: deNwa-mo                            (score = 11.516224)
   3: buN-nado                            (score = 11.741713)
   4: deNwa-o                             (score = 11.832608)
   5: deNwa-goro                          (score = 11.850207)

(047) MAU_SB3_04   2561.0  3239.0   |kokusai|
 > 1: kokusai                             (score = 11.665617)
   2: kokusai-ni                          (score = 12.038598)
   3: kono-sai                            (score = 12.210689)
   4: kokusai-yori                        (score = 12.568772)
   5: kokusai-o                           (score = 12.577571)

(048) MAU_SB3_04   3561.0  4159.0   |kaigini|
   1: kai-ni                              (score = 13.343533)
 > 2: kaigi-ni                            (score = 13.366632)
   3: kaigi                               (score = 13.443728)
   4: daibu                               (score = 13.548823)
   5: kikai-ni                            (score = 13.647618)

(049) MAU_SB3_04   4591.0  5481.5   |saNkasuru|
 > 1: saNka-suru                          (score = 11.573721)
   2: saNka-sa-re-ru                      (score = 12.003400)
   3: saN-gatsu                           (score = 12.079096)
   4: saN-nitaisuru                       (score = 12.120194)
   5: saNka-sure                          (score = 12.140993)

(050) MAU_SB3_04   5738.5  6359.0   |tameniwa|
 > 1: tame-niwa                           (score = 12.351282)
   2: tame-dewa                           (score = 12.605770)
   3: kane-niwa                           (score = 12.672066)
   4: kaigi-niwa                          (score = 12.802860)
   5: kagi-ru-niwa                        (score = 12.804960)
```

```
(051) MAU_SB3_04   6906.0  7694.0   |shoteino|
 > 1: syotei-no                          (score = 11.440528)
   2: syotei-eno                         (score = 11.481226)
   3: syotei-mo                          (score = 11.514224)
   4: syuteN-o                           (score = 11.806510)
   5: syuteN-no                          (score = 11.874106)

(052) MAU_SB3_04   8148.5  8894.0   |moushikomi|
 > 1: moushikomi                         (score = 12.333383)
   2: moushiko-mu                        (score = 12.576171)
   3: moushikomi-ni                      (score = 12.756662)
   4: moushiko-me-ru                     (score = 12.809460)
   5: omoushikomi                        (score = 12.848558)

(053) MAU_SB3_04   9268.5  9939.0   |youshio|
   1: youshi-yo                          (score = 11.734713)
 > 2: youshi-o                           (score = 11.754412)
   3: youshi-mo                          (score = 11.762212)
   4: youshi-no                          (score = 11.790610)
   5: youshiki-o                         (score = 11.987601)

(054) MAU_SB3_04  10296.0 10999.0   |mochiite|
 > 1: mochii-te                          (score = 12.273386)
   2: mochii-te-i-ku                     (score = 12.588571)
   3: mochii-re-te                       (score = 12.737163)
   4: mochii-te-i-ru                     (score = 12.743363)
   5: mochii-te-o-ku                     (score = 12.829059)

(055) MAU_SB3_04  11476.0 12154.0   |saNka|
 > 1: saNka                              (score = 11.608320)
   2: saN-ka                             (score = 11.608320)
   3: saN-kara                           (score = 11.934003)
   4: saNka-ga                           (score = 12.105195)
   5: saNka-wa                           (score = 12.134093)

(056) MAU_SB3_04  12576.0 13406.5   |tourokusuru|
 > 1: touroku-suru                       (score = 11.394930)
   2: too-ku-suru                        (score = 11.562722)
   3: touroku-sure                       (score = 12.092395)
   4: touroku-sa-re-ru                   (score = 12.188191)
   5: koutsuu                            (score = 12.218189)

(057) MAU_SB3_04  13808.5 14376.5   |kotoga|
 > 1: koto-ga                            (score = 11.905305)
   2: kotoba                             (score = 12.166192)
   3: otona                              (score = 12.286586)
   4: koto-ya                            (score = 12.332183)
   5: koto-da                            (score = 12.356782)

(058) MAU_SB3_04  14646.0 15544.0   |hitsuyoudesu|
 > 1: hitsuyou-desu                      (score = 12.575171)
   2: shiryou-desu                       (score = 12.774161)
   3: shi-ru-you-desu                    (score = 12.886356)
   4: ki-ku-you-desu                     (score = 12.915854)
   5: shi-ta-you-desu                    (score = 12.940653)

(059) MAU_SB3_05    291.0   974.0   |kaigini|
 > 1: kaigi-ni                           (score = 12.630568)
   2: kai-ni                             (score = 12.944453)
   3: kaN-ni                             (score = 12.955852)
   4: kaigai-ni                          (score = 13.022549)
   5: kaNreN-ni                          (score = 13.138243)
```

```
(060) MAU_SB3_05   1166.0  2569.0   |happyousurunodewa|
 > 1: haQpyou-suru-nodewa                (score = 11.636918)
   2: haQpyou-sa-re-ru-nodewa            (score = 12.195290)
   3: haQpyou-suru-niwa                  (score = 12.274786)
   4: haQpyou-shi-te-i-ru-nodewa         (score = 12.352482)
   5: hara-Q-te-i-ma-su-nodewa           (score = 12.399380)

(061) MAU_SB3_05   2861.0  3456.5   |nakute|
 > 1: na-ku-te                           (score = 12.069797)
   2: i-na-ku-te                         (score = 12.196490)
   3: na-ku-shi-te                       (score = 12.661667)
   4: i-na-ku-shi-te                     (score = 12.788461)
   5: na-Q-te                            (score = 12.818259)

(062) MAU_SB3_05   3826.0  5239.0   |choukousurudakedato|
 > 1: cyoukou-suru-dakedato              (score = 11.871206)
   2: cyoukou-sa-re-ru-dakedato          (score = 12.557772)
   3: cyoukou-sa-re-na-ku-na-to          (score = 12.562172)
   4: cyoukou-suru-rashi-ku-na-to        (score = 12.609070)
   5: cyoukou-shi-te-i-ru-dakedato       (score = 12.687166)

(063) MAU_SB3_05   5773.5  6459.0   |hiyouwa|
   1: shiryou-wa                         (score = 11.771111)
 > 2: hiyou-wa                           (score = 11.815809)
   3: syoyou-wa                          (score = 12.137593)
   4: shiryou-da                         (score = 12.153192)
   5: hiyou-da                           (score = 12.198390)

(064) MAU_SB3_05   6868.5  7414.0   |ikura|
 > 1: ikura                              (score = 12.410579)
   2: i-ku-ga                            (score = 12.762462)
   3: iki-da                             (score = 12.820359)
   4: i-ku-ka                            (score = 12.896455)
   5: roku-da                            (score = 12.926754)

(065) MAU_SB3_05   7661.0  8461.5   |kakarimasuka|
 > 1: kaka-ri-ma-su-ka                   (score = 12.485876)
   2: kaka-re-ma-su-ka                   (score = 12.628469)
   3: ka-ka-re-ma-su-ka                  (score = 12.628469)
   4: ki-ka-re-ma-su-ka                  (score = 12.767862)
   5: kaka-ri-ma-su-ga                   (score = 12.864057)

(066) MAU_SB3_06    288.5  1234.0   |gohappyouo|
   1: gohaQpyou                          (score = 11.072246)
 > 2: gohaQpyou-o                        (score = 11.322334)
   3: gohaQpyou-mo                       (score = 11.636718)
   4: gohaQpyou-yo                       (score = 11.738613)
   5: gohaQpyou-no                       (score = 11.776011)

(067) MAU_SB3_06   1563.5  2466.5   |kibousareru|
 > 1: kibou-sa-re-ru                     (score = 11.651117)
   2: kibou-sa-re-re-ru                  (score = 11.859707)
   3: kibou-sa-se-ru                     (score = 12.058397)
   4: kibou-suru                         (score = 12.070197)
   5: kibou-sa-re-re-re-ru               (score = 12.141793)

(068) MAU_SB3_06   2706.0  3486.5   |baainiwa|
 > 1: baai-niwa                          (score = 12.489376)
   2: baai-dewa                          (score = 12.707165)
   3: guai-niwa                          (score = 12.752262)
   4: guai-dewa                          (score = 12.969752)
   5: baai-ga                            (score = 13.059747)
```

```
(069) MAU_SB3_06   4498.5  5581.5   |yokoushuudai|
 > 1: yokousyuudai                        (score = 11.775711)
   2: yokousyuudai-ni                     (score = 12.016299)
   3: yokousyuu-gurai                     (score = 12.117094)
   4: yokousyuu-kurai                     (score = 12.187791)
   5: yokousyuudai-yori                   (score = 12.307785)

(070) MAU_SB3_06   6181.0  7119.0   |tourokuryouo|
   1: tourokuryou                         (score = 12.147993)
 > 2: tourokuryou-o                       (score = 12.301785)
   3: tourokuryou-mo                      (score = 12.551772)
   4: tourokuryou-no                      (score = 12.674766)
   5: tourokuryou-yo                      (score = 12.702265)

(071) MAU_SB3_06   7448.5  8194.0   |fukumeta|
 > 1: hukume-ta                           (score = 12.290585)
   2: huku-me-ta                          (score = 12.290585)
   3: hukume-ta-ra                        (score = 12.435378)
   4: huku-me-ta-ra                       (score = 12.435378)
   5: hukume-re-ta                        (score = 12.557272)

(072) MAU_SB3_06   8593.5  9236.5   |saNka|
 > 1: saNka                               (score = 11.758012)
   2: saN-ka                              (score = 11.758012)
   3: saN-kara                            (score = 12.117094)
   4: saNka-ga                            (score = 12.259587)
   5: saigo-ka                            (score = 12.311784)

(073) MAU_SB3_06   9601.0 10231.5   |hiyouwa|
 > 1: hiyou-wa                            (score = 12.145593)
   2: hiyou-da                            (score = 12.220189)
   3: shiryou-wa                          (score = 12.272386)
   4: shiryou-da                          (score = 12.344783)
   5: hiyou-ga                            (score = 12.502075)

(074) MAU_SB3_06  11061.0 12264.0   |yoNmaNeNdesu|
 > 1: yoN-maNeN-desu                      (score = 11.582321)
   2: yoN-maN-naNdesu                     (score = 11.898705)
   3: yo-ma-re-ru-N-desu                  (score = 11.938703)
   4: yo-ma-re-re-ru-N-desu               (score = 11.963202)
   5: yomiage-ru-N-desu                   (score = 11.980001)

(075) MAU_SB3_07    296.0  1361.5   |choukounomino|
 > 1: cyoukou-nomino                      (score = 11.893405)
   2: zyouhou-nomino                      (score = 12.410479)
   3: cyoukou-madeno                      (score = 12.500375)
   4: kyouto-nomino                       (score = 12.562172)
   5: cyoukou-nadono                      (score = 12.577171)

(076) MAU_SB3_07   1736.0  2459.0   |baaiwa|
 > 1: baai-wa                             (score = 11.690315)
   2: baai-da                             (score = 11.952802)
   3: baai-niwa                           (score = 11.990700)
   4: baai-ga                             (score = 12.037898)
   5: baai-ya                             (score = 12.166992)

(077) MAU_SB3_07   3001.0  3801.5   |toujitsuno|
 > 1: touzitsu-no                         (score = 11.309335)
   2: touzitsu-mo                         (score = 11.348833)
   3: touzitsu-o                          (score = 11.514024)
   4: touzitsu-eno                        (score = 11.782011)
   5: touzitsu-goro                       (score = 11.789911)
```

```
(078) MAU_SB3_07   4123.5  4934.0   |uketsukemo|
 > 1: uketsuke-mo                        (score = 12.271986)
   2: uketsuke-no                        (score = 12.330683)
   3: uketsuke-eno                       (score = 12.403080)
   4: betsu-eno                          (score = 12.693665)
   5: uketsuke-o                         (score = 12.737863)

(079) MAU_SB3_07   5163.5  5816.5   |kanoude|
 > 1: kanou-de                           (score = 11.573421)
   2: kanou-ne                           (score = 12.078696)
   3: kinou-de                           (score = 12.110694)
   4: kibou-de                           (score = 12.130094)
   5: kaNkou-de                          (score = 12.130293)

(080) MAU_SB3_07   6891.0  8026.5   |yokoushuudaio|
   1: yokousyuudai-yo                    (score = 12.005400)
 > 2: yokousyuudai-o                     (score = 12.185991)
   3: yokousyuudai-mo                    (score = 12.224689)
   4: yokousyuudai-no                    (score = 12.284786)
   5: yokousyuudai-goro                  (score = 12.398280)

(081) MAU_SB3_07   8353.5  9084.0   |fukumeta|
 > 1: hukume-ta                          (score = 12.169692)
   2: huku-me-ta                         (score = 12.169692)
   3: hukume-ta-ra                       (score = 12.400580)
   4: huku-me-ta-ra                      (score = 12.400580)
   5: hukume-re-ta                       (score = 12.455177)

(082) MAU_SB3_07   9393.5 10039.0   |hiyouwa|
 > 1: hiyou-wa                           (score = 12.191190)
   2: hiyou-da                           (score = 12.270886)
   3: shiryou-wa                         (score = 12.327684)
   4: shiryou-da                         (score = 12.406480)
   5: i-ku-you-da                        (score = 12.455977)

(083) MAU_SB3_07  10556.0 11364.0   |saNmaN|
 > 1: saN-maN                            (score = 12.181691)
   2: sono-baN                           (score = 12.403280)
   3: saN-maNeN                          (score = 12.548773)
   4: saN-ka-mimaN                       (score = 12.584371)
   5: sono-raineN                        (score = 12.865457)

(084) MAU_SB3_07  11573.5 12469.0   |goseNeN|
 > 1: go-seNeN                           (score = 11.540623)
   2: go-seNeN-o                         (score = 12.195190)
   3: go-seNeN-mo                        (score = 12.319484)
   4: go-seNeN-no                        (score = 12.359782)
   5: go-seNeN-ni                        (score = 12.364082)

