Evaluation of Tree-trellis based Decoding in Over-million LVCSR

Naoaki Ito, Yoshihiko Nankaku, Akinobu Lee, Keiichi Tokuda

Department of Scientific and Engineering Simulation,
Nagoya Institute of Technology, Nagoya, Japan
{itonao,nankaku,ri,tokuda}@sp.nitech.ac.jp

Abstract

Very large vocabulary continuous speech recognition (CSR) that can recognize every sentence is one of important goals in speech recognition. Several attempts have been made to achieve very large vocabulary CSR. However, very large vocabulary CSR using a tree-trellis based decoder has not been reported. We report the performance evaluation and improvement of the "Julius" tree-trellis based decoder in large vocabulary CSR (LVCSR) involving more than one million vocabulary, referred to here as over-million LVCSR. Experiments indicated that Julius achieved a word accuracy of about 91% and a real time factor of about 2 in over-million LVCSR for Japanese newspaper speech transcription.

Index Terms: Speech recognition, Tree-trellis search algorithm, LVCSR, Google N-gram

1. Introduction

The ultimate goal in speech recognition is to recognize every sentence without limiting the vocabulary. Huge language databases such as the Web have recently made it possible to build very large vocabulary language models. However, very large vocabulary continuous speech recognition (CSR) may require huge amount of probability calculation and the number of search errors may increase due to approximation errors. Therefore, an efficient algorithm tailored for very large vocabulary CSR should be studied. Several attempts have been made to achieve very large vocabulary CSR [1]. Several commercial very large vocabulary CSR system is already running, for example, Google Voice Search [2]. However, this system focus on short search queries, not long sentences. On the other hand, a tree-trellis based decoder has been developed for years [4, 5]. However, no reports have been published on very large vocabulary CSR with a tree-trellis based decoder. Thus, it's performance for such very large vocabulary tasks should be examined.

We evaluate the performance of a tree-trellis based decoder "Julius" in over-million LVCSR and investigate an improvement the search algorithm for very large vocabulary tasks. Since the number of word combinations increases in very large vocabulary tasks, the problem of decoding arises at word-level computing, especially when handling word contexts in a tree-trellis based decoder. Word-pair approximation [3] is common method for handling word contexts. However, this method requires a huge amount of computation in very large vocabulary tasks. An approach is taken to use light-weight one-best approximation with error recovery. The errors that occur in word context are recalculated and corrected at the word-end. This method is thus expected to minimize accuracy reduction and achieve more efficient searches.

The rest of this paper is organized as follows. Section 2 briefly describes decoding algorithm of a tree-trellis based decoder from the viewpoint of very large vocabulary CSR, and Section 3 describes the delayed LM rescoring method. The experimental conditions are explained and the results are analyzed in Section 4. Section 5 is the conclusion.

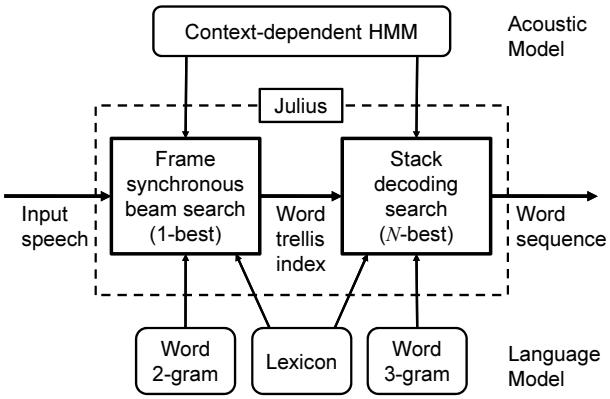

Figure 1: Overview of Julius.

2. Decoding algorithm

A tree-trellis search algorithm is often used in large vocabulary continuous speech recognition (LVCSR) [4]. "Julius" an LVCSR engine based on the two-pass tree-trellis search algorithm [5] was used. An overview of the algorithm of Julius is illustrated in Fig. 1. On the first pass, a tree-structured lexicon assigned with a language model constraint was applied with a frame-synchronous beam search. Various approximations, such as one-best approximation and 1-gram factoring, can be adopted to increase the search speed. A word trellis index is generated at the end of the first pass. On the final pass, a full language model and cross-word context dependency is applied for rescoring by referring to the result of the first pass.

