Large vocabulary Speech Recognition System: SPOJUS++

YASUHISA FUJII, KAZUMASA YAMAMOTO, SEIICHI NAKAGAWA
Toyohashi University of Technology
Department of Computer Science and Engineering
1-1 Hibarigaoka, Tenpaku-cho, Toyohashi, Aichi 441-8580
JAPAN
{fujii, kyama, nakagawa}@slp.cs.tut.ac.jp

Abstract: In this paper, we describe Large vocabulary Continuous Speech Recognition (LVCSR) system SPOJUS++ which has been developed in our laboratory for over 20 years and recently fully reimplemented from scratch. SPOJUS++ employs a context-dependent Hidden Markov Model (HMM) as an acoustic model and an N-gram model as a language model to decode speech. SPOJUS++ has many novel features including a dynamic expansion of linear dictionary, a use of likelihood index for efficient handling of the inter-word dependency and one pass decoding. SPOJUS++ can decode speech in real time on current standard PC without painful degradation of the performance even though it uses a context-dependent HMM and a high order N-gram language model (N >= 3). Also, SPOJUS++ can construct a confusion network which leads to word error rate minimization recognition. Constructed confusion networks can be used in many kinds of post-processing applications which require automatic speech recognition results. We evaluated SPOJUS++ in terms of word accuracy, real time factor and search error. Experimental results showed that SPOJUS++ is comparable to state-of-the-arts.

Key–Words: Automatic speech recognition, LVCSR, decoder

1 Introduction

Large Vocabulary Continuous Speech Recognition (LVCSR) is quite difficult because which should deal with large vocabulary which has sometimes over 1 million words and continuous speech where word boundaries are not known, nevertheless, current research efforts for achieving LVCSR had made it possible [1, 2].

In our laboratory, we also have been developed LVCSR system called SPOJUS (SPOntaneous Japanese Understanding System) for over 20 years [3, 4, 5]. Originally, although it was written in C language, recently we have rewritten it in C++ language from scratch. We named the new LVCSR system SPOJUS++.

SPOJUS++ is a one pass decoder which employs a context-dependent Hidden Markov Model (HMM) as an acoustic model and an N-gram language model as a language model. Novel features of SPOJUS++ including a dynamic expansion of linear dictionary, a use of likelihood index for efficient handling of the inter-word dependency and one pass decoding are described in this paper. SPOJUS++ can decode speech in real time on current standard PC without painful degradation of the performance even though it uses a context-dependent HMM and constraints with a high order N-gram language model (N >= 3). Also, SPOJUS++ can construct a confusion network which leads to word error rate minimization recognition. Constructed confusion networks can be used in many kinds of post-processing applications which require automatic speech recognition results.

We evaluated SPOJUS++ in terms of word accuracy, real time factor and search error on serveral Japanse corpora. These experimental results showed that SPOJUS++ is comparable to other state-of-the-art decoders.

This paper is organized as follows. In the next section, basic algorithms used in SPOJUS++ is provided. Novel features of SPOJUS++, namely, dynamic expansion of linear dictionary, decoding using likelihood index and one pass decoding with high order N-gram language model are described in Sections 3, 4, 5, respectively. In Section 6, a two pass decoding scheme employed for SPOJUS++ is discussed. In Section 7, a construction method of confusion network [6] from word trellis instead of word lattice is explained. Experimental results are presented in Section 8. Finally, this paper is summarized in Section 9.

2 Basic Algorithms

SPOJUS++ is a one pass decoder which employs a HMM as an acoustic model and an N-gram model as a

language model. Suppose O ($|O| = N$) represents an acoustic observation sequence and $P_W(i)$ represents the score of the partial hypothesis W ending at time i. LVCSR using an N-gram language model can be formulated as follows:

$$P_{W'=Ww}(i) = \max_{w} P_{W,w}(i), \quad (1)$$

$$P_{W,w}(i) = \max_{j<i}\{P_W(j) + Q_w(j,i) + L(w|W)\}, \quad (2)$$

where $P_{W,w}(i)$ means a score of a hypothesis Ww ending at time i which was resulted in by connecting the word w to the partial hypothesis W, $Q_w(j,i)$ means an acousic score when the word w was expanded at time j and ending at time i, and $L(w|W)$ is a language score of word w when connecting to W. The number of allowable combination of words in the optimization problem of Eq. (1) explodes exponentially as time increases, therefore, we need to solve the problem with accurate approximation and pruning.