(085) MAU_SB3_07  12731.0 13484.0   |kakarimasu|
 > 1: kaka-ri-ma-su                      (score = 12.289186)
   2: kaka-re-ma-su                      (score = 12.593270)
   3: ka-ka-re-ma-su                     (score = 12.593270)
   4: ka-ki-ma-su                        (score = 12.594270)
   5: kakari-desu                        (score = 12.721664)

(086) MAU_SB3_08    278.5   914.0   |saNka|
 > 1: saNka                              (score = 11.402030)
   2: saN-ka                             (score = 11.402030)
   3: saN-kara                           (score = 11.951502)
   4: zikaN-ka                           (score = 12.076296)
   5: saNka-wa                           (score = 12.105095)
```

```
(087) MAU_SB3_08   1311.0  2049.0   |tourokuno|
   1: touroku-mo                          (score = 11.619819)
 > 2: touroku-no                          (score = 11.635418)
   3: touroku-o                           (score = 11.812409)
   4: touroku-goro                        (score = 12.013499)
   5: touroku-hodo                        (score = 12.093395)

(088) MAU_SB3_08   2303.5  3056.5   |moushikomi|
 > 1: moushikomi                          (score = 12.164492)
   2: moushikomi-ni                       (score = 12.506775)
   3: moushiko-mu                         (score = 12.576771)
   4: omoushikomi                         (score = 12.680566)
   5: moushikomi-o                        (score = 12.799460)

(089) MAU_SB3_08   3258.5  3904.0   |youshiwa|
 > 1: youshi-wa                           (score = 11.941903)
   2: youshi-da                           (score = 12.154692)
   3: youshi-ga                           (score = 12.232088)
   4: youshiki-wa                         (score = 12.379381)
   5: youshiki-da                         (score = 12.392380)

(090) MAU_SB3_08   4463.5  5024.0   |dono|
 > 1: dono                                (score = 10.726964)
   2: dou                                 (score = 10.899755)
   3: go-no                               (score = 11.013349)
   4: doumo                               (score = 11.080746)
   5: go-mo                               (score = 11.094645)

(091) MAU_SB3_08   5246.0  5859.0   |youni|
 > 1: youni                               (score = 11.849708)
   2: youi-ni                             (score = 11.971901)
   3: yoru-ni                             (score = 12.226889)
   4: hiyou-ni                            (score = 12.277586)
   5: yoN-ni                              (score = 12.335283)

(092) MAU_SB3_08   6066.0  6576.5   |shite|
 > 1: shi-te                              (score = 11.641418)
   2: shi-e                               (score = 11.818709)
   3: shi-re                              (score = 11.944303)
   4: shita-e                             (score = 12.024899)
   5: shi-te-i-ru                         (score = 12.064097)

(093) MAU_SB3_08   6958.5  7419.0   |teni|
 > 1: te-ni                               (score = 11.376631)
   2: ke-ni                               (score = 11.652017)
   3: ue-ni                               (score = 11.763312)
   4: to-ni                               (score = 12.135393)
   5: keN-ni                              (score = 12.165092)

(094) MAU_SB3_08   7706.0  8394.0   |irereba|
 > 1: ire-re-ba                           (score = 12.239988)
   2: i-re-re-ba                          (score = 12.239988)
   3: ki-re-re-ba                         (score = 12.303085)
   4: i-e-re-ba                           (score = 12.355182)
   5: ie-re-ba                            (score = 12.355182)

(095) MAU_SB3_08   8686.0  9869.0   |yoroshiinodeshouka|
   1: yoroshi-iNdesyouka                  (score = 13.010050)
 > 2: yoroshiinodesyouka                  (score = 13.075146)
   3: youi-shi-na-idesyouka               (score = 13.250437)
   4: yu-u-rashi-iNdesyouka               (score = 13.287136)
   5: yoroshi-idesyouka                   (score = 13.290535)
```

```
(096) MAU_SB3_09    281.0  1104.0   |onamaeto|
 > 1: onamae-to                          (score = 11.619119)
   2: onamae-ato                         (score = 11.884106)
   3: namae-to                           (score = 11.915904)
   4: namae-ato                          (score = 12.180991)
   5: mawa-re-ru-to                      (score = 12.212389)

(097) MAU_SB3_09   1228.5  2094.0   |gojuushoo|
 > 1: gozyuusyo-o                        (score = 12.434978)
   2: gozyuusyo                          (score = 12.482576)
   3: gozyuusyo-mo                       (score = 12.737163)
   4: gozyuusyo-no                       (score = 12.818959)
   5: gozyuusyo-yo                       (score = 12.893455)

(098) MAU_SB3_09   2318.5  3029.0   |oshirase|
 > 1: oshirase                           (score = 12.005400)
   2: oshirase-e                         (score = 12.121894)
   3: oshirase-ne                        (score = 12.471576)
   4: oshie-ma-sure                      (score = 12.615469)
   5: oshirase-de                        (score = 12.633068)

(099) MAU_SB3_09   3261.0  4014.0   |itadakereba|
 > 1: itadake-re-ba                      (score = 13.332633)
   2: itada-ke-re-ba                     (score = 13.332633)
   3: yu-i-ta-kereba                     (score = 13.400130)
   4: i-ta-kereba                        (score = 13.425229)
   5: itada-ke-ba                        (score = 13.451127)

(100) MAU_SB3_09   4608.5  5236.5   |kokusai|
 > 1: kokusai                            (score = 11.846108)
   2: kokusai-ni                         (score = 12.389781)
   3: kokusai-o                          (score = 12.706265)
   4: kokusai-e                          (score = 12.846158)
   5: kokusai-yori                       (score = 12.939753)
```

Appendix B

Sentence Recognition Results

This appendix enumerates the sentence recognition results, using the two-level LR parsing described in Chapter 5. Two grammars, the intra- and inter-phrase grammars, were used for sentence recognition. These grammars are reported in [Hosaka 91].

```
(001) MAU_MA2_01  |moshimoshi|
 > 1: moshimoshi
   2: mochii-ru-rashii
   3: mo-chi-masyo-u
   4: mochii-masyo-u
   5: ma-chi-masyo-u

(002) MAU_MA2_02  |sochirawa kaigijimukyokudesuka|
 > 1: sochira-wa-kaigizimukyoku-desu-ka
   2: sochira-wa-kaigizimukyoku-desu-ga
   3: sochira-wa-kaigizimukyoku-deshi-ta
   4: sochira-ni-wa-kaigizimukyoku-desu-ka
   5: sochira-ga-kaigizimukyoku-desu-ka

(003) MAU_MA2_03  |hai|
 > 1: hai

(004) MAU_MA2_04  |soudesu|
 > 1: sou-desu
   2: i-u-soudesu
   3: i-i-soudesu
   4: souhu-desu

(005) MAU_MA2_05  |kaigini moushikomitainodesuga|
   1: kaigi-ni-moushiko-mi-tai-N-desu-ga
 > 2: kaigi-ni-moushiko-mi-tai-no-desu-ga
   3: kaigi-ni-moushiko-mi-tai-N-desu-ka
   4: kaigi-ni-moushiage-tai-N-desu-ga
   5: kaigi-ni-moushiko-mi-tai-no-desu-ka

(006) MAU_MA2_06  |tourokuyoushiwa sudeni omochideshouka|
 > 1: tourokuyoushi-wa-sudeni-omochi-desyo-u-ka
   2: tourokuyoushi-wa-sudeni-omachi-desyo-u-ka
   3: tourokuyoushi-ya-sudeni-omochi-desyo-u-ka
   4: tourokuyoushi-ya-sudeni-omachi-desyo-u-ka
   5: tourokuyoushi-e-wa-sudeni-omochi-desyo-u-ka
```

```
(007) MAU_MA2_07  |iie|
 > 1: iie

(008) MAU_MA2_08  |madadesu|
 > 1: mada-desu
   2: ima-desu
   3: n-ai-desu
   4: wadai-desu
   5: baai-desu

(009) MAU_MA2_09  |wakarimashita|
 > 1: waka-ri-mashi-ta
   2: waka-ru-rashii-ga
   3: waka-ri-mashi-te
   4: waka-ri-masyo-u-ga
   5: waka-ri-mashi-ta-ga

(010) MAU_MA2_10  |soredewa tourokuyoushio ookuriitashimasu|
   1: soredewa-tourokuyoushi-mo-ookuri-ita-shi-masu
   2: sore-de-wa-tourokuyoushi-mo-ookuri-ita-shi-masu
 > 3: soredewa-tourokuyoushi-o-ookuri-ita-shi-masu
   4: sore-de-wa-tourokuyoushi-o-ookuri-ita-shi-masu
   5: soredewa-tourokuyoushi-e-mo-ookuri-ita-shi-masu

(011) MAU_MA2_11  |gojuushoto onamaeo onegaishimasu|
 > 1: gozyuusyo-to-onamae-o-onegai-shi-masu
   2: gozyuusyo-to-o-onamae-o-onegai-shi-masu
   3: gozyuusyo-to-onamae-mo-onegai-shi-masu
   4: gozyuusyo-to-o-onamae-mo-onegai-shi-masu
   5: gozyuusyo-to-onamae-e-mo-onegai-shi-masu

(012) MAU_MA2_12  |juushowa oosakashi kitaku chayamachi nijuusaNdesu|
 > 1: zyuusyo-wa-oosaka-shi-kita-ku-cyayamachi-ni-zyuu-saN-desu
   2: zyuusyo-ga-oosaka-shi-kita-ku-cyayamachi-ni-zyuu-saN-desu
   3: zyuusyo-wa-oosaka-shi-kita-ku-cyayamachi-go-zyuu-saN-desu
   4: gozyuusyo-wa-oosaka-shi-kita-ku-cyayamachi-ni-zyuu-saN-desu
   5: zyuusyo-nara-oosaka-shi-kita-ku-cyayamachi-ni-zyuu-saN-desu

(013) MAU_MA2_13  |namaewa suzukimayumidesu|
 > 1: namae-wa-suzuki-mayumi-desu
   2: namae-wa-suzuki-na-N-desu
   3: namae-wa-suzuki-mayumi-nomi-desu
   4: namae-ni-wa-suzuki-mayumi-desu
   5: namae-wa-suzuki-mayumi-desu-yo

(014) MAU_MA2_14  |wakarimashita|
 > 1: waka-ri-mashi-ta
   2: kaka-ri-mashi-ta
   3: na-ri-mashi-ta
   4: waka-ru-rashii-ga
   5: waka-ru-rashii-ka

(015) MAU_MA2_15  |tourokuyoushiwa shikyuu okuraseteitadakimasu|
 > 1: tourokuyoushi-wa-shikyuu-oku-ra-se-te-itada-ki-masu
   2: tourokuyoushi-ya-shikyuu-oku-ra-se-te-itada-ki-masu
   3: tourokuyoushi-ga-shikyuu-oku-ra-se-te-itada-ki-masu
   4: tourokuyoushi-e-wa-shikyuu-oku-ra-se-te-itada-ki-masu
   5: tourokuyoushi-wa-shikyuu-oku-ra-se-te-itadake-masu

(016) MAU_MA2_16  |wakaranai teNga gozaimashitara itsudemo okikikudasai|
 > 1: waka-ra-nai-teN-ga-gozaima-shi-tara-itsu-demo-okiki-kudasai
   2: waka-ra-nai-teN-ga-gozaima-shi-tara-itsu-demo-okiki-nasai
   3: waka-ra-nai-teN-ga-gozaima-shi-tara-itsu-e-mo-okiki-kudasai
   4: waka-ra-nai-teN-ga-gozaima-shi-tara-itsu-demo-ookuri-kudasai
```

```
(017) MAU_MA2_17  |arigatougozaimasu|
 > 1: arigat-ou-gozaima-su
   2: arigat-ou-gozaima-su-yo
   3: arigat-ou-gozaima-su-ne
   4: arigat-ou-gozaima-su-ga
   5: arigat-ou-gozaima-se-N

(018) MAU_MA2_18  |soredewa shitsureishimasu|
 > 1: soredewa-shitsurei-shi-masu
   2: sore-de-wa-shitsurei-shi-masu
   3: sore-wa-shitsurei-shi-masu
   4: sore-e-wa-shitsurei-shi-masu
   5: soko-de-wa-shitsurei-shi-masu

(019) MAU_MA2_19  |doumo shitsureiitashimasu|
 > 1: doumo-shitsurei-ita-shi-masu
   2: dou-ni-mo-shitsurei-ita-shi-masu
   3: gogo-shitsurei-ita-shi-masu
   4: gogo-o-shitsurei-ita-shi-masu
   5: dou-shitsurei-ita-shi-masu

(020) MAU_MB2_01  |moshimoshi|
 > 1: moshimoshi
   2: mochii-ru-rashii
   3: ita-shi-mashi-te
   4: mo-chi-masyo-u

(021) MAU_MB2_02  |kochirawa kaigijimukyokudesu|
 > 1: kochira-wa-kaigizimukyoku-desu
   2: kochira-wa-kaigizimukyoku-nomi-desu
   3: kochira-wa-kaigizimukyoku-kurai-desu
   4: kochira-wa-kaigizimukyoku-desu-yo
   5: kochira-wa-kaigizimukyoku-gurai-desu

(022) MAU_MB2_03  |kaigini saNkashitainodesuga|
 > 1: kaigi-ni-saNka-shi-tai-no-desu-ga
   2: kaigi-ni-saNka-shi-tai-N-desu-ga
   3: kaigi-ni-saNka-shi-tai-no-desu-ka
   4: kaigi-ni-saNka-shi-tai-N-desu-ka
   5: kaigi-ni-saNka-shi-nai-no-desu-ga