The following problems are assumed to occur in very large vocabulary CSR with a tree-trellis based decoder.

- Increased amount of approximation errors, especially word-level approximation
- Increased beam width in order to keep the maximum likelihood hypothesis
- Increased memory usage due to large language model

If the approximation errors of the preliminary pass increase with this algorithm, the result of the final pass will be adversely affected.

The problem of decoding arises at word-level computation since word combinations increase in very large vocabulary

tasks. Therefore, approximation methods related to word-level computation are briefly described from the view of very large vocabulary tasks.

2.1. LM factoring

For speech recognition using a tree-structured lexicon, a word cannot be identified until the word end. Therefore, LM factoring, which applies language probabilities ahead of schedule, is used. If 2-gram probabilities are applied, precise scores are calculated; however, the amount of calculation becomes too large for very large vocabulary tasks since it roughly increases in proportion to the vocabulary. 1-gram factoring applies 1-gram probabilities at halfway nodes of a tree-structured lexicon and 2-gram probabilities when the word is identified. 1-gram factoring is fast since language probabilities can be applied statically to halfway nodes in advance. However, since these probabilities are 1-gram probabilities that ignore word contexts, pruning errors increase when the number of nodes increases in very large vocabulary CSR.

2.2. Word context dependency approximations

2.2.1. Word-pair approximation

Discrete passes need to be calculated at each word context to obtain precise Viterbi passes since word boundaries depend on word contexts. However, it is difficult to calculate precise Viterbi passes since the number of word contexts greatly increases in LVCSR. Therefore, word context dependency approximation, for example word-pair approximation [3] and N-best word-pair approximation, is generally adopted to reduce the amount of calculation. Word-pair approximation treats discrete passes at each last word. An overview of word-pair approximation is illustrated in Fig. 2.

In word-pair approximation, reasonable Viterbi passes and 2-gram probabilities can be calculated. However, a tree-structured lexicon is copied at each word, as shown in the figure, to store discrete passes at each last word. In very large vocabulary tasks, the execution time increases since the number of tree-structured lexicons increases in proportion to the vocabulary.

In contrast, the N-best word-pair approximation treats discrete passes at each last top N words. This method reduces the amount of calculation since it limits the number of tree-structured lexicons to N.

2.2.2. One-best approximation

One-best approximation treats only the best Viterbi pass, ignoring word contexts at each frame [5]. An overview of one-best approximation is also shown in Fig. 2.

Since one-best approximation uses only a single tree-structured lexicon, it is noticeably efficient for LVCSR. However, errors occur in Viterbi passes and language scores since it treats only the best pass, ignoring word contexts in interword transitions. If the amount of vocabulary increases, errors in language scores will also increase and adversely affect the result.

3. Improvements for over-million task

One-best approximation is efficient for very large vocabulary tasks. However, errors occur in Viterbi passes and language scores. Since word-pair approximation requires a very long execution time in very large vocabulary tasks, it is difficult to reduce the amount of word context dependency approximation errors.

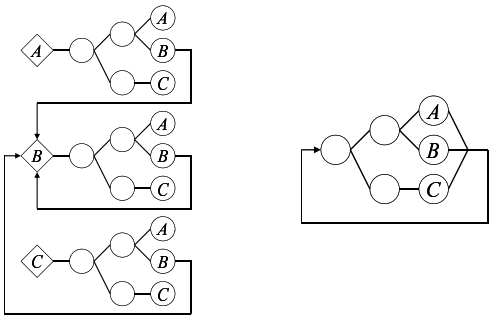

Figure 2: Word-pair approximation and one-best approximation.