SPOJUS++ expresses a word lexicon by tree structure and expands only the best hypothesis at each frame by assuming 1-best approximation as described in Section 3.2. When using a tree lexicon, we cannot apply language score to a hypothesis until the hypothesis reaches leaves of the tree. Before applying the true language score at a leaf of the tree, uni-gram language score is assigned to the hypothesis. Uni-gram score of each word is factored from the leaves of the tree to the root node. The uni-gram score is replaced with the true N-gram score when hypothesis reaches a leaf node.

Pruning is conducted based on the number of active nodes, the score difference between a hypothesis and the best hypothesis and the number of words ending at each frame. At each frame, only the predifined number of hypotheses are retained. The hypotheses whose scores have larger difference compared to the score of the best hypothesis are pruned. The predifined number of words are allowed to connect and examin the score when backtrace is conducted.

To speed up the decoding, SPOJUS++ also uses language model score cache by assuming that hypotheses which reach a word node (a leaf in the tree lexicon) are same during several frames.

In addition to these basic algorithms which are often used in other LVCSR systems, SPOJUS++ has several novel features which are described in the following sections.

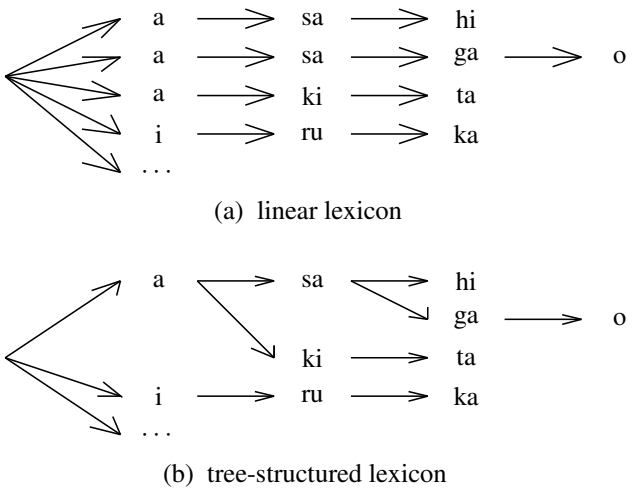


Figure 1: Linear lexicon and tree-structured lexicon.

3 Dynamic Expansion of Linear dictionary [4]

3.1 Linear lexicon and tree-structured lexicon

HMM-based LVCSR takes more computational cost when the vocabulary size becomes larger. When using a linear lexicon illustrated in Fig. 1 (a), the number of expanded HMM states increases proportionally to the vocabulary size and then the computational costs is also proportional to the vocabulary size. The tree-structured lexicon illustrated in Fig. 1 (b) is conventionally used to reduce the number of expanded HMM states.

3.2 Recognition using tree-structured lexicon and 1-best approximations of word history dependency

The likelihood and boundary of a word depend on the word history. So we have to execute the forward decoding procedure independently for all new hypotheses generated by concatenating a word to all the possible hypotheses. Since many hypotheses are generated in this process, we have to make an assumption on the word history dependency and bundle the hypotheses using an approximation based on the assumption.

Compared to other methods such as word-pair approximation [7] which assumes that the word boundaries depend on just one previous word, the 1-best approximation [8] does not assume any word-history dependencies. This approximation reduces the computational cost dramatically because the algorithm does not need copies of the HMM network and uses only

ISBN: 978-960-474-283-7

one reentrant network. However, this approximation is not exact. The HMM network is shared by the hypotheses expanded at all the time frame and thus many paths are overrided and wiped off by the other path. So the path which is optimal when considering the language score which can be applied at the leaf node may be rejected by this mechanism. In this way, the optimal word sequence is not guaranteed to be obtained by tree-structured lexicon search under the 1-best approximation.

3.3 Recognition using linear lexicon

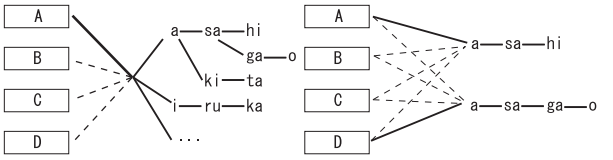
Linear lexicons for vocabulary words are connected each other from the tails to the heads in an HMM network. Using this network, any sub-word HMMs are not shared among words unlike the tree lexicon. This enables the system to apply language scores at the first node of each word whereas the tree-structured lexicon does not. This means that the system can use language look-ahead, resulting in efficient constraints of the search space. Also, because there are no conflicts among words, the first best hypothesis is guaranteed to be obtained even though the HMM nodes in linear lexicons are also shared among histories and thus the optimal N-best hypotheses are not guaranteed to be obtained.

Fig. 2 (a) shows an example of the new word attachment to hypotheses using a linear lexicon. The word "a-sa-ga-o" is attached to the hypothesis D which maximizes the sum of the total score of hypothesis D and the language score P(a-sa-ga-o|D). On the contrary, the HMM network is attached to the hypothesis A in the case of a tree-structured lexicon as illustrated in Fig. 2 (b) because of its highest likelihood among the hypotheses not considering the word probabilities when given the histories.