(023) MAU_MB2_04  |dousureba yoroshiidesuka|
 > 1: dousureba-yoroshiidesuka
   2: mou-se-ba-youshi-desu-ka
   3: mou-se-ba-yoroshi-i-desu-ka
   4: gogo-sae-wa-youshi-desu-ka
   5: gogo-sae-wa-yoroshi-i-desu-ka

(024) MAU_MB2_05  |mazu tourokuyoushide tetsuzukio shiteitadakanakutewa narimaseNga|
 > 1: mazu-tourokuyoushi-de-tetsuzuki-o-shi-te-itada-ka-naku-te-wa-na-ri-mase-N-ga

(025) MAU_MB2_06  |mou tourokuyoushiwa omochideshouka|
 > 1: mou-tourokuyoushi-wa-omochi-desyo-u-ka
   2: mou-tourokuyoushi-ya-omochi-desyo-u-ka
   3: mou-tourokuyoushi-wa-omachi-desyo-u-ka
   4: mou-tourokuyoushi-e-wa-omochi-desyo-u-ka
   5: mou-tourokuyoushi-yara-omochi-desyo-u-ka

(026) MAU_MB2_07  |madadesu|
 > 1: mada-desu
   2: wadai-desu
   3: nani-ka-desu
   4: naN-ka-desu
   5: ima-na-N-desu
```

```
(027) MAU_MB2_08  |youshio okuttekudasai|
 > 1: youshi-o-oku-Q-te-kudasai
   2: youshi-mo-oku-Q-te-kudasai
   3: youshi-o-oku-Q-te-kudasai-yo
   4: youshi-mo-oku-Q-te-kudasai-yo
   5: youshi-o-oku-Q-te-kudasai-ne

(028) MAU_MB2_09  |dewa gojuushoto onamaeo onegaishimasu|
 > 1: dewa-gozyuusyo-to-onamae-o-onegai-shi-masu
   2: dewa-gozyuusyo-to-o-onamae-o-onegai-shi-masu
   3: dewa-gozyuusyo-to-onamae-mo-onegai-shi-masu
   4: dewa-gozyuusyo-to-o-onamae-mo-onegai-shi-masu
   5: dewa-gozyuusyo-to-onamae-e-mo-onegai-shi-masu

(029) MAU_MB2_10  |juushowa oosakashi higashiku tokuimachi ichino nidesu|
 > 1: zyuusyo-wa-oosaka-shi-higashi-ku-tokuimachi-ichi-no-ni-desu
   2: zyuusyo-ga-oosaka-shi-higashi-ku-tokuimachi-ichi-no-ni-desu
   3: gozyuusyo-wa-oosaka-shi-higashi-ku-tokuimachi-ichi-no-ni-desu
   4: zyuusyo-ni-wa-oosaka-shi-higashi-ku-tokuimachi-ichi-no-ni-desu
   5: zyuusyo-ya-oosaka-shi-higashi-ku-tokuimachi-ichi-no-ni-desu

(030) MAU_MB2_11  |namaewa shimizutaroodesu|
 > 1: namae-wa-shimizu-taroo-desu
   2: namae-ya-shimizu-taroo-desu
   3: namae-ni-wa-shimizu-taroo-desu
   4: namae-ga-shimizu-taroo-desu
   5: namae-wa-shimizu-taroo-hodo-desu

(031) MAU_MB2_12  |wakarimashita|
 > 1: waka-ri-mashi-ta
   2: kaka-ri-mashi-ta
   3: waka-ru-N-deshi-ta
   4: waka-ri-mashi-ta-ga
   5: waka-ru-rashii-ga

(032) MAU_MB2_13  |saNkaryouwa irunodeshouka|
 > 1: saNkaryou-wa-i-ru-no-desyo-u-ka
   2: saNkaryou-wa-ya-ru-no-desyo-u-ka
   3: saNkaryou-wa-yo-i-no-desyo-u-ka
   4: saNkaryou-wa-i-u-no-desyo-u-ka
   5: saNkaryou-wa-i-i-N-desyo-u-ka

(033) MAU_MB2_14  |hai|
 > 1: hai

(034) MAU_MB2_15  |tourokuhitoshite ohitori saNmaN goseNeNga hitsuyoudesu|
 > 1: tourokuhi-toshite-ohitori-saN-maN-go-seN-eN-ga-hitsuyou-desu
   2: tourokuhi-toshite-ohitori-saN-maN-go-seN-eN-wa-hitsuyou-desu
   3: tourokuhi-toshite-ohitori-saN-maN-go-seN-eN-ya-hitsuyou-desu
   4: tourokuhi-toshite-ohitori-saN-maN-go-seN-eN-nara-hitsuyou-desu
   5: touroku-toshite-ohitori-saN-maN-go-seN-eN-ga-hitsuyou-desu

(035) MAU_MB2_16  |soudesuka|
 > 1: sou-desu-ka
   2: i-i-soudesu-ka
   3: i-u-soudesu-ka
   4: sou-deshi-ta
   5: sou-desu-ga

(036) MAU_MB2_17  |doumo arigatougozaimashita|
 > 1: doumo-arigat-ou-gozaima-shi-ta
   2: dou-ni-mo-arigat-ou-gozaima-shi-ta
   3: doumo-arigat-ou-gozaima-shi-ta-ga
   4: dou-arigat-ou-gozaima-shi-ta
   5: doumo-arigat-ou-gozaima-su-ka
```

```
(037) MAU_MB2_18  |shitsureiitashimasu|
 > 1: shitsurei-ita-shi-masu
   2: shitsurei-ita-shi-masu-yo
   3: shitsurei-ita-shi-masu-ne
   4: shitsurei-ita-shi-masu-ga
   5: shitsurei-ita-shi-masu-ka

(038) MAU_M12_01  |moshimoshi|
 > 1: moshimoshi
   2: mochii-ru-rashii
   3: mo-chi-masyo-u
   4: mochii-masyo-u
   5: mou-shi-masyo-u

(039) MAU_M12_02  |sochirawa kaigijimukyokudesuka|
 > 1: sochira-wa-kaigizimukyoku-desu-ka
   2: sochira-wa-kaigizimukyoku-desu-ga
   3: sochira-wa-kaigizimukyoku-deshi-ta
   4: kochira-wa-kaigizimukyoku-desu-ka
   5: sochira-ga-kaigizimukyoku-desu-ka

(040) MAU_M12_03  |hai|
 > 1: hai

(041) MAU_M12_04  |soudesu|
 > 1: sou-desu
   2: i-u-soudesu
   3: i-i-soudesu
   4: souhu-desu
   5: i-ku-soudesu

(042) MAU_M12_05  |donoyouna goyoukeNdeshouka|
 > 1: dono-youna-goyoukeN-desyo-u-ka
   2: dono-youna-goyoukeN-desyo-u-ka
   3: dono-youna-goyoukeN-desyo-u-ga
   4: dono-youna-goyoukeN-desyo-u-ga
   5: gogo-yori-wa-goyoukeN-desyo-u-ka

(043) MAU_M12_06  |kaigini moushikomitainodesuga|
 > 1: kaigi-ni-moushiko-mi-tai-no-desu-ga
   2: kaiiN-ni-moushiko-mi-tai-no-desu-ga
   3: kaigi-ni-moushiko-mi-tai-N-desu-ga
   4: kaiiN-ni-moushiko-mi-tai-N-desu-ga
   5: kaigi-ni-moushiko-mi-tai-no-desu-ka

(044) MAU_M12_07  |donoyouna tetsuzukio sureba yoroshiinodeshouka|
 > 1: dono-youna-tetsuzuki-o-sure-ba-yoroshi-i-no-desyo-u-ka
   2: dono-youna-tetsuzuki-mo-sure-ba-yoroshi-i-no-desyo-u-ka
   3: dono-youna-tetsuzuki-o-sure-ba-yoroshi-i-N-desyo-u-ka
   4: dono-youna-tetsuzuki-mo-sure-ba-yoroshi-i-N-desyo-u-ka
   5: dono-youna-tetsuzuki-e-mo-sure-ba-yoroshi-i-no-desyo-u-ka

(045) MAU_M12_08  |tourokuyoushide tetsuzukio shitekudasai|
   1: tourokuyoushi-e-tetsuzuki-o-shi-te-kudasai
   2: tourokuyoushi-e-tetsuzuki-mo-shi-te-kudasai
   3: tourokuyoushi-e-tetsuzuki-e-mo-shi-te-kudasai
   4: tourokuyoushi-e-tetsuzuki-to-shi-te-kudasai
 > 5: tourokuyoushi-de-tetsuzuki-o-shi-te-kudasai

(046) MAU_M12_09  |tourokuyoushiwa sudeni omochideshouka|
 > 1: tourokuyoushi-wa-sudeni-omochi-desyo-u-ka
   2: tourokuyoushi-wa-sudeni-omachi-desyo-u-ka
   3: tourokuyoushi-ya-sudeni-omochi-desyo-u-ka
   4: tourokuyoushi-e-wa-sudeni-omochi-desyo-u-ka
   5: tourokuyoushi-wa-sudeni-mo-tsu-desyo-u-ka
```

```
(047) MAU_M12_10  |iie|
 > 1: iie

(048) MAU_M12_11  |madadesu|
 > 1: mada-desu
   2: wadai-desu
   3: n-ai-desu
   4: ima-desu
   5: baai-desu

(049) MAU_M12_12  |wakarimashita|
 > 1: waka-ri-mashi-ta
   2: kaka-ri-mashi-ta
   3: waka-ri-masyo-u-ka
   4: waka-ru-rashii-ka
   5: waka-ru-rashii-ga

(050) MAU_M12_13  |soredewa tourokuyoushio ookuriitashimasu|
   1: sore-e-wa-tourokuyoushi-o-ookuri-ita-shi-masu
   2: sore-e-wa-tourokuyoushi-mo-ookuri-ita-shi-masu
   3: kore-e-wa-tourokuyoushi-o-ookuri-ita-shi-masu
   4: kore-e-wa-tourokuyoushi-mo-ookuri-ita-shi-masu
 > 5: soredewa-tourokuyoushi-o-ookuri-ita-shi-masu

(051) MAU_M12_14  |gojuushoto onamaeo onegaishimasu|
 > 1: gozyuusyo-to-onamae-o-onegai-shi-masu
   2: gozyuusyo-to-o-onamae-o-onegai-shi-masu
   3: gozyuusyo-to-mo-onamae-o-onegai-shi-masu
   4: gozyuusyo-to-namae-o-onegai-shi-masu
   5: gozyuusyo-to-onamae-mo-onegai-shi-masu

(052) MAU_M12_15  |juushowa oosakashi kitaku chayamachi nijuusaNdesu|
 > 1: zyuusyo-wa-oosaka-shi-kita-ku-cyayamachi-ni-zyuu-saN-desu
   2: zyuusyo-wa-oosaka-shi-kita-ku-cyayamachi-go-zyuu-saN-desu
   3: zyuusyo-ga-oosaka-shi-kita-ku-cyayamachi-ni-zyuu-saN-desu
   4: gozyuusyo-wa-oosaka-shi-kita-ku-cyayamachi-ni-zyuu-saN-desu
   5: zyuusyo-ni-wa-oosaka-shi-kita-ku-cyayamachi-ni-zyuu-saN-desu

(053) MAU_M12_16  |namaewa suzukimayumidesu|
 > 1: namae-wa-suzuki-mayumi-desu
   2: namae-wa-suzuki-na-N-desu
   3: namae-wa-suzuki-mayumi-desu-yo
   4: namae-ni-wa-suzuki-mayumi-desu
   5: namae-wa-suzuki-mayumi-desu-ne

(054) MAU_M12_17  |wakarimashita|
 > 1: waka-ri-mashi-ta
   2: kaka-ri-mashi-ta
   3: waka-ru-rashii-ka
   4: waka-ri-mashi-ta-ga
   5: waka-ru-rashii-ga

(055) MAU_M12_18  |tourokuyoushio shikyuu okuraseteitadakimasu|
 > 1: tourokuyoushi-o-shikyuu-oku-ra-se-te-itada-ki-masu
   2: tourokuyoushi-mo-shikyuu-oku-ra-se-te-itada-ki-masu
   3: tourokuyoushi-e-mo-shikyuu-oku-ra-se-te-itada-ki-masu
   4: tourokuyoushi-o-shikyuu-oku-ra-se-te-itadake-masu
   5: tourokuyoushi-mo-shikyuu-oku-ra-se-te-itadake-masu