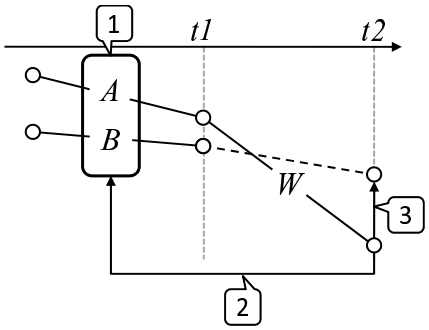

Figure 3: Delayed LM rescoring.

A method was adopted for treating only the best Viterbi pass as a one-best approximation and recalculating 2-gram probabilities with every last word at the word end. This method is similar to delayed bigram [6].

The algorithm of this method is described below, and an overview is shown in Fig. 3.

1. When a word W on the best Viterbi pass is identified, the last-word set that ends at the same frame with the start of W is obtained.

2. 2-gram probabilities of W and all words in the last-word set are calculated.

3. The language score is replaced with the best 2-gram probability at the word end of W.

These values correspond to Fig. 3, which shows that two words, A and B, reach word end and transit to W at the same frame $t1$. If the score of A is the best at $t1$, only the pass of A will transit to W. However, if the score of word sequences (A, W) is lower than (B, W) at $t2$, that is, the word end of W, an error will occur in the best pass. In this method, 2-gram probabilities of word sequence (A, W) and (B, W) are calculated, and the best probability is replaced with the language score at $t2$. This can reduce errors in language scores in one-best approximation.

In this method, the amount of calculation for word context dependency can be greatly reduced by treating only the best pass, while the resulting error will be recovered at later frame with a smaller computational cost.

Table 1: Configuration of algorithm

Configuration	Approximation method	
	LM factoring	Word context
Default	1-gram	One-best
2-gram	2-gram	One-best
Word-pair	1-gram	Word-pair
10-best-wp	1-gram	10-best-wp
Delayed	1-gram	Delayed

Table 2: Size of Google 3-gram

Vocabulary size		1,693,860
Number of entries	2-gram	80,328,381
	3-gram	392,559,067
File size (gzipped)		5,170MB

4. Experiments

To evaluate the performance of Julius and the proposed improvement in very large vocabulary CSR, LVCSR using an LM with one with more than a million ("over-million") vocabulary was carried out. Word-level approximations, 2-gram factoring, 1-gram factoring, word-pair approximation, N-best word-pair approximation, one-best approximation, and delayed LM rescoring method were compared. In these experiments, the N was set to 10.

4.1. Conditions

Julius rev-4.1.5 was used in these experiments. A configuration of the algorithm is described in Table 1.

The test set was the Japanese Newspaper Article Sentences (JNAS) IPA-98-TestSet, which consists of 200 items of reading speech data from newspaper articles. A triphone hidden Markov model (HMM) consisting of 1,507 states and 16 mixtures for each state was trained by the JNAS speech data. Experiments were conducted on a computer with an Intel Xeon X5450 3-GHz processor and 64-bit Linux OS.

Google Web Japanese N-gram [7] was used for this experiment. 2-gram and backward 3-gram was extracted using IRSTLM [8]. Google N-gram is a large-scale language resource consisting of 2,565,424 vocabulary Japanese word N-gram and their frequencies obtained from the Web. The pronunciation dictionary was built using MeCab [9] and a Web text mining method [10]. Words that could not be annotated were removed from the vocabulary. The finall vocabulary size was 1,683,860. The size of Google 3-gram is given in Table 2.

JNAS 3-gram, 20-k vocabulary newspaper LM was also used to evaluate the performance of very large vocabulary CSR. Memory size which is shown in Table 3 required to carry out very large vocabulary CSR with Google 3-gram.

4.2. Experimental results

The baseline performance of very large vocabulary CSR using the default configuration is plotted in Fig. 4. In this experiment, beam widths were set to 2,000, 4,000, 6,000, 8,000 and 10,000. For the Google 3-gram, the maximum word accuracy was 91.41% with a beam width of 10,000 and the execution time was about 1.5 times that of the JNAS 3-gram with the same beam width. Word accuracy was lower than that of the JNAS 3-gram. This was due to a mismatch between the language model and test set tasks and a greater number of approximation errors.