Decoding using a linear lexicon is more accurate than using a tree lexicon, however, the computational cost needed is much higher than that for a tree-structured lexicon.

3.4 Combinational use of dynamically expanded linear lexicon and static tree-structured lexicon

SPOJUS++ employs a new search method using search on dynamically expanded linear lexicon along with the 1-best approximation search on a tree-structured lexicon [4]. That is, we basically used the 1-best approximation tree-structured lexicon search and also used the linear lexicon search in parallel. Only the small number of words N_{lin}, which were dynamically selected, are expanded in a linear lexicon



(a) Tree-structured lexicon expansion with 1-best approximation

(b) Linear lexicon expansion

Figure 2: Word attachment with partial sentence hypothesis (assuming $P(\mathrm{A}) > P(\mathrm{B})$, $P(\mathrm{C}), P(\mathrm{D})$ and P(a-sa-ga-o|D) $\gg P$(a-sa-ga-o|A)).

network.

In our dynamic selection of the words to be expanded in a linear network, we evaluated all the vocabulary words at every time frame using the following equations:

$$Q_{tree}(w,t) = \max_{v}(P_{hyp}(v,t) + L(w|v)), \quad (3)$$

$$Q_{lin}(w,t) = \max_{s_w}(P_{state}(s_w,t)), \quad (4)$$

where $P_{hyp}(v,t)$ is the likelihood of the final HMM state of the hypothesis which ends with the word v, $P_{state}(s_w,t)$ is the likelihood of the HMM state s_w in word w which is expanded on a linear HMM network at the time frame t and $L(w|v)$ is a look-ahead probability which is the occurrance probability of word w given a word history v. $Q_{tree}(w,t)$ in Equation (3) means the maximum value among the sums of likelihood of preceding word sequence ended by word v in a tree-structured network and the look-ahead probability. $Q_{lin}(w,t)$ in Equation (4), which means the maximum value among the HMM states of the word w, is calculated for the word w which is already expanded in the linear HMM network. If a word w corresponds to both $Q_{tree}(w,t)$ and $Q_{lin}(w,t)$, then the maximum value of $Q_{tree}(w,t)$ and $Q_{lin}(w,t)$ is selected as the evaluation value for the word.

The N_{lin}-best words are selected based on the scores computed by Eqs. (3) and (4). The words selected and not expanded on the linear network yet will be expanded. The words selected and already expanded on the network will be kept on the network. The words not selected on the linear network will be removed from the network. Using the linear lexicon, the optimal path lost by the 1-best approximation search on the tree-structured lexicon will be likely to be kept. On the other hand, the paths with low linguistic probability will be pruned out from the linear network without acoustic evaluation because of the language look-ahead, however, such paths also have chance to be kept on the tree-structured network.

ISBN: 978-960-474-283-7

4 Inter-Word Context-Dependent Modeling using Likelihood Index [9]

Because Japanese syllables consist of a consonant and vowel pair, there are relatively small number (approx. 120) of syllables and we only need to consider small variations (approx. 7 or 8 including vowels and pause) of left context. So we can use full syllable modeling considering C-V context dependency and also easily extend the Context Independent HMM (CIHMM)s to full left Context Dependent HMM (CDHMM)s to consider V-C context dependency.

As well known, intra-word context-dependent modeling can be realized by describing the CD syllables in the dictionary. However, inter-word modeling should be realized by recognition process[1].

SPOJUS++ deals with the context dependency as illustrated in Fig. 3. As described in Fig. 3 (a), we only need to make branches for the head syllable according to the contexts and the paths are merged at the second syllable for the linear lexicon. For the tree-structured lexicon, branches are made in a similar way. At the end node of a word, the language scores have to be compensated considering the inter-word context, however, the scores of contexts other than that of the best history are lost because of the merge at the second syllable. To solve this problem, we introduce the *'likelihood difference index'* (Fig. 3(b)). The likelihood differences between phonetic contexts are calculated at the merging points and the difference index is inherited to the succeeding nodes. For example, Fig. 3(b) shows that *i-a* ("a" with the left context "i") after 1st syllable matching is the best. Using this index, language scores are accurately compensated considering inter-word phonetic contexts at the end node of the words.

Even though this handling of conext dependency is accurate only for left context dependent models such as ours, this algorithm can be used to deal with tri-phone model by just ignoring inter-word right context and considering only inter-word left context of tri-phone. This is not accurate for one pass decoding, however, we can use a two pass recognition scheme described in Section 6 in such a case.