(056) MAU_M12_19  |yoroshiku onegaishimasu|
 > 1: yoroshiku-onegai-shi-masu
   2: yoroshiku-onegai-shi-masu-yo
   3: yoroshiku-onegai-shi-masu-ne
   4: yoroshi-ku-te-onegai-shi-masu
   5: youshi-e-onegai-shi-masu
```

(057) MAU_M12_20 |soredewa shitsureishimasu|
> 1: soredewa-shitsurei-shi-masu
2: sore-de-wa-shitsurei-shi-masu
3: sore-wa-shitsurei-shi-masu
4: sore-e-wa-shitsurei-shi-masu
5: soko-de-wa-shitsurei-shi-masu

(058) MAU_M22_01 |hai|
> 1: hai

(059) MAU_M22_02 |kochirawa kaigijimukyokudesu|
> 1: kochira-wa-kaigizimukyoku-desu
2: kochira-wa-kaigizimukyoku-desu-yo
3: akira-wa-kaigizimukyoku-desu
4: kochira-ga-kaigizimukyoku-desu
5: oheya-wa-kaigizimukyoku-desu

(060) MAU_M22_03 |kaigino saNkaryounitsuite oshieteitadakitainodesuga|
1: kagi-ri-no-saNkaryou-nitsuite-oshie-te-itada-ki-tai-N-desu-ga
2: kaigi-no-saNkaryou-nitsuite-oshie-te-itada-ki-tai-N-desu-ga
3: kagi-ri-no-saNkaryou-nitsuite-oshie-te-itada-ki-tai-no-desu-ga
> 4: kaigi-no-saNkaryou-nitsuite-oshie-te-itada-ki-tai-no-desu-ga
5: kaigi-mo-saNkaryou-nitsuite-oshie-te-itada-ki-tai-N-desu-ga

(061) MAU_M22_04 |ima kaigini moushikomeba saNkaryouwa ikuradesuka|
> 1: ima-kaigi-ni-moushiko-me-ba-saNkaryou-wa-ikura-desu-ka
2: i-u-ga-kaigi-ni-moushiko-me-ba-saNkaryou-wa-ikura-desu-ka
3: ima-kaiiN-ni-moushiko-me-ba-saNkaryou-wa-ikura-desu-ka
4: i-u-ga-kaiiN-ni-moushiko-me-ba-saNkaryou-wa-ikura-desu-ka
5: i-ru-ga-kaigi-ni-moushiko-me-ba-saNkaryou-wa-ikura-desu-ka

(062) MAU_M22_05 |hai|
> 1: hai

(063) MAU_M22_06 |saNkaryouwa geNzai ohitori saNmaN goseNeNdesu|
> 1: saNkaryou-wa-geNzai-ohitori-saN-maN-go-seN-eN-desu
2: saNkaryou-wa-geNzai-ohutari-saN-maN-go-seN-eN-desu
3: saNkaryou-wa-geNzai-ni-ohitori-saN-maN-go-seN-eN-desu
4: saNkaryou-wa-geNzai-ohitori-saN-maN-go-seN-eN-na-N-desu
5: saNkaryou-wa-geNzai-ohitori-saN-maN-go-seN-eN-nomi-desu

(064) MAU_M22_07 |raigetsu omoushikomininarimasuto yoNmaNeNdesu|
> 1: raigetsu-omoushikomi-ni-na-ri-masu-to-yoN-maN-eN-desu
2: raigetsu-omoushikomi-deki-masu-to-yoN-maN-eN-desu
3: raigetsu-omoushikomi-ni-na-ri-masu-to-ya-ra-nai-N-desu
4: raigetsu-omoushikomi-ni-na-ri-masu-to-yoN-maN-eN-na-N-desu
5: raigetsu-omoushikomi-ni-na-ri-masu-to-yoN-maN-eN-nomi-desu

(065) MAU_M22_08 |saNkaryouniwa yokoushuudaito kaNgeikaihiga fukumareteimasu|
> 1: saNkaryou-ni-wa-yokousyuudai-to-kaNgeikaihi-ga-huku-ma-re-te-i-masu
2: saNkaryou-ni-wa-yokousyuudai-to-o-kaNgeikaihi-ga-huku-ma-re-te-i-masu
3: saNkaryou-ya-yokousyuudai-to-kaNgeikaihi-ga-huku-ma-re-te-i-masu
4: saNkaryou-ya-yokousyuudai-to-o-kaNgeikaihi-ga-huku-ma-re-te-i-masu
5: saNkaryou-ni-wa-yokousyuu-nari-to-kaNgeikaihi-ga-huku-ma-re-te-i-masu

(066) MAU_M22_09 |watashiwa jouhoushorigakkaino kaiiNnanodesuga|
> 1: watashi-wa-zyouhousyorigaQkai-no-kaiiN-na-no-desu-ga
2: watashi-wa-zyouhousyorigaQkai-mo-kaiiN-na-no-desu-ga
3: watashi-wa-zyouhousyorigaQkai-no-kaiiN-nado-desu-ga
4: watashi-wa-zyouhousyorigaQkai-mo-kaiiN-nado-desu-ga
5: watashi-wa-zyouhousyorigaQkai-e-no-kaiiN-na-no-desu-ga

```
(067) MAU_M22_10  |saNkaryouno waribikiwa nainodesuka|
   1: saNkaryou-mo-waribiki-wa-n-ai-no-desu-ka
 > 2: saNkaryou-no-waribiki-wa-n-ai-no-desu-ka
   3: saNkaryou-mo-waribiki-wa-na-ru-no-desu-ka
   4: saNkaryou-mo-waribiki-ga-n-ai-no-desu-ka
   5: saNkaryou-no-waribiki-wa-na-ru-no-desu-ka

(068) MAU_M22_11  |koNkaiwa waribikio okonatteorimaseN|
 > 1: koNkai-wa-waribiki-o-okona-Q-te-ori-mase-N
   2: koNkai-wa-waribiki-mo-okona-Q-te-ori-mase-N
   3: koNkai-e-wa-waribiki-o-okona-Q-te-ori-mase-N
   4: koNkai-ya-waribiki-o-okona-Q-te-ori-mase-N
   5: koNkai-e-wa-waribiki-mo-okona-Q-te-ori-mase-N

(069) MAU_M22_12  |soudesuka|
 > 1: sou-desu-ka
   2: i-u-soudesu-ka
   3: i-i-soudesu-ka
   4: i-ku-soudesu-ka
   5: souhu-desu-ka

(070) MAU_M22_13  |saNkaryouwa donoyouni oshiharaishitara yoinodesuka|
   1: saNkaryou-wa-dono-youni-oshiharai-shi-tara-ya-ru-no-desu-ka
   2: saNkaryou-wa-dono-youni-oshiharai-shi-ta-ga-ya-ru-no-desu-ka
 > 3: saNkaryou-wa-dono-youni-oshiharai-shi-tara-yo-i-no-desu-ka
   4: saNkaryou-wa-dono-youni-oshiharai-shi-ta-ga-yo-i-no-desu-ka
   5: saNkaryou-wa-dono-youni-oshiharai-shi-tara-ya-ru-N-desu-ka

(071) MAU_M22_14  |saNkaryouwa giNkoufurikomidesu|
 > 1: saNkaryou-wa-giNkouhurikomi-desu
   2: saNkaryou-ga-giNkouhurikomi-desu
   3: saNka-yori-wa-giNkouhurikomi-desu
   4: saNkaryou-wa-giNkouhurikomi-desu-yo
   5: saNkaryou-wa-giNkouhurikomi-nomi-desu

(072) MAU_M22_15  |aNnaishoni kisaisareteiru kouzabaNgouni furikoNdekudasai|
 > 1: aNnaisyo-ni-kisai-sa-re-te-iru-kouzabaNgou-ni-huriko-N-de-kudasai

(073) MAU_M22_16  |mata kigeNwa kotoshiippaidesu|
 > 1: mata-kigeN-wa-kotoshi-iQpai-desu
   2: mata-kigeN-ga-kotoshi-iQpai-desu
   3: mata-kigeN-ni-wa-kotoshi-iQpai-desu
   4: mata-kigeN-nara-kotoshi-iQpai-desu
   5: mata-kigeN-wa-kotoshi-iQpai-na-N-desu

(074) MAU_M22_17  |wakarimashita|
 > 1: waka-ri-mashi-ta
   2: waka-ri-mashi-ta-ga
   3: kaka-ri-mashi-ta
   4: waka-ru-rashii-ka
   5: waka-ri-masu-ka

(075) MAU_M22_18  |doumo arigatougozaimashita|
 > 1: doumo-arigat-ou-gozaima-shi-ta
   2: dou-arigat-ou-gozaima-shi-ta
   3: dou-ni-mo-arigat-ou-gozaima-shi-ta
   4: gogo-arigat-ou-gozaima-shi-ta
   5: doumo-arigat-ou-gozaima-su-ka

(076) MAU_M22_19  |dou itashimashite|
 > 1: dou-ita-shi-mashi-te
   2: doumo-ita-shi-mashi-te
   3: dou-otazune-shi-te
   4: dou-tsui-tachi-deshi-ta
   5: dou-to-ita-shi-mashi-te
```

(077) MAU_M22_20 |wakaranai teNga gozaimashitara itsudemo okikikudasai|
> 1: waka-ra-nai-teN-ga-gozaima-shi-tara-itsu-demo-okiki-kudasai
2: waka-ra-nai-teN-ga-gozaima-shi-tara-itsu-demo-okiki-nasai
3: waka-ra-nai-teN-ga-gozaima-shi-tara-itsu-e-mo-okiki-kudasai
4: waka-ra-nai-teN-ga-gozaima-shi-tara-itsu-demo-ookuri-kudasai
5: waka-ra-nai-teN-ga-gozaima-shi-te-wa-itsu-demo-okiki-kudasai

(078) MAU_M22_21 |shitsureiitashimasu|
> 1: shitsurei-ita-shi-masu
2: shitsurei-ita-shi-masu-yo
3: shitsurei-ita-shi-masu-ne
4: kisai-ita-shi-masu
5: shitsurei-ita-shi-masu-N-desu

(079) MAU_M32_01 |hai|
> 1: hai

(080) MAU_M32_02 |kochirawa kaigijimukyokudesu|
> 1: kochira-wa-kaigizimukyoku-desu
2: akira-wa-kaigizimukyoku-desu
3: kochira-wa-kaigizimukyoku-desu-yo
4: hoka-wa-kaigizimukyoku-desu
5: oheya-wa-kaigizimukyoku-desu

(081) MAU_M32_03 |kaigini roNbuNo happyoushitaito omotteirunodesuga|
> 1: kaigi-ni-roNbuN-o-haQpyou-shi-tai-to-omo-Q-te-iru-no-desu-ga
2: kaigi-ni-roNbuN-mo-haQpyou-shi-tai-to-omo-Q-te-iru-no-desu-ga

(082) MAU_M32_04 |kaigino naiyounitsuite oshietekudasai|
> 1: kaigi-no-naiyou-nitsuite-oshie-te-kudasai
2: kaN-shi-no-naiyou-nitsuite-oshie-te-kudasai
3: kagi-ri-no-naiyou-nitsuite-oshie-te-kudasai
4: kaigi-mo-naiyou-nitsuite-oshie-te-kudasai
5: kaiiN-no-naiyou-nitsuite-oshie-te-kudasai

(083) MAU_M32_05 |koNkaino kaigiwa tsuuyakudeNwani kaNreNsuru kouhaNna keNkyuubuNyao fukuNdeimasu|
> 1: koNkai-no-kaigi-wa-tsuuyakudeNwa-ni-kaNreN-suru-kouhaN-na-keNkyuubuNya-o-huku-N-de-i-masu

(084) MAU_M32_06 |geNgogakuya shiNrigakuo seNkousuru katanimo saNkashiteitadaku yoteidesu|
> 1: geNgogaku-ya-shiNrigaku-o-seNkou-suru-kata-ni-mo-saNka-shi-te-itada-ku-yotei-desu

(085) MAU_M32_07 |wakarimashita|
> 1: waka-ri-mashi-ta
2: kaka-ri-mashi-ta
3: waka-ru-rashii-ka
4: waka-ru-rashii-ga
5: waka-ri-masyo-u-ka

(086) MAU_M32_08 |tokorode kaigideno koushikigeNgowa naNdesuka|
1: tokorode-kaigi-de-mo-koushikigeNgo-wa-naN-desu-ka
2: tokorode-kaigi-demo-koushikigeNgo-wa-naN-desu-ka
> 3: tokorode-kaigi-de-no-koushikigeNgo-wa-naN-desu-ka
4: tokorode-kaigi-de-mo-koushikigeNgo-wa-na-ru-N-desu-ka
5: tokorode-kaigi-demo-koushikigeNgo-wa-na-ru-N-desu-ka

(087) MAU_M32_09 |eegoto nihoNgodesu|
> 1: eego-to-nihoNgo-desu
2: eego-to-yo-i-N-desu
3: eego-to-o-yo-i-N-desu
4: eego-to-nihoNgo-desu
5: eego-to-yo-i-no-desu

```
(088) MAU_M32_10  |watashiwa nihoNgoga zeNzeN wakaranainodesuga|
   1: watashi-wa-nihoNgo-wa-zeNzeN-waka-ra-nai-N-desu-ga
   2: watashi-ga-nihoNgo-wa-zeNzeN-waka-ra-nai-N-desu-ga
   3: watashi-wa-nihoNgo-wa-zeNzeN-waka-ra-nai-no-desu-ga

(089) MAU_M32_11  |happyouga nihoNgode okonawareru baai eegoeno doujitsuuyakuwa arunodesuka|
 > 1: haQpyou-ga-nihoNgo-de-okona-wa-reru-baai-eego-e-no-douzitsuuyaku-wa-a-ru-no-desu-ka
   2: haQpyou-ga-nihoNgo-de-okona-wa-reru-baai-eego-e-no-douzitsuuyaku-wa-ya-ru-no-desu-ka
   3: haQpyou-ga-nihoNgo-de-okona-wa-reru-baai-eego-e-no-douzitsuuyaku-wa-ka-ku-no-desu-ka
   4: haQpyou-ga-nihoNgo-de-okona-wa-reru-baai-eego-e-no-douzitsuuyaku-wa-na-ru-no-desu-ka

(090) MAU_M32_12  |hai|
 > 1: hai

(091) MAU_M32_13  |eegoeno doujitsuuyakuo youishiteorimasu|
 > 1: eego-e-no-douzitsuuyaku-o-youi-shi-te-ori-masu
   2: eego-e-mo-douzitsuuyaku-o-youi-shi-te-ori-masu
   3: eego-e-no-douzitsuuyaku-o-youi-shi-te-ori-masu
   4: eego-e-mo-douzitsuuyaku-o-youi-shi-te-ori-masu
   5: eego-e-no-douzitsuuyaku-mo-youi-shi-te-ori-masu

(092) MAU_M32_14  |wakarimashita|
 > 1: waka-ri-mashi-ta
   2: waka-ri-mashi-ta-ga
   3: waka-ru-rashii-ka
   4: kaka-ri-mashi-ta
   5: waka-ri-mashi-ta-ka

(093) MAU_M32_15  |doumo arigatougozaimashita|
 > 1: doumo-arigat-ou-gozaima-shi-ta
   2: gogo-arigat-ou-gozaima-shi-ta
   3: gogo-o-arigat-ou-gozaima-shi-ta
   4: dou-ni-mo-arigat-ou-gozaima-shi-ta
   5: dou-arigat-ou-gozaima-shi-ta

(094) MAU_M32_16  |sayounara|
 > 1: sayounara
   2: suru-youda

(095) MAU_M42_01  |kochirawa kaigijimukyokudesu|
 > 1: kochira-wa-kaigizimukyoku-desu
   2: kochira-wa-kaigizimukyoku-desu-yo
   3: kochira-wa-kaigizimukyoku-desu-ne
   4: kochira-wa-kaigizimukyoku-desu-ka
   5: kochira-wa-kaigizimukyoku-hodo-desu

(096) MAU_M42_02  |kaiginitsuite kuwashii kotoo oshietekudasai|
 > 1: kaigi-nitsuite-kuwashi-i-koto-o-oshie-te-kudasai
   2: kaiiN-nitsuite-kuwashi-i-koto-o-oshie-te-kudasai
   3: kaigi-nitsuite-kuwashi-i-koto-mo-oshie-te-kudasai
   4: kaigi-nomi-nitsuite-kuwashi-i-koto-o-oshie-te-kudasai
   5: kaiiN-nitsuite-kuwashi-i-koto-mo-oshie-te-kudasai

(097) MAU_M42_03  |kaigino aNnaishowa omochidesuka|
 > 1: kaigi-no-aNnaisyo-wa-omochi-desu-ka
   2: kaigi-no-aNnaisyo-wa-omachi-desu-ka
   3: kagi-ri-no-aNnaisyo-wa-omochi-desu-ka
   4: kagi-ri-no-aNnaisyo-wa-omachi-desu-ka
   5: kaigi-mo-aNnaisyo-wa-omochi-desu-ka

(098) MAU_M42_04  |iie|
 > 1: iie
```

(099) MAU_M42_05 |motteimaseN|
> 1: mo-Q-te-i-mase-N
2: no-Q-te-i-mase-N
3: mo-Q-te-i-masu
4: no-Q-te-i-masu
5: omo-Q-te-i-mase-N

(100) MAU_M42_06 |soudesuka|
> 1: sou-desu-ka
2: i-u-soudesu-ka
3: i-i-soudesu-ka
4: sou-desu-ga
5: i-u-soudesu-ga

(101) MAU_M42_07 |kaigiwa hachigatsu nijuuninichikara nijuugonichimade kyoutokokusaikaigijoude kaisaisaremasu|
> 1: kaigi-wa-hachi-gatsu-nizyuu-ni-nichi-kara-nizyuu-go-nichi-made-kyoutokokusaikaigizyou-de -kaisai-sa-re-masu
2: kagi-re-ba-hachi-gatsu-nizyuu-ni-nichi-kara-nizyuu-go-nichi-made-kyoutokokusaikaigizyou-de -kaisai-sa-re-masu
3: kaigi-ga-hachi-gatsu-nizyuu-ni-nichi-kara-nizyuu-go-nichi-made-kyoutokokusaikaigizyou-de -kaisai-sa-re-masu

(102) MAU_M42_08 |saNkaryouwa yoNmaNeNdesu|
> 1: saNkaryou-wa-yoN-maN-eN-desu
2: saNkaryou-ga-yoN-maN-eN-desu
3: saNka-yori-wa-yoN-maN-eN-desu
4: saNkaryou-ni-wa-yoN-maN-eN-desu
5: saNkaryou-nara-yoN-maN-eN-desu

(103) MAU_M42_09 |happyouo kibousarerunodeshitara saNgatsu hatsukamadeni youyakuo teishutsushitekudasai|
> 1: haQpyou-o-kibou-sa-reru-no-deshi-tara-saN-gatsu-hatsu-ka-made-ni-youyaku-o-teisyutsu-shi-te-kudasai

(104) MAU_M42_10 |kaigino aNnaishoo ookuriitashimasunode soreo goraNkudasai|
> 1: kaigi-no-aNnaisyo-o-ookuri-ita-shi-masu-node-sore-o-goraN-kudasai
2: kaigi-no-aNnaisyo-o-ookuri-ita-shi-masu-no-de-sore-o-goraN-kudasai
3: kaigi-mo-aNnaisyo-o-ookuri-ita-shi-masu-node-sore-o-goraN-kudasai
4: kaigi-mo-aNnaisyo-o-ookuri-ita-shi-masu-no-de-sore-o-goraN-kudasai

(105) MAU_M42_11 |shitsureidesuga onamaeto gojuushoo onegaiitashimasu|
> 1: shitsurei-desu-ga-onamae-to-gozyuusyo-o-onegai-ita-shi-masu
2: shitsurei-desu-ga-onamae-to-o-gozyuusyo-o-onegai-ita-shi-masu
3: shitsurei-desu-nara-onamae-to-gozyuusyo-o-onegai-ita-shi-masu

(106) MAU_M42_12 |adamusumisudesu|
> 1: adamu-sumisu-desu
2: adamu-sumisu-desu-yo
3: adamu-sumisu-desu-ne
4: adamu-sumisu-desu-ga
5: adamu-sumisu-desu-ka

(107) MAU_M42_13 |juushowa oosakashi higashiku tamatsukuri nichoume nijuunanano nanadesu|
> 1: zyuusyo-wa-oosaka-shi-higashi-ku-tamatsukuri-ni-cyoume-ni-zyuu-nana-no-nana-desu
2: zyuusyo-wa-oosaka-shi-higashi-ku-tamatsukuri-ni-cyoume-go-zyuu-nana-no-nana-desu
3: zyuusyo-ga-oosaka-shi-higashi-ku-tamatsukuri-ni-cyoume-ni-zyuu-nana-no-nana-desu
4: zyuusyo-wa-oosaka-shi-higashi-ku-tamatsukuri-go-cyoume-ni-zyuu-nana-no-nana-desu
5: gozyuusyo-wa-oosaka-shi-higashi-ku-tamatsukuri-ni-cyoume-ni-zyuu-nana-no-nana-desu

(108) MAU_M42_14 |wakarimashita|
> 1: waka-ri-mashi-ta
2: kaka-ri-mashi-ta
3: waka-ru-rashii-ka
4: waka-ru-rashii-ga
5: waka-ri-mashi-ta-ga

```
(109) MAU_M42_15  |deNwabaNgoumo okikishitainodesuga|
 > 1: deNwabaNgou-mo-okiki-shi-tai-no-desu-ga
   2: deNwabaNgou-mo-okiki-shi-tai-N-desu-ga
   3: deNwabaNgou-mo-okiki-shi-nai-no-desu-ga
   4: deNwabaNgou-mo-okiki-shi-tai-no-desu-ka
   5: deNwabaNgou-ni-mo-okiki-shi-tai-no-desu-ga

(110) MAU_M42_16  |hai|
 > 1: hai

(111) MAU_M42_17  |saNnananino hachizeroichihachidesu|
 > 1: saN-nana-ni-no-hachi-zero-ichi-hachi-desu
   2: saN-nana-go-no-hachi-zero-ichi-hachi-desu
   3: saN-nana-ni-no-hachi-zero-hachi-hachi-desu
   4: saN-nana-nana-no-hachi-zero-ichi-hachi-desu
   5: saN-nana-ni-no-hachi-zero-zero-hachi-desu

(112) MAU_M42_18  |saNnananino hachizeroichihachidegozaimasune|
 > 1: saN-nana-ni-no-hachi-zero-ichi-hachi-de-gozaima-su-ne
   2: saN-nana-ni-no-hachi-zero-ichi-hachi-de-gozaima-su
   3: saN-nana-ni-no-hachi-zero-zero-hachi-de-gozaima-su-ne
   4: saN-nana-ni-no-hachi-zero-zero-ichi-de-gozaima-su-ne
   5: saN-nana-ni-no-hachi-zero-hachi-hachi-de-gozaima-su-ne

(113) MAU_M42_19  |hai|
 > 1: hai

(114) MAU_M42_20  |soudesu|
 > 1: sou-desu
   2: i-u-soudesu
   3: i-i-soudesu
   4: souhu-desu
   5: i-ku-soudesu

(115) MAU_M42_21  |soredewa yoroshiku onegaishimasu|
   1: sore-e-wa-yoroshiku-onegai-shi-masu
   2: sore-e-wa-yoroshi-ku-onegai-shi-masu
 > 3: soredewa-yoroshiku-onegai-shi-masu
   4: sore-de-wa-yoroshi-ku-onegai-shi-masu
   5: sore-de-wa-yoroshiku-onegai-shi-masu

(116) MAU_M42_22  |shitsureishimasu|
 > 1: shitsurei-shi-masu
   2: shitsurei-shi-masu-yo
   3: shitsurei-shi-masu-ne

(117) MAU_M52_01  |hai|
 > 1: hai

(118) MAU_M52_02  |kochirawa kaigijimukyokudegozaimasu|
 > 1: kochira-wa-kaigizimukyoku-de-gozaima-su
   2: kochira-wa-kaigizimukyoku-bakari-desu
   3: hoka-wa-kaigizimukyoku-de-gozaima-su
   4: kochira-wa-kaigizimukyoku-kurai-desu
   5: akira-wa-kaigizimukyoku-de-gozaima-su

(119) MAU_M52_03  |chotto onegaiga arunodesuga|
 > 1: cyoQto-onegai-ga-a-ru-no-desu-ga
   2: cyoQto-onegai-ga-a-ru-no-desu-ka
   3: cyoQto-onegai-ga-ya-ru-no-desu-ga
   4: cyoQto-onegai-ga-ka-ku-no-desu-ga
   5: cyoQto-onegai-ga-na-ru-no-desu-ga
```

(120) MAU_M52_04 |watashiwa kaigini moushikomio shita monodesu|
> 1: watashi-wa-kaigi-ni-moushikomi-o-shi-ta-mono-desu
2: watashi-wa-kaiiN-ni-moushikomi-o-shi-ta-mono-desu
3: watashi-wa-kaigi-ni-moushiko-mi-mo-shi-ta-mono-desu
4: watashi-wa-kaigi-ni-moushikomi-mo-shi-ta-mono-desu
5: watashi-wa-kaigi-ni-moushikomi-o-shi-ta-gogo-desu

(121) MAU_M52_05 |saNkao torikeshitainodesuga|
> 1: saNka-o-torike-shi-tai-no-desu-ga
2: saNka-o-torike-shi-tai-no-desu-ka
3: saNka-o-torike-shi-tai-N-desu-ga
4: saNka-o-torike-shi-tai-N-desu-ka
5: saNka-mo-torike-shi-tai-no-desu-ga

(122) MAU_M52_06 |onamaeo oukagaidekimasudeshouka|
> 1: onamae-o-oukagai-deki-masu-desyo-u-ka
2: onamae-mo-oukagai-deki-masu-desyo-u-ka
3: onamae-e-mo-oukagai-deki-masu-desyo-u-ka
4: namae-o-oukagai-deki-masu-desyo-u-ka
5: onamae-o-oukagai-deki-masu-desyo-u-ga

(123) MAU_M52_07 |hai|
> 1: hai

(124) MAU_M52_08 |berukeNno jimuwaiberudesu|
> 1: berukeN-no-zimu-waiberu-desu
2: berukeN-mo-zimu-waiberu-desu
3: berukeN-ni-no-zimu-waiberu-desu
4: berukeN-ni-mo-zimu-waiberu-desu
5: berukeN-e-no-zimu-waiberu-desu

(125) MAU_M52_09 |sudeni tourokuryouno hachimaN goseNeNo furikomareteoraremasune|
1: sudeni-tourokuryou-mo-hachi-maN-go-seN-eN-mo-huriko-ma-re-te-ora-re-masu-ne
2: sudeni-tourokuryou-no-hachi-maN-go-seN-eN-mo-huriko-ma-re-te-ora-re-masu-ne
3: sudeni-tourokuryou-mo-hachi-maN-go-seN-eN-o-huriko-ma-re-te-ora-re-masu-ne
4: sudeni-tourokuryou-mo-hachi-maN-go-seN-eN-mo-huriko-ma-re-te-ora-re-masu
5: sudeni-tourokuryou-no-hachi-maN-go-seN-eN-mo-huriko-ma-re-te-ora-re-masu

(126) MAU_M52_10 |hai|
> 1: hai

(127) MAU_M52_11 |soudesu|
> 1: sou-desu
2: i-u-soudesu
3: i-i-soudesu
4: souhu-desu
5: i-ku-soudesu

(128) MAU_M52_12 |tourokuryouo haraimodoshiteitadakemasuka|
1: tourokuryou-o-haraimodo-shi-te-itada-ki-masu-ka
> 2: tourokuryou-o-haraimodo-shi-te-itadake-masu-ka
3: tourokuryou-o-haraimodo-shi-te-itada-ki-masu-ga
4: tourokuryou-o-haraimodo-shi-te-itadake-masu-ga
5: tourokuryou-mo-haraimodo-shi-te-itada-ki-masu-ka

(129) MAU_M52_13 |okinodokudesuga dekimaseN|
> 1: okinodoku-desu-ga-deki-mase-N
2: okinodoku-desu-nara-deki-mase-N
3: okinodoku-desu-kara-deki-mase-N
4: okinodoku-desu-ga-deki-masu
5: okinodoku-desu-nara-ba-deki-mase-N