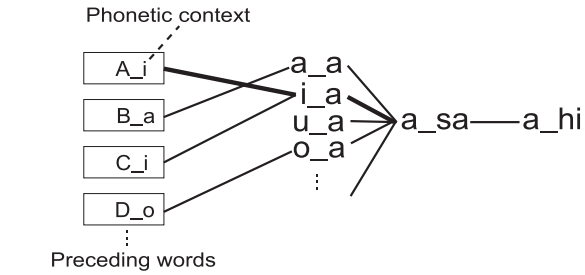
5 One Pass N-gram decoding [5]

Similar with using bigram language models, we expand bigram language models to trigram language models and 4-gram language models (It can be expanded to arbitrary N-gram). If we adopt trigram language models and 4-gram language models, Eq. (2) would turn into Eqs. (5) and (6), respectively:

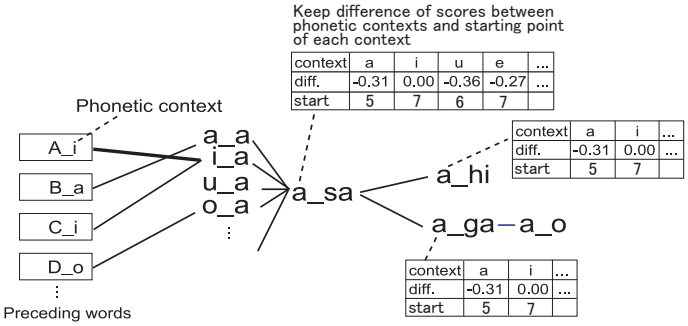
$$Q_{la}(w,t) = \max_{v}(P_{hyp}(v,t) + L(w|u,v)), \quad (5)$$

$$Q_{la}(w,t) = \max_{v}(P_{hyp}(v,t) + L(w|s,u,v)), \quad (6)$$

where $L(w|u,v)$ and $L(w|s,u,v)$ are look-ahead probabilities which are the occurrance probabilities of word w given word history v, u and v, u, s, respectively. When adopting bigram language models, the optimal hypothesis is guaranteed to be obtained using a linear lexicon. However, adopting trigram and 4-gram language models, optimal hypotheses may be lost, because optimal language score for the current word according to two or three word history can not obtain even if we use a linear lexicon. The two or three word hisotry is traced by backtracking $P_{\hat{u}v}$ and $P_{\hat{\hat{s}}\hat{u}v}$ as shown in Fig. 4.



(a) Linear lexicon



(b) Tree-structured lexicon

Figure 3: Approximate inter-word context-dependent modeling and recognition.

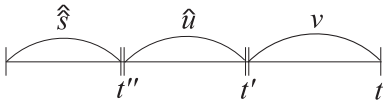


Figure 4: Backtracking of word history.

[1] Inter-word description can be prepared *a priori* as in the case of WFST based method, however such modeling is often realized by some kinds of approximations to reduce memory consumption.

6 Two pass decoding

SPOJUS++ can achive accurately one pass search by employing the algorithms explained in this paper. However, because there are still some approximations such that using 1-best approximation and assuming fixed word boundaries except adjacent words, there is no guarantee that the most plausible hypotheses are obtained. Therefore, two pass decoding is also implemented for SPOJUS++.

Two pass decoding of SPOJUS++ uses best-first stack decoding on word trellis obtained from the result of one pass search for the second pass search. To obtain forward heuristics which stack decoding require, SPOJUS++ runs a backward algorithm on the word trellis. If we adopt backward stack decoding, we can utilize scores obtained by the first pass result as backward heuristics without additional forward recursion to compute heuristics [8]. However, SPOJUS++ is a one pass decoder and uses an N-gram language model where $N > 2$ on the first pass. Hence, having backward N-gram in addition to forward N-gram is not efficient.

During the second pass stack decoding, only language model scores are recomputed. The language model used in the second pass can be changed from the one used in the first pass. To make the second pass search be efficient, hypotheses which share context are bundled and stack size is limited to a predifined number. N-best hypotheses resulted in are rescored in terms of acoustic score where the model used for rescoreing can be changed from the one used in the first pass similar to the language model.

7 Construction of Confusion Network from Trellis [10]

According to Mangu's algorithm, a confusion network is constructed from a word lattice [6]; however, SPOJUS++ does not produce a word lattice. Since it assumes a 1-best approximation in the first pass, SPOJUS++ generates a word trellis instead of a word lattice. Although a lattice could be generated by searching the trellis in the second pass, it would be redundant since we do not need a lattice, but a confusion network. Hence, we need a method to construct a confusion network from a word trellis, instead of a word lattice.