```
(130) MAU_M52_14  |aNnaishonimo kaiteimasuga kugatsu nijuunananichi igono torikeshinitaisuru
                   haraimodoshiwa dekimaseN|
 > 1: aNnaisyo-ni-mo-ka-i-te-i-masu-ga-ku-gatsu-nizyuu-nana-nichi-igo-no-torikeshi-nitaisuru
      -haraimodoshi-wa-deki-mase-N
   2: aNnaisyo-ni-mo-ka-i-te-i-masu-ga-ku-gatsu-nizyuu-nana-nichi-igo-no-torikeshi-nitaisuru
      -haraimodoshi-wa-deki-masu
   3: aNnaisyo-ni-mo-ka-i-te-i-masu-ga-ku-gatsu-nizyuu-nana-nichi-igo-no-torikeshi-nitaisuru
      -haraimodoshi-wa-i-ki-mase-N
   4: aNnaisyo-ni-mo-ka-i-te-i-masu-ga-ku-gatsu-nizyuu-nana-nichi-igo-no-torikeshi-nitaisuru
      -haraimodoshi-wa-deki-masu-ne
   5: aNnaisyo-ni-mo-ka-i-te-i-masu-ga-ku-gatsu-nizyuu-nana-nichi-igo-no-torikeshi-nitaisuru
      -haraimodoshi-wa-mi-mase-N

(131) MAU_M52_15  |gojitsu puroguramuto yokoushuuo ookuriitashimasu|
 > 1: gozitsu-puroguramu-to-yokousyuu-o-ookuri-ita-shi-masu
   2: gozitsu-puroguramu-to-o-yokousyuu-o-ookuri-ita-shi-masu
   3: gozitsu-puroguramu-to-yokousyuu-mo-ookuri-ita-shi-masu

(132) MAU_M52_16  |dewa darekaga watashino kawarini saNkasuru kotowa dekimasuka|
 > 1: dewa-dare-ka-ga-watashi-no-kawari-ni-saNka-suru-koto-wa-deki-masu-ka
   2: dewa-dare-ka-ga-watashi-no-kawari-ni-saNka-suru-koto-wa-deki-masu-ga

(133) MAU_M52_17  |sorewa betsuni moNdaiarimaseN|
 > 1: sore-wa-betsuni-moNdaiarimaseN
   2: sore-to-wa-betsuni-moNdaiarimaseN
   3: sore-ni-wa-betsuni-moNdaiarimaseN
   4: sore-ya-betsuni-moNdaiarimaseN
   5: sore-yara-betsuni-moNdaiarimaseN

(134) MAU_M52_18  |dairiniNga saNkasuru baaiwa arakajime kochiramade oshirasekudasai|
 > 1: dairiniN-ga-saNka-suru-baai-wa-arakazime-kochira-made-oshirase-kudasai

(135) MAU_M52_19  |wakarimashita|
 > 1: waka-ri-mashi-ta
   2: kaka-ri-mashi-ta
   3: waka-ri-mashi-ta-ga
   4: waka-ru-rashii-ka
   5: waka-ru-rashii-ga

(136) MAU_M52_20  |dairiniNga kimarimashitara oshiraseitashimasu|
 > 1: dairiniN-ga-kima-ri-mashi-tara-oshirase-ita-shi-masu
   2: dairiniN-ga-kima-ri-mashi-ta-ga-oshirase-ita-shi-masu
   3: dairiniN-ya-kima-ri-mashi-tara-oshirase-ita-shi-masu
   4: dairiniN-ga-kima-ri-mashi-cya-oshirase-ita-shi-masu
   5: dairiniN-ga-kima-ri-mashi-ta-nara-oshirase-ita-shi-masu

(137) MAU_M52_21  |dewa shitsureishimasu|
 > 1: dewa-shitsurei-shi-masu
   2: dewa-kisai-shi-masu
   3: dewa-shitsurei-shi-masu-yo
   4: dewa-suzuki-desu
   5: ima-shitsurei-shi-masu
```

Bibliography

[Aho 86] Aho, A. V., Sethi, R., Ullman, J. D.
Compilers, Principles, Techniques, and Tools.
Addison-Wesley, Massachusetts, 1986.

[Araki 89] Araki, T., Murakami, J., Ikehara, S.
"Effect of Reducing Ambiguity of Recognition Candidates in Japanese Bunsetsu Units by 2nd-Order Markov Model of Syllables".
Transactions of Information Processing Society of Japan, Vol. 30, No. 4, pp. 467-477, April, 1989 (in Japanese).

[Asadi 90] Asadi, A., Schwartz, R., Makhoul, J.
"Automatic Detection of New Words in a Large Vocabulary Continuous Speech Recognition System".
Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 125-128, April, 1990.

[Averbuch 87] Averbuch, A., Bahl, L., Bakis, R., Brown, P., Daggett, G., Das, S., Davies, K., Gennaro, S. D., Souza, P., Epstein, E., Fraleigh, D., Jelinek, F., Lewis, B., Mercer, R., Moorhead, J., Nadas, A., Nahamoo, D., Picheny, M., Shichman, G., Spinelli, P., Compernolle, D. V., Wilkens, H.
"Experiments with the TANGORA 20,000 Words Speech Recognizer".
Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 701-704, April, 1987.

[Bahl 83] Bahl, L. R., Jelinek, F., Mercer, R. L.
"A Maximum Likelihood Approach to Continuous Speech Recognition".
IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. PAMI-5, No. 2, pp. 179-190, March, 1983.

[Baker 75] Baker, J. K.
"The DRAGON System — An Overview".
IEEE Transactions on Acoustics, Speech, and Signal Processing, Vol. ASSP-23, pp. 24-29, February, 1975.

[Bridle 82] Bridle, J. S., Brown, M. D., Chamberlain, R. M.
"An Algorithm for Connected Word Recognition".
Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 899-902, May, 1982.

[Chomsky 57] Chomsky, N.
Syntactic Structures,
Mouton Publishers, The Hague, 1957.

[Chow 87] Chow, Y. L., Dunham, M. O., Kimball, O. A., Krasner, M. A., Kubala, G. F., Makhoul, J., Price, P. J., Roucos, S., Schwartz, R. M.
"BYBLOS: The BBN Continuous Speech Recognition System".
Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 89-92, April, 1987.

[Chow 89] Chow, Y. L., Roukos, S.
"Speech Understanding Using a Unification Grammar".
Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 727-730, May, 1989.

[Cohen 90] Cohen, M., Murveit, H., Bernstein, J.
"The DECIPHER Speech Recognition System".
Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 77-80, April, 1990.

[Earley 70] Earley, J.
"An Efficient Context-Free Parsing Algorithm".
Communication of the ACM, Vol. 13, No. 2, pp. 94-102, 1970.

[Ehara 90] Ehara, T., Ogura, K., Morimoto, T.
"ATR Dialogue Database".
Proceedings of the International Conference on Spoken Language Processing, pp. 1093-1096, November, 1990.

[Erman 80] Erman, L. D., Lesser, V. R.
"The HEARSAY-II Speech Understanding System: A Tutorial".
Trends in Speech Recognition, Lea, W. A. Ed, pp. 361-381, Prentice-Hall, 1980.

[Fu 74] Fu, K. S.
Syntactic Methods in Pattern Recognition.
Academic Press, New York, 1974.

[Fujisaki 91] Fujisaki, T., Jelinek, F., Cocke, E., Black, E., Nishino, T.
"A Probabilistic Parsing Method for Sentence Disambiguation".
Current Issues in Parsing Technology, Tomita, M. Ed, pp. 139-152, Kluwer Academic Publishers, Boston, 1991.

[Gupta 87] Gupta, V. N., Lennig, M., Mermelstein, P.
"Integration of Acoustic Information in a Large Vocabulary Word Recognizer".
Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 697-700, April, 1987.

[Hanazawa 90a] Hanazawa, T., Kita, K., Nakamura, S., Kawabata, T., Shikano, K.
"ATR HMM-LR Continuous Speech Recognition System".
Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 53-56, April, 1990.

[Hanazawa 90b] Hanazawa, T., Kita, K., Nakamura, S., Kawabata, T., Shikano, K.
"HMM-LR Speech Recognition System Performance".
The Journal of the Acoustical Society of Japan, Vol. 46, No. 10, pp. 817-823, October, 1990 (in Japanese).

[Hemphill 89] Hemphill, C., Picone, J.
"Speech Recognition in a Unification Grammar Framework".
Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 723-726, May, 1989.

[Hosaka 91] Hosaka, J., Takezawa, T.
"Syntactic Rules for Speech Recognition in SL-TRANS".
ATR Technical Report, TR-I-0193, February, 1991 (in Japanese).

[Itakura 75] Itakura, F.
"Minimum Prediction Residual Principle Applied to Speech Recognition".
IEEE Transactions on Acoustics, Speech, and Signal Processing, Vol. ASSP-23, pp. 67-72, February, 1975.

[Jelinek 80] Jelinek, F., Mercer, R.
"Interpolated Estimation of Markov Source Parameters from Sparse Data".
Pattern Recognition in Practice, Gelsema, E. S. and Kanal, L. N. Eds, pp. 381-397, 1980.

[Jelinek 90] Jelinek, F.
"Self-Organized Language Modeling for Speech Recognition".
Readings in Speech Recognition, Waibel, A. and Lee, K. F. Eds, pp. 450-506, Morgan Kaufmann Publishers, San Mateo, 1990.

[Kawabata 89a] Kawabata, T., Shikano, K., Kita, K.
"Task Entropy and Phone Perplexity".
Proceedings of the Acoustical Society of Japan Spring Meeting, pp. 93-94, March, 1989 (in Japanese).

[Kawabata 89b] Kawabata, T., Shikano, K.
"Island-Driven Continuous Speech Recognizer Using Phone-Based HMM Word Spotting".
Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 461-464, May, 1989.

[Kawabata 90] Kawabata, T., Hanazawa, T., Itoh, K., Shikano, K.
"Japanese Phonetic Typewriter Using HMM Phone Units and Syllable Trigrams".
Proceedings of the International Conference on Spoken Language Processing, pp. 717-720, November, 1990.

[Kita 89a] Kita, K., Kawabata, T., Saito, H.
"HMM Continuous Speech Recognition Using Predictive LR Parsing".
Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 703-706, May, 1989.

[Kita 89b] Kita, K., Kawabata, T., Saito, H.
"Parsing Continuous Speech by HMM-LR Method".
Proceedings of the International Workshop on Parsing Technologies, pp. 126-131, August, 1989.

[Kita 90a] Kita, K.
"Generalized LR Parsing in Hidden Markov Model".
ATR Technical Report, TR-I-0161, April, 1990.

[Kita 90b] Kita, K., Kawabata, T., Saito, H.
"HMM Continuous Speech Recognition Using Generalized LR Parsing".
Transactions of Information Processing Society of Japan, Vol. 31, No. 3, pp. 472-480, March, 1990 (in Japanese).

[Kita 90c] Kita, K., Kawabata, T., Hanazawa, T.
"HMM Continuous Speech Recognition Using Stochastic Language Models".
Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 581-584, April, 1990.

[Kita 90d] Kita, K., Takezawa, T., Hosaka, J., Ehara, T., Morimoto, T.
"Continuous Speech Recognition Using Two-Level LR Parsing".
Proceedings of the International Conference on Spoken Language Processing, pp. 905-908, November, 1990.

[Kita 91a] Kita, K., Ehara, T., Morimoto, T.
"Processing Unknown Words in Continuous Speech Recognition".
Proceedings of the Second International Workshop on Parsing Technologies, pp. 136-142, February, 1991.

[Kita 91b] Kita, K., Ehara, T., Morimoto, T.
"Very Large-Vocabulary Speech Recognition Using HMM-LR".
Proceedings of the Information Processing Society of Japan Spring Meeting, Vol. 2, 6D-3, pp. 110-111, March, 1991 (in Japanese).

[Kita 91c] Kita, K., Kawabata, T., Hanazawa, T.
"HMM Speech Recognition Using Stochastic Language Models".
The Journal of the Acoustical Society of Japan (E), Vol. 12, No. 3, pp. 99-105, May, 1991.

[Kita 91d] Kita, K., Ward, W. H.
"Incorporating LR Parsing into SPHINX".
Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 269-272, May, 1991.

[Kita 91e] Kita, K., Takezawa, T., Morimoto, T.
"Continuous Speech Recognition Using Two-Level LR Parsing".
IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, Vol. E74, No. 7, pp. 1806-1810, July, 1991.

[Kita 91f] Kita, K., Ehara, T., Morimoto, T.
"Processing Unknown Words in Continuous Speech Recognition".
IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, Vol. E74, No. 7, pp. 1811-1816, July, 1991.

[Kita 91g] Kita, K., Morimoto, T., Sagayama, S.
"LR Parsing with a Category Reachability Test Applied to Speech Recognition".
Proceedings of the Acoustical Society of Japan Autumn Meeting, pp. 171-172, October, 1991.

[Klatt 77] Klatt, D. H.
"Review of the ARPA Speech Understanding Project".
The Journal of the Acoustical Society of America, Vol. 62, No. 6, pp. 1324-1366, December, 1977.

[Kobayashi 90] Kobayashi, Y., Niimi, Y.
"Evaluation of a Speech Understanding System — SUSKIT-2".
Proceedings of the International Conference on Spoken Language Processing, pp. 725-728, November, 1990.

[Kohonen 88] Kohonen, T.
"The 'Neural' Phonetic Typewriter".
IEEE Computer, pp. 11-22, March, 1988.

[Komori 91] Komori, Y., Hatazaki, K.
"An Integration of Knowledge and Neural Networks toward a Phoneme Typewriter without a Language Model".
IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, Vol. E74, No. 7, pp. 1797-1805, July, 1991.

[Kurematsu 87] Kurematsu, A.
"Automatic Telephone Interpretation: A Basic Study".
ATR Technical Report, TR-I-0001, May, 1987.

[Kuwabara 89] Kuwabara, H., Takeda, K., Sagisaka, Y., Katagiri, S., Morikawa, S., Watanabe, T.
"Construction of a Large-Scale Japanese Speech Database and its Management System".
Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 560-563, May, 1989.

[Kwasny 81] Kwasny, S. C., Sondheimer, N. K.
"Relaxation Techniques for Parsing Grammatically Ill-Formed Input in Natural Language Understanding Systems".
American Journal of Computational Linguistics, Vol. 7, No. 2, pp. 99-108, April-June, 1981.

[Langendoen 84] Langendoen, D. T., Postal, P. M.
The Vastness of Natural Languages.
Basil Blackwell, Oxford, 1984.

[Lee 89] Lee, K. F.
Automatic Speech Recognition: The Development of the SPHINX *System.*
Kluwer Academic Publishers, Boston, 1989.

[Levinson 83] Levinson, S. E., Rabiner, L. R., Sondhi, M. M.
"An Introduction to the Application of the Theory of Probabilistic Functions of a Markov Process to Automatic Speech Recognition".
The Bell System Technical Journal, Vol. 62, No. 4, April, 1983.

[Linde 80] Linde, Y., Buzo, A., Gray, R. M.
"An Algorithm for Vector Quantizer Design".
IEEE Transactions on Communication, Vol. COM-28, No. 1, pp. 84-95, January, 1980.

[Lowerre 80] Lowerre, B., Reddy, R.
"The HARPY Speech Understanding System".
Trends in Speech Recognition, Lea, W. A. Ed, pp. 340-360, Prentice-Hall, 1980.

[Makino 91] Makino, S., Ito, A., Endo, M., Kido, K.
"A Japanese Text Dictation System Based on Phoneme Recognition and a Dependency Grammar".
Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 273-276, May, 1991.

[Maruyama 89] Maruyama, K., Hanazawa, T., Kawabata, T., Shikano, K.
"English Word Recognition Using HMM Phone Concatenated Training".
IEICE Technical Report, SP 88-119, January, 1989 (in Japanese).

[Matsunaga 90] Matsunaga, S., Sagayama, S., Homma, S., Furui, S.
"A Continuous Speech Recognition System Based on a Two-Level Grammar Approach".
Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 589-592, April, 1990.

[Morimoto 90] Morimoto, T., Shikano, K., Iida, H., Kurematsu, A.
"Integration of Speech Recognition and Language Processing in Spoken Language Translation System (SL-TRANS)".
Proceedings of the International Conference on Spoken Language Processing, pp. 921-924, November, 1990.

[Myers 81] Myers, C. S., Rabiner, L. R.
"A Level Building Dynamic Time Warping Algorithm for Connected Word Recognition".
IEEE Transactions on Acoustics, Speech, and Signal Processing, Vol. ASSP-29, pp. 284-297, April, 1981.

[Nagai 91] Nagai, A., Sagayama, S., Kita, K.
"Phoneme-Context-Dependent LR Parsing Algorithms for HMM-Based Continuous Speech Recognition".
Proceedings of the European Conference on Speech Technology, pp. 1397-1400, September, 1991.

[Nakagawa 76] Nakagawa, S.
A Machine Understanding System for Spoken Japanese Sentences.
Ph. D. Thesis, Kyoto University, October, 1976.

[Nakagawa 87] Nakagawa, S.
"Spoken Sentence Recognition by Time-Synchronous Parsing Algorithm of Context-Free Grammar".
Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 829-832, April, 1987.

[Nakagawa 90] Nakagawa, S., Ooguro, Y., Murase, I.
"An Evaluation Method for Continuous Speech Recognition Systems — Relationship between Task Complexity and Sentence Recognition Accuracy—".
The Transactions of the Institute of Electronics, Information and Communication Engineers, Vol. J73-D-II, No. 5, pp. 683-693, May, 1990 (in Japanese).

[Nakamura 89] Nakamura, S., Shikano, K.
"Speaker Adaptation Applied to HMM and Neural Networks".
Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 89-92, May, 1989.

[Ney 87] Ney, H.
"Dynamic Programming Speech Recognition Using a Context-Free Grammar".
Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 69-72, April, 1987.

[Ng 91] Ng, S. K., Tomita, M.
"Probabilistic LR Parsing for General Context-Free Grammars".
Proceedings of the Second International Workshop on Parsing Technologies, pp. 154-163, February, 1991.

[Nilsson 80] Nilsson, N. J.
Principles of Artificial Intelligence.
Tioga Publishing Co., Palo Alto, 1980.

[Ozeki 87] Ozeki, K.
"A Multi-Stage Decision Algorithm to Select Optimum Kakariuke Structures from Bunsetsu Lattice".
The Transactions of the Institute of Electronics, Information and Communication Engineers, Vol. J70-D, No. 12, pp. 2621-2629, December, 1987 (in Japanese).

[Paeseler 89] Paeseler, A., Ney, H.
"Continuous-Speech Recognition Using a Stochastic Language Models".
Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 719-722, May, 1989.

[Pereira 80] Pereira, F., Warren, D.
"Definite Clause Grammars for Language Analysis — A Survey of the Formalism and a Comparison with Augmented Transition Networks".
Artificial Intelligence, Vol. 13, No. 3, pp. 231-278, May, 1980.

[Poritz 88] Poritz, A. B.
"Hidden Markov Models: A Guided Tour".
Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 7-13, April, 1988.

[Rabiner 89] Rabiner, L. R.
"A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition".
Proceedings of the IEEE, Vol. 77, No. 2, pp. 257-286, February, 1989.

[Robinson 82] Robinson, J. J.
"DIAGRAM: A Grammar for Dialogues".
Communication of the ACM, Vol. 25, No. 1, pp. 27-47, 1982.

[Sagayama 89] Sagayama, S.
"Phoneme Environment Clustering for Speech Recognition".
Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 397-400, May, 1989.

[Saito 88a] Saito, H.
"A Phoneme Lattice Parsing for Continuous Speech Recognition".
ATR Technical Report, TR-I-0033, July, 1988.

[Saito 88b] Saito, H., Tomita, M.
"Parsing Noisy Sentences".
Proceedings of the 12th International Conference on Computational Linguistics, pp. 561-566, August, 1988.

[Saito 90] Saito, H.
"Gap-Filling LR Parsing for Noisy Spoken Input: Towards Interactive Speech Recognition".
Proceedings of the International Conference on Spoken Language Processing, pp. 909-912, November, 1990.

[Sakoe 78] Sakoe, H., Chiba, S.
"Dynamic Programming Algorithm Optimization for Spoken Word Recognition".
IEEE Transactions on Acoustics, Speech, and Signal Processing, Vol. ASSP-26, pp. 43-49, February, 1978.

[Sakoe 79] Sakoe, H.
"Two-Level DP-Matching — A Dynamic Programming-Based Pattern Matching Algorithm for Connected Word Recognition".
IEEE Transactions on Acoustics, Speech, and Signal Processing, Vol. ASSP-27, pp. 588-595, December, 1979.

[Savitch 87] Savitch, W. J.
The Formal Complexity of Natural Language.
D. Reidel Publishing Company, Holland, 1987.

[Sawai 90] Sawai, H.