A word trellis contains information about the beginning and end points of each word stored in the trellis. Using this information, a huge lattice can be generated from the trellis by regarding words stored in the trellis as edges and the beginning and end frames of each word as the nodes, from which the confusion network can be constructed. In most cases, we need lattice pruning based on the posteriors before constructing the confusion network because the lattice usually contains a huge number of edges (typically more than 0.1 million),

8 Experiments

8.1 Setup

To evaluate the performance, we used three kinds of data sets. The first set is IPA100 test set consisting of 100 utterances by 23 speakers from Japanese newspaper read speech in JNAS corpus[2]. The second set is 4 lecture speech utterances from the Corpus of Spontaneous Japanese (CSJ) [3]. The final set is 8 lecture speech utterances from the Corpus of Japanese Lecture Class (CJLC) [11] [4]. All of speech data were sampled with a sampling frequency of 16 kHz, and the signal was pre-emphasized by a factor of 0.98. A Hamming window of 25 ms length was applied and shifted with the step of 10 ms. The 38 dimensional feature vectors were used including 12 dimensional MFCCs, their first and second deviation coefficients and the first and second deviations of log power.

For JNAS evaluation data set, 27992 utterances read by 175 male speakers (JNAS corpus) were used to train 116 Japanese context-independent syllable HMMs (strictly speaking, mora HMMs [12]) including short pause and silence. For CSJ and CJLC evaluation data set, 116 CSJ context-independent HMMs were adapted from the 116 JNAS context-independent HMMs by MAP [13, 14] using 987 lectures from the CSJ corpus uttered by 987 male speakers. Using these context-independent HMMs (CIHMMs) as base models, we also trained 928 context-dependent HMMs (CDHMMs) with 8 left contexts (5 vowels, silence, /N/, and short pause including /q/). Each continuous density HMM had 5 states, and 4 of them had pdfs of output probability. Each pdf consisted of 4 Gaussians with full-covariance matrices.

For the JNAS evaluation data set, trigram language models with a 20000 word vocabulary size were trained from the text of 75 months Mainichi Japanese newspaper. For the CSJ and CJLC evaluation data set, trigram language model with a 20000 word vacabulary size was trained using a text made by correctly transcribing lecture speech of CSJ corpus (970 lectures).

[2] http://www.mibel.cs.tsukuba.ac.jp/_090624/jnas/instruct.html

[3] We used *a01m0007*, *a01m0035*, *a01m0074* and *a05m0031*

[4] We used *L11M0011*, *L11M0012*, *L11M0031*, *L11M0032*, *L11M0041*, *L11M0042*, *L11M0051* and *L11M0052*

ISBN: 978-960-474-283-7

Table 1: Experimental condition

OS	Linux 2.6.26-2-amd64
CPU	Intel(R) Xeon(R) CPU X5365 3.00GHz
Memory	32GB
Compiler	g++ (GCC) 4.4.5
Optimization	-O3

We evaluated SPOJUS++ in terms of word accuracy, real time factor and search error. Word accuracy is computed as follows:

$$Acc. = \frac{\#Cor. - \#Sub. - \#Del. - \#Ins.}{\#Cor.} \quad (7)$$

where #Cor., #Sub., #Ins., #Del. mean the number of correct words, the number of substituted words, the number of inserted words and the number of deleted words, respectively. Real time factor is computed as follows:

$$xRT = \frac{T_{hyp}(i)}{D_{ref}(i)} \quad (8)$$

where $T_{hyp}(i)$ means time needed to decode speech data i and $D_{ref}(i)$ means the duration of speech data i. If xRT is less than 1.0, speech can be decoded in real time. The experimental condition to assess the real time factor is shown in Table 1. Search error is evaluated as follows:

$$SE = \sum_i \delta(P_{ref}(i) > P_{hyp}(i)) \quad (9)$$

where $\delta(x)$ is a function 1 if x is true, 0 otherwize and $P_{ref}(i)$ means the score for speech data i and $P_{hyp}(i)$ means the best hypothesis for speech data i. If the score of the best hypothesis system returned is lower than the score of correct hypothesis(manual transcription), we can regard it as a search error.

8.2 Results

8.2.1 Experiments on IPA100 test set

In this section, we evaluated SPOJUS++ on IPA100 test set. Using the test set, the novel algorithms implemented in SPOJUS++ except construction of confusion network from trellis was evaluated.

Evaluation of one pass algorithm Table 2 shows the word accuracies on IPA 100 test set by one pass recognition described in Section 5. In Table 2, #Node and #Beams are pruning parameters and they means the number of active nodes which are retained at each frame and #Beam means the number of hypotheses which allowed to expand at each frame, respectively. #Linear means the number of words expanded as the linear lexicon. We can see that the word accuracies increase as pruning parameters are relaxed. The combinational use of dynamically expanded linear lexicon and static tree-structured lexicon was effective especially when pruning parameters were tight.

Table 2: One pass recognition results on IPA 100 test set [%].

#Node	#Beam	#Linear			
		0	100	250	500
1000	5	90.3	91.6	91.7	91.6
1000	10	91.8	92.5	92.6	92.3
1000	20	92.1	92.3	92.4	92.4
1000	30	92.2	92.3	92.5	92.5
1500	5	90.8	91.7	91.8	91.7
1500	10	92.4	92.5	92.5	92.5
1500	20	92.6	92.4	92.5	92.5
1500	30	92.5	92.5	92.6	92.6
3000	5	90.6	91.6	91.6	91.5
3000	10	92.3	92.5	92.4	92.4
3000	20	92.6	92.5	92.5	92.5
3000	30	92.7	92.5	92.5	92.5
5000	5	90.8	91.9	91.9	91.9
5000	10	92.6	92.9	92.7	92.7
5000	20	92.9	92.8	92.8	92.8
5000	30	93.0	92.9	92.9	92.9

Evaluation of two pass algorithm Table 3 shows the word accuracies on IPA 100 test set by two pass recognition described in Section 6. Two pass recognition results were slightly superior to one pass recognition results shown in Table 2. Therefore, the one pass algorithm described in Section 5 was not perfect. However, since the differences were not so large, our one pass algorithm would be effective when we need recognition results as fast as possible.

Evaluation of search error and real time factor Tables 4 and 5 show the search error and real time factor on sevral conditions, respectively. From the tables, we can see that pruning parameters and search error is trade-off and two pass recognition consitently decrease the search error compared to one pass recognition. SPOJUS++ can recognize speech in real time by tuning pruning parameters appropriately.

Evaluation of dealing with tri-phone SPOJUS++ is tuned to deal with left context dependency and not tuned to deal with right context which is needed for dealing with tri-phone. However, as described in Section 4, SPOJUS++ can use a tri-phone model by just

Table 5: Real time factor on IPA 100 test set (one pass / two pass) [%].

#Node	#Beam	#Linear			
		0	100	250	500
1000	5	0.39 / 2.01	1.80 / 3.50	1.32 / 2.86	1.37 / 2.92
1000	10	0.57 / 2.19	1.62 / 3.20	1.64 / 3.21	1.73 / 3.30
1000	20	0.69 / 2.35	2.00 / 3.65	2.04 / 3.69	2.18 / 3.80
1000	30	0.80 / 2.54	2.33 / 4.06	2.39 / 4.10	2.54 / 4.29
1500	5	0.65 / 2.26	1.49 / 3.07	1.50 / 3.06	1.55 / 3.11
1500	10	0.75 / 2.39	1.81 / 3.42	1.85 / 3.46	1.93 / 3.53
1500	20	0.88 / 2.53	2.40 / 4.08	2.47 / 4.18	2.62 / 4.32
1500	30	1.04 / 2.83	2.52 / 4.28	2.84 / 4.62	2.85 / 4.62
3000	5	1.05 / 2.63	1.87 / 3.45	1.91 / 3.47	1.95 / 3.50
3000	10	1.15 / 2.76	2.37 / 3.99	2.25 / 3.87	2.59 / 4.25
3000	20	1.33 / 3.04	2.64 / 4.32	2.69 / 4.38	3.20 / 4.92
3000	30	1.68 / 3.52	3.25 / 5.03	3.33 / 5.11	3.19 / 4.99
5000	5	1.51 / 3.06	2.33 / 3.92	2.35 / 3.91	2.43 / 4.00
5000	10	1.65 / 3.28	2.69 / 4.30	2.74 / 4.38	3.10 / 4.72
5000	20	1.81 / 3.51	3.11 / 4.81	3.17 / 4.83	3.27 / 4.95
5000	30	2.01 / 3.81	3.47 / 5.29	3.55 / 5.34	3.69 / 5.48

Table 3: Two pass recognition results on IPA 100 test set [%].

#Node	#Beam	#Linear			
		0	100	250	500
1000	5	90.9	92.2	92.2	92.0
1000	10	92.5	93.0	93.2	93.0
1000	20	92.6	92.9	93.0	93.0
1000	30	92.7	93.0	93.1	93.1
1500	5	91.4	92.3	92.3	92.2
1500	10	93.2	93.2	93.2	93.2
1500	20	93.2	93.1	93.2	93.2
1500	30	93.1	93.2	93.3	93.3
3000	5	91.2	92.0	91.9	91.8
3000	10	93.0	93.2	93.1	93.1
3000	20	93.2	93.1	93.1	93.1
3000	30	93.2	93.2	93.2	93.2
5000	5	91.4	92.3	92.2	92.2
5000	10	93.4	93.5	93.4	93.4
5000	20	93.5	93.4	93.4	93.4
5000	30	93.6	93.5	93.5	93.5

Table 4: Search error on IPA 100 test set (one pass / two pass)[%].