"The TDNN-LR Large-Vocabulary and Continuous Speech Recognition System".
Proceedings of the International Conference on Spoken Language Processing, pp. 1349-1352, November, 1990.

[Schwartz 85] Schwartz, R., Chow, Y., Kimball, O., Roucos, S., Krasner, M., Makhoul, J.
"Context-Dependent Modeling for Acoustic-Phonetic Recognition of Continuous Speech".
Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 1205-1208, March, 1985.

[Schwartz 90] Schwartz, R., Chow, Y. L.
"The N-Best Algorithm: An efficient and Exact Procedure for Finding The N Most Likely Sentence Hypotheses".
Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 81-84, April, 1990.

[Shieber 86] Shieber, S. M.
"An Introduction to Unification-Based Approaches to Grammar".
CSLI Lecture Notes, No. 4, Stanford University, 1986.

[Shikano 81] Shikano, K.
"Acoustic Processing in the Conversational Speech Recognition System".
Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 1164-1167, March-April, 1981.

[Shikano 87] Shikano, K.
"Improvement of Word Recognition Results by Trigram Model".
Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 1261-1264, April, 1987.

[Soong 90] Soong, F. K, Huang, E. F.
"A Tree-Trellis Based Fast Search for Finding the N Best Sentence Hypotheses in Continuous Speech Recognition".
Proceedings of the International Conference on Spoken Language Processing, pp. 709-712, November, 1990.

[Takeda 88] Takeda, K., Sagisaka, Y., Katagiri, S., Abe, M., Kuwabara, H.
"Speech Database User's Manual".
ATR Technical Report, TR-I-0028 or TR-A-0026, May, 1988 (in Japanese).

[Takezawa 90] Takezawa, T., Morimoto, T.
"Linguistic Knowledge for Spoken Dialogue Processing".
Proceedings of the International Conference on Spoken Language Processing, pp. 1309-1312, November, 1990.

[Takezawa 91] Takezawa, T., Kita, K., Hosaka, J., Morimoto, T.
"Linguistic Constraints for Continuous Speech Recognition in Goal-Directed Dialogue".
Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 801-804, May, 1991.

[Tomita 86] Tomita, M.
Efficient Parsing for Natural Language: A Fast Algorithm for Practical Systems.
Kluwer Academic Publishers, Boston, 1986.

[Tomita 87] Tomita, M.
"An Efficient Augmented-Context-Free Parsing Algorithm".
Computational Linguistics, Vol. 13, No. 1-2, pp. 31-46, January-June, 1987.

[Tseng 87] Tseng, H. P., Sabin, M. J., Lee, E. A.
"Fuzzy Vector Quantization Applied to Hidden Markov Modeling".
Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 641-644, April, 1987.

[Ward 89] Ward, W. H.
"Understanding Spontaneous Speech".
Proceedings of the DARPA Speech and Natural Language Workshop, pp. 137-141, February, 1989.

[Woods 70] Woods, W. A.
"Transition Network Grammars for Natural Language Analysis".
Communications of the ACM, Vol. 13, pp. 591-606, October, 1970.

[Wright 87] Wright, J. H.
"Linguistic Control in Speech Recognition".
Proceedings of the European Conference on Speech Technology, pp. 104-107, September, 1987.

[Wright 88] Wright, J. H., Wrigley, E. N.
"Linguistic Control in Speech Recognition".
Proceedings of the 7th FASE Symposium, pp. 545-552, August, 1988.

[Wright 89] Wright, J. H., Wrigley, E. N.
"Probabilistic LR Parsing for Speech Recognition".
Proceedings of the International Workshop on Parsing Technologies, pp. 105-114, August, 1989.

[Yamaoka 90] Yamaoka, T., Iida, H.
"A Method to Predict the Next Utterance Using a Four-Layered Plan Recognition Model".
Proceedings of the 9th European Conference on Artificial Intelligence, pp. 726-731, August, 1990.

[Younger 67] Younger, D. H.
"Recognition and Parsing of Context-Free Languages in Time n^3".
Information and Control, Vol. 10, No. 2, pp. 189-208, 1967.

[Zue 90] Zue, V. W., Glass, J. R., Goddeau, D., Goodine, D., Leung, H. C., McCandless, M. K., Phillips, M. S., Polifroni, J., Seneff, S., Whitney, D.
"Recent Progress on the MIT VOYAGER Spoken Language System".
Proceedings of the International Conference on Spoken Language Processing, pp. 1317-1320, November, 1990.

Index

a priori probability, 5, 45
A^* search algorithm, 95
accepted rate, 69
accuracy, 70
acoustic modeling, 2
acoustic probability, 5, 11, 45
action table, 14
allophonic model, 95
ambiguous grammar, 14
ARPA project, 1
ATN, *see* augmented transition network
augmented transition network, 94
auxiliary start symbol, 18, 51, 74

backward calculation, 12
Baum-Welch algorithm, 12
Bayes' rule, 5
beam-search, 24
bigram, 6, 46
blackboard architecture, 1
bunsetsu, 62

canonical collection, 18, 51, 74
canonical derivation, 14
canonical LR, 17
category reachability, 71
cell, 23
CFG, *see* context-free grammar
Chomsky hierarchy, 4
closure function, 17, 50, 73
Cocke-Younger-Kasami algorithm, *see* CYK algorithm, *see* CYK algorithm
codebook, 10
codebook mapping, 36
composite model, 36
conference registration task, 29
conflict, 16
context-dependent model, 95
context-free grammar, 4, 13
context-sensitive grammar, 4
CYK algorithm, 4, 16

DCG, *see* definite clause grammar
definite clause grammar, 94
deleted interpolation, 47, 52
derivation, 14
deterministic context-free grammar, 14
dialogue model, 96
discrete HMM, 10
DTW, *see* dynamic time warping
duration control, 24, 32, 34
dynamic time warping, 2

Earley's algorithm, 4, 16
entropy, 28

false alarm, 83
feature structure, 94
filled pause, 66
finite-state network, 4
follow function, 18
forward algorithm, 11
forward calculation, 11
forward-backward algorithm, 12
fuzziness, 35
fuzzy membership function, 35
fuzzy vector quantization, 35

gap-filling LR parsing, 95
general grammar, 29, 68

generalized LR parsing, 4, 16
generative power, 4
goto function, 18, 51, 73
goto table, 14
graph-structured stack, 16, 24

Hidden Markov Model, 10
HMM, *see* Hidden Markov Model
HMM-LR
    algorithm, 23
    baseline system, 31
    improved system, 33
    overview, 19

inter-phrase grammar, 63, 68
inter-phrase LR parsing table, 65
interjection, 66
intra-phrase grammar, 63, 68
intra-phrase LR parsing table, 65
    construction, 66
item, 17, 50, 73

k-nearest neighbor rule, 35
kakariuke relationship, 62
Kleene closure, 13

LALR, 17
language model
    stochastic, 5, 45
    syntactic, 3
language modeling, 2
language source model, 18
lattice, 9
lattice parsing, 33, 68
left-recursive grammar, 52
left-to-right strategy, 95
leftmost derivation, 14
legal action, 71
linguistic database, 6, 30, 53
LR grammar, 14
LR parsing, 14
    with a category reachability test, 71
LR parsing table, 14
    construction, 17, 50, 73
    inter-phrase, 65
    intra-phrase, 65
    stochastic, 50
LR-CRT, 71

maximum likelihood estimation, 12
model conversation, 40, 69
multiple codebooks, 34
multiple entry, 16

N-best search paradigm, 94
N-gram model, 6, 46
nonterminal symbol, 13

output probability, 10

PEC, *see* phoneme environment clustering
perplexity, 28
phone perplexity, 29
phoneme environment clustering, 95
phonetic context, 95
phonetic grammar, 82
phonetic typewriter, 80
phrase-structure grammar, 4
pre-terminal, 83
predictive LR parsing, 18
probability array, 19
production, 13

recursively enumerable set, 4
regular grammar, 4
relaxation technique, 95
restart, 66
rewriting rule, 13
rightmost derivation, 14

sentence accuracy, 70
sentential form, 14
separate vector quantization, 34
shared-packed forest, 16
simple LR, 17
SLR, *see* simple LR
speaker adaptation, 36
speech database, 30

spontaneous speech, 66
stack-splitting, 16
start symbol, 13
stationary phone, 31
stochastic context-free grammar, 6, 48
stochastic grammar, 6
stochastic language model, 5, 45
stochastic LR parsing, 6, 47, 50
stochastic LR parsing table, 50
    construction, 50
syntactic language model, 3

task entropy, 28
task grammar, 82
task-specific grammar, 29
terminal symbol, 13
test-set perplexity, 54
transient phone, 31
transition probability, 10
trellis diagram, 11
trigram, 6, 46
    context-free rewriting rules, 6, 52
    Japanese syllables, 6, 46
two-level LR parsing, 63
    improved, 74
type 0 grammar, 4
type 1 grammar, 4
type 2 grammar, 4
type 3 grammar, 4

unification-based formalism, 94
unification-based grammar, 94
unknown word processing, 79
unrestricted grammar, 4

vector quantization, 10
very large vocabulary, 40
Viterbi algorithm, 11
VQ, *see* vector quantization

word accuracy, 70

研究業績

＜論文＞

1. “HMM 音韻認識と拡張 LR 構文解析法を用いた連続音声認識”.
   北 研二, 川端 豪, 斎藤 博昭.
   情報処理学会論文誌, Vol. 31, No. 3, pp. 472-480, March, 1990.

2. “HMM-LR 音声認識システムの性能評価”.
   花沢 利行, 北 研二, 中村 哲, 川端 豪, 鹿野 清宏.
   日本音響学会誌, Vol. 46, No. 10, pp. 817-823, October, 1990.

3. “HMM Speech Recognition Using Stochastic Language Models”.
   Kenji Kita, Takeshi Kawabata, Toshiyuki Hanazawa.
   The Journal of the Acoustical Society of Japan, Vol. 12, No. 3, pp. 99-105, May, 1991.

4. “Continuous Speech Recognition Using Two-Level LR Parsing”.
   Kenji Kita, Toshiyuki Takezawa, Tsuyoshi Morimoto.
   IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, Vol. E 74, No. 7, pp. 1806-1810, July, 1991.

5. “Processing Unknown Words in Continuous Speech Recognition”.
   Kenji Kita, Terumasa Ehara, Tsuyoshi Morimoto.
   IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, Vol. E 74, No. 7, pp. 1811-1816, July, 1991.

＜国際会議＞

1. "HMM Continuous Speech Recognition Using Predictive LR Parsing".
Kenji Kita, Takeshi Kawabata, Hiroaki Saito.
Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 703-706, May, 1989.

2. "Parsing Continuous Speech by HMM-LR Method".
Kenji Kita, Takeshi Kawabata, Hiroaki Saito.
Proceedings of the International Workshop on Parsing Technologies, pp. 126-131, August, 1989.

3. "ATR HMM-LR Continuous Speech Recognition System".
Toshiyuki Hanazawa, Kenji Kita, Satoshi Nakamura, Takeshi Kawabata, Kiyohiro Shikano.
Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 53-56, April, 1990.

4. "HMM Continuous Speech Recognition Using Stochastic Language Models".
Kenji Kita, Takeshi Kawabata, Toshiyuki Hanazawa.
Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 581-584, April, 1990.

5. "Continuous Speech Recognition Using Two-Level LR Parsing".
Kenji Kita, Toshiyuki Takezawa, Junko Hosaka, Terumasa Ehara, Tsuyoshi Morimoto.
Proceedings of the International Conference on Spoken Language Processing, pp. 905-908, November, 1990.

6. "Processing Unknown Words in Continuous Speech Recognition".
Kenji Kita, Terumasa Ehara, Tsuyoshi Morimoto.
Proceedings of the Second International Workshop on Parsing Technologies, pp. 136-142, February, 1991.

7. "Incorporating LR Parsing into SPHINX".
Kenji Kita, Wayne H. Ward.
Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 269-272, May, 1991.

8. "Linguistic Constraints for Continuous Speech Recognition in Goal-Directed Dialogue".
Toshiyuki Takezawa, Kenji Kita, Junko Hosaka, Tsuyoshi Morimoto.
Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 801-804, May, 1991.

9. “Phoneme-Context-Dependent LR Parsing Algorithms for HMM-Based Continuous Speech Recognition”.
Akito Nagai, Shigeki Sagayama, Kenji Kita.
Proceedings of the Second European Conference on Speech Communication and Technology, pp. 1397-1400, September, 1991.

＜研究会 & シンポジウム＞

1. “HMM 音韻認識と予測 LR パーザを用いた文節認識”.
北 研二, 川端 豪, 斎藤 博昭.
電子情報通信学会音声研究会, pp. 63-69, October, 1988.

2. “Spoken Language Processing in SL-TRANS”.
Tsuyoshi Morimoto, Kentaro Ogura, Kenji Kita, Kiyoshi Kogure, Koji Kakigahara.
ATR Symposium on Basic Research for Telephone Interpretation, pp. 2·1-2·6, December, 1989.

3. “HMM-LR 音声認識システムの性能評価”.
花沢 利行, 北 研二, 中村 哲, 川端 豪, 鹿野 清宏.
電子情報通信学会音声研究会, pp. 63-70, December, 1989.

4. “HMM-LR 音声認識システムにおける統計的言語情報の利用”.
北 研二, 川端 豪, 花沢 利行.
電子情報通信学会音声研究会, pp. 1-6, January, 1990.

5. “構文規則を用いた文音声認識”.
竹澤 寿幸, 保坂 順子, 北 研二, 森元 逞, 江原 暉将.
電子情報通信学会音声研究会, pp. 41-48, December, 1990.

6. “HMM-LR 連続音声認識における音素環境依存型パーザの実現アルゴリズム”.
永井 明人, 嵯峨山 茂樹, 北 研二.
電子情報通信学会音声研究会, pp. 41-48, June, 1991.

＜一般講演＞

1. “HMM 音韻認識と予測 LR パーザを用いた文節認識”.
北 研二, 川端 豪, 斎藤 博昭.
日本音響学会秋季研究発表会, pp. 259-260, October, 1988.

2. “HMM-LR 連続音声認識システムにおける計算量削減のための一検討”.
北 研二, 川端 豪, 森元 逞.
日本音響学会春季研究発表会, pp. 77-78, March, 1989.

3. “HMM 音韻モデルの文節認識による評価”.
花沢 利行, 北 研二, 川端 豪, 鹿野 清宏.
日本音響学会春季研究発表会, pp. 81-82, March, 1989.

4. “音韻パープレキシティの提案”.
川端 豪, 鹿野 清宏, 北 研二.
日本音響学会春季研究発表会, pp. 93-94, March, 1989.

5. “音声認識候補の統計処理による絞り込み”.
坂野 俊哉, 北 研二, 森元 逞.
電子情報通信学会春季全国大会, p. 1·21, March, 1989.

6. “音声認識候補の正規化認識確率に関する考察”.
坂野 俊哉, 北 研二, 森元 逞.
情報処理学会第 39 回全国大会, pp. 571-572, October, 1989.

7. “SL-TRANS における文節音声認識 — HMM 音韻認識と LR 構文解析法による文節音声認識 —”.
北 研二, 坂野 俊哉, 保坂 順子, 川端 豪.
情報処理学会第 39 回全国大会, pp. 718-719, October, 1989.

8. “HMM-LR 音声認識システムにおける統計的言語情報の利用”.
北 研二, 川端 豪, 花沢 利行.
日本音響学会春季研究発表会, pp. 85-86, March, 1990.

9. “音声認識システムにおける確率文法の有効性”.
北 研二, 森元 逞.
情報処理学会第 41 回全国大会, pp. 2·237-2·238, September, 1990.

10. “HMM-LR 法における音素文脈依存型 LR パーザの検討”.
永井 明人, 北 研二, 嵯峨山 茂樹.
日本音響学会秋季研究発表会, pp. 125-126, September, 1990.

11. “2 段階 LR 構文解析法を用いた文認識”.
北 研二, 竹澤 寿幸, 保坂 順子, 江原 暉将, 森元 逞.
日本音響学会秋季研究発表会, pp. 127-128, September, 1990.

12. “HMM-LR 音声認識の大語彙への適用”.
北 研二, 江原 暉将, 森元 逞.
情報処理学会第 42 回全国大会, pp. 2·110-2·111, March, 1991.

13. “音素コンテキスト依存型 LR テーブルの生成アルゴリズム”.
永井 明人, 嵯峨山 茂樹, 北 研二.
日本音響学会春季研究発表会, pp. 91-92, March, 1991.

14. “連続音声認識における未知語処理”.
北 研二, 江原 暉将, 森元 逞.
日本音響学会春季研究発表会, pp. 93-94, March, 1991.

15. “HMM-LR 連続音声認識装置の開発と性能評価”.
永井 明人, 北 研二, 花沢 利行, 鈴木 忠, 岩崎 知弘, 川端 豪, 中島 邦男,
鹿野 清宏, 森元 逞, 嵯峨山 茂樹, 栂松 明.
日本音響学会秋季研究発表会, pp. 45-46, October, 1991.

16. “上位文法カテゴリへの到達可能性照合機構を備えた LR 解析法とその音声認識への応用”.
北 研二, 森元 逞, 嵯峨山 茂樹.
日本音響学会秋季研究発表会, pp. 171-172, October, 1991.

17. “文脈自由文法から音素コンテキスト依存文法への変換アルゴリズム”.
永井 明人, 菊池 英明, 嵯峨山 茂樹, 北 研二.
日本音響学会春季研究発表会, pp. 81-82, March, 1992.

18. “HMM-LR 連続音声認識における A^* アルゴリズムを用いた探索手法の検討”.
山口 耕市, 嵯峨山 茂樹, 北 研二, Frank K. Soong.
日本音響学会春季研究発表会, pp. 87-88, March, 1992.

＜書籍＞

1. “ATR HMM-LR Continuous Speech Recognition System”.
Toshiyuki Hanazawa, Kenji Kita, Satoshi Nakamura, Takeshi Kawabata, Kiyohiro Shikano.
Readings in Speech Recognition, Waibel, A. and Lee, K. F. Eds., Morgan Kaufmann Publishers, pp. 611-614, 1990.

2. “GLR Parsing in Hidden Markov Model”.
Kenji Kita, Takeshi Kawabata, Hiroaki Saito.
Generalized LR Parsing, Tomita, M. Ed., Kluwer Academic Publishers, pp. 153-164, 1991.

< その他 (本研究以外)>

1. “直観主義的解析学の諸原理”.
   北 研二.
   科学基礎論研究, Vol. 16, Nos. 1,2, pp. 69-75, January, 1983.

2. “自然言語理解システム ELLE(1) — 理解モデルについて —”.
   安原 宏, 小松 英二, 北 研二.
   情報処理学会第 29 回全国大会, pp. 1193-1194, October, 1984.

3. “自然言語理解システム ELLE(2) — 解析部の概要 —”.
   小松 英二, 安原 宏, 北 研二.
   情報処理学会第 29 回全国大会, pp. 1195-1196, October, 1984.

4. “自然言語理解システム ELLE(3) — インプリメンテーション —”.
   北 研二, 安原 宏, 小松 英二.
   情報処理学会第 29 回全国大会, pp. 1197-1198, October, 1984.

5. “自然言語理解システム ELLE における動詞の意味表現について”.
   安原 宏, 小松 英二, 北 研二, 山本 由紀雄, 加藤 安彦.
   情報処理学会自然言語処理研究会, pp. 33-38, January, 1985.

6. “自然言語理解システム ELLE における言語理解方式”.
   小松 英二, 北 研二, 山本 由紀雄, 安原 宏.
   情報処理学会第 30 回全国大会, pp. 1345-1346, March, 1985.

7. “自然言語理解システム ELLE の自然言語インタフェースへの適用”.
   山本 由紀雄, 加藤 安彦, 北 研二, 小松 英二, 安原 宏.
   情報処理学会第 30 回全国大会, pp. 1347-1348, March, 1985.

8. “自然言語インタフェースにおける柔軟な入力文対応について”.
   加藤 安彦, 北 研二, 安原 宏.
   情報処理学会第 31 回全国大会, pp. 1195-1196, October, 1985.

9. “自然言語によるデータ検索システム ELLE/TG”.
   北 研二, 加藤 安彦, 安原 宏.
   情報処理学会第 32 回全国大会, pp. 1265-1266, March, 1986.

10. “要約支援システム COGITO — 言語解析部 —”.
    北 研二, 安原 宏.
    情報処理学会第 33 回全国大会, pp. 1743-1744, October, 1986.

11. “要約支援システム COGITO”.
    北 研二, 小松 英二, 安原 宏.
    情報処理学会自然言語処理研究会, pp. 1-8, November, 1986.

12. “テキスト・データベースからの慣用表現の自動抽出”.
北 研二, 森元 逞.
情報処理学会第 37 回全国大会, pp. 1032-1033, September, 1988.

13. “慣用表現を利用した形態素情報収集法”.
小倉 健太郎, 北 研二, 森元 逞.
情報処理学会第 40 回全国大会, pp. 488-489, March, 1990.