#Node	#Beam	#Linear			
		0	100	250	500
1000	5	8 / 7	5 / 4	4 / 3	4 / 3
1000	10	5 / 3	3 / 2	2 / 1	2 / 1
1000	20	4 / 2	3 / 2	2 / 1	2 / 1
1000	30	4 / 2	3 / 2	2 / 1	2 / 1
1500	5	8 / 7	5 / 4	4 / 3	4 / 3
1500	10	5 / 3	3 / 2	2 / 1	2 / 1
1500	20	4 / 2	3 / 2	2 / 1	2 / 1
1500	30	4 / 2	3 / 2	2 / 1	2 / 1
3000	5	8 / 7	5 / 4	5 / 4	5 / 4
3000	10	4 / 2	2 / 1	2 / 1	2 / 1
3000	20	3 / 1	2 / 1	2 / 1	2 / 1
3000	30	3 / 1	2 / 1	2 / 1	2 / 1
5000	5	7 / 6	4 / 3	4 / 3	4 / 3
5000	10	3 / 1	1 / 0	1 / 0	1 / 0
5000	20	2 / 0	1 / 0	1 / 0	1 / 0
5000	30	2 / 0	1 / 0	1 / 0	1 / 0

ignoring right context in the first pass. Although this one pass recognition is not exact, SPOJUS++ can conduct two pass recognition to improve recognition results.

Table 6 shows the recognition results when using tri-phone as an acoustic model in SPOJUS++. We used 2000 states PTM tri-phones for the evaluation. Each tri-phone consisted of 3 states which had 16 Gaussians with diagonal covariance matrices. The table shows that one pass algorithm employed in SPOJUS++ is not accurate when using tri-phone, however, two pass recognition could improve the recognition results. For comparison, the recognition results by open source state-of-the-art decoder Julius[5] are also

[5]Julius Version 4.1.4

ISBN: 978-960-474-283-7

Table 7: Recognition results on CSJ and CJLC [%].

Corpus	method	#Linear	Acc.
CSJ	MAP(1pass)	0	70.2
	MAP(1pass)	250	70.6
	MAP(2pass)	0	70.9
	MAP(2pass)	250	71.4
	ConfNet	0	71.0
	ConfNet	250	71.4
CJLC	MAP(1pass)	0	51.9
	MAP(1pass)	250	54.4
	MAP(2pass)	0	52.2
	MAP(2pass)	250	55.5
	ConfNet	0	53.6
	ConfNet	250	56.8

listed in Table 6. We used the same acoustic model and language model with SPOJUS++ and tuned Julius to achive the highest accuracy. From the result, we can say that the two pass recognition of SPOJUS++ is comparable to the result of Julius in terms of word accuracy[6]. Also, we can say that recognition result in Tables 2 and 3 are comparable to the result of the state-of-the-art recognizer.

8.2.2 Experiments on CSJ and CJLC

In this section, we evaluated SPOJUS++ on CSJ and CJLC test sets. Using the test sets, the consensus decoding [6] by constructing confusion network from trellis was evaluated. Table 7 shows recognition results on CSJ and CJLC when using Maximum a Posteriori decoding (MAP) or consensus decoding (ConfNet). The table shows that consensus decoding is superior to the one pass decoding and is at least comparable to the two pass decoding. This result shows that the effectiveness of consensus decoding by constructing confusion network from trellis. Because a confusion network can provide multiple hypotheses with word posterior information, it is suitable for the intermediate representation of ASR results [15].

9 Conclusion

In this paper, we described the LVCSR decoder SPOJUS++. We evaluated the decoder in terms of word accuracy, search error and real time factor. Experimental results showed that SPOJUS++ was comparable to state-of-the-arts. In the future, since we evaluated SPOJUS++ only on Japanese corpora in this paper, we want to examine SPOJUS++ on corpora of foreign languages. Recently, SPOJUS++ was applied to English recognition successfully [16]. Therefore, we expect SPOJUS++ can be used broadly for several languages. Also, we want to explore utilizing tied-state left context dependent HMM as similar to PTM tri-phones to make the decoding of SPOJUS++ be more efficient while preserving the accuracy.

[6]Even though parameters were set to achive the highest accuracy, the real time factor of Julius was 0.45. Julius achived the high accuracy with the high speed. Of course Julius is the state-of-the-art decoder, however, the algorithm employed by Julius such as the backward search on the second pass (i.e., backward trigram) would be especially effective for Japanese recognition.

Acknowledgements: This work was partially supported by Global COE Program" Frontiers of Intelligent Sensing "from the Ministry of Education,Culture,Sports, Science and Technology, Japan.

References:

[1] L. Nguyen, X. Guo, R. Schwartz, and J. Makhoul. Japanese broadcast news transcription. In *Proc. ICSLP*, pages 1749 – 1752, 2002.

[2] T. Hori, C. Hori, Y. Minami, and A. Nakamura. Efficient wfst-based one-pass decoding with on-the-fly hypothesis rescoring in extremely large vocabulary continuous speech recognition. *IEEE Transactions on Audio, Speech and Language Processing*, 15:1352 – 1365, 2007.

[3] S. Nakagawa, Y. Ohguro, and Y. Hashimoto. The Syntax-Oriented Speech Understanding System -SPOJUS-SYNO-. In *Proc. Eurospeech*, volume 2, pages 224 – 227, Sep. 1989.

[4] N. Kitaoka, N. Takahashi, and S. Nakagawa. Large vocabulary continuous speech recognition using linear lexicon search and 1-best approximation tree-structured lexicon search. *Systems and Computers in Japan*, 36(7):31–39, 2005.

[5] N. Kitaoka, Y. Liang, N. Takahashi, and S. Nakagawa. One-pass LVCSR algorithm using linear lexicon search and 1-best approximation tree-structured lexicon search. In *Proc. 9-th ISSPA*, pages 1–4, 2 2007.

[6] L. Mangu, E. Brill, and A. Stolcke. Finding consensus in speech recognition: word error minimization and other applications of confusion networkds. *Computer Speech and Language*, 14(4):373–400, 2000.

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

[8] T. Kawahara, T. Kobayashi, K. Takeda, N. Minematsu, K. Ito, M. Yamamoto, A. Yamada, T. Utsuro, and K. Shikano. Sharable software repository for Japanese large vocabulary continuous

Table 6: Recognition results on IPA 100 test set when using tri-phone.

Algorithm	condition	Acc. [%]
one pass	#Node=1500, #Beam=15, #Linear=0	88.8
two pass	#Node=1500, #Beam=15, #Linear=0	90.6
one pass	#Node=5000, #Beam=30, #Linear=0	90.2
two pass	#Node=5000, #Beam=30, #Linear=0	93.2
one pass	#Node=10000, #Beam=30, #Linear=0	90.8
two pass	#Node=10000, #Beam=30, #Linear=0	93.3
one pass	#Node=10000, #Beam=30, #Linear=250	91.1
two pass	#Node=10000, #Beam=30, #Linear=250	93.9
Julius	accurate mode	93.6

speech recognition. In *Proc. ICSLP-98*, pages 3257 – 3260.

[9] J. Zhang, L. Wang, and S. Nakagawa. LVCSR based on context dependent syllable acoustic models. In *Asian Workshop on Speech Science and Technology, SP2007-200*, pages 81–86, Mar. 2008.

[10] Y. Fujii, K. Yamamoto, and S. Nakagawa. Improving the readability of class lecture asr results using multiple hypotheses. In *SPPRA 2010*, pages 678–091, Feb. 2010.

[11] M. Tsuchiya, S. Kogure, H. Nishizaki, K. Ohta, and S. Nakagawa. Developing Corpus of Japanese Classroom Lecture Speech Contents. In *Proc. LREC*, Jun.

[12] S. Nakagawa, K. Hanai, K. Yamamoto, and N. Minematsu. Comparison of syllable-based hmms and triphone-based hmms in japanese speech recognition. In *Proc. International Workshop on Automatic Speech Recognition and Understanding*, pages 393–396, 1999.

[13] J. Gauvain and C. Lee. Maximum a posteriori estimation for multivariate gaussian mixture observations of markov chains. *IEEE Trans. Acoust. Speech Signal Processing.*, 2(2):291 – 298, 1994.

[14] Y. Tsurumi and S. Nakagawa. An unsupervised speaker adaptation method for continuous parameter hmm by maximum a posteriori probability estimation. In *Proc. ICSLP*, pages 431 – 434, 1994.

[15] Y. Fujii, K. Yamamoto, and S. Nakagawa. Improving the readability of class lecture asr results using a confusion network. In *Proc. Interspeech*, pages 3078 – 3081, Sep. 2010.

[16] W. Naptali, M. Tsuchiya, and S. Nakagawa. Topic dependent class based language model evaluation on automatic speech recognition. In *IEEE Workshop on Spoken Language Technology (SLT 2010)*, pages 383 – 388, Dec. 2010.