Table 3: Memory usage

		Memory usage (MB)
JNAS 3-gram	Process size	150
	LM size	30
	Tree size	8
Google 3-gram	Process size	7,500
	LM size	5,400
	Tree size	1,054

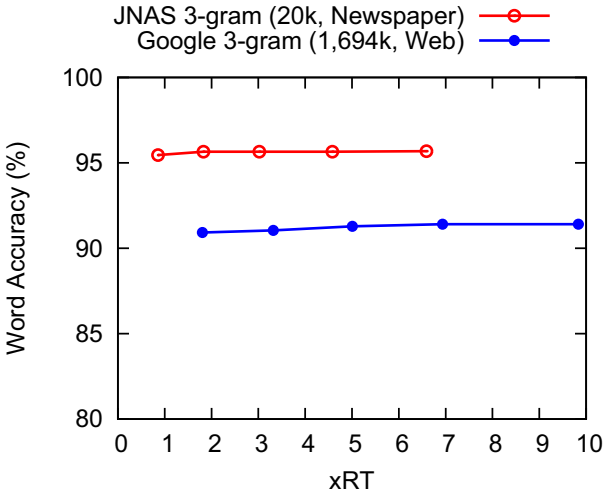

Figure 4: Word accuracy and decoding speed (TestSet: Newspaper).

A comparison of approximations on the over-million task is shown in Tables 4 and 5. In these experiments, beam widths of 3,000 and 10,000 were used. These tables indicate that 2-gram factoring improved word accuracy by 3.5% on the first pass and 0.5% on the final pass over 1-gram factoring. However, the amount of calculation increased substantially. Therefore, it is proved that 1-gram factoring can run very efficiently in very large vocabulary tasks.

Word-pair approximation improved word accuracy by 7.0% on the first pass and 0.7% on the final pass from one-best approximation. However, it was much slower than one-best approximation. The 10-best word-pair approximation performed as accurately as word-pair approximation in a much shorter time. Nevertheless, it required twice the execution time as one-best approximation. Thus, it was confirmed that one-best approximation is advantageous in very large vocabulary tasks from the viewpoint of the computational cost.

The delayed LM rescoring method described in Section 3 improved word accuracy on the final pass by 0.8% relative to one-best approximation and by 0.1% relative to word-pair approximation. This result indicates that recalculating reasonable language scores using only the best pass can reduce the number of errors.

With a beam width of 3,000, the delayed LM rescoring method was faster than 10-best word-pair approximation. Furthermore, there is room for a further increase in speed with

Table 4: Comparison of approximations on over-million task (Beam width: 3,000)

Algorithm Configuration	Word Accuracy (%)		xRT
	Final pass	First pass	
Default	91.05	77.49	2.77
2-gram	91.66	81.06	109.23
Word-pair	91.73	84.18	64.05
10-best-wp	91.73	84.01	6.72
Delayed	**91.86**	77.28	4.53

(LM: Google 3-gram, beam width: 3,000)

Table 5: Comparison of approximations on over-million task (Beam width: 10,000)

Algorithm Configuration	Word Accuracy (%)		xRT
	Final pass	First pass	
Default	91.41	77.48	9.70
2-gram	91.89	81.06	246.17
Word-pair	92.12	84.48	478.56
10-best-wp	92.25	83.91	18.26
Delayed	**92.24**	76.62	24.05

(LM: Google 3-gram, beam width: 10,000)

the delayed LM rescoring method by introducing a kind of inter-frame cache. With a beam width of 10,000, however, the delayed LM rescoring method was slower than 10-best word-pair approximation. In the delayed LM rescoring method, the amount of calculation necessary to obtain language scores increases in proportion to the number of hypotheses. The numbers of hypotheses per frame with beam widths of 3,000 and 10,000 are listed in Table 6. This problem is expected to be solved by using the cache.

The delayed LM rescoring method was also tested on smaller task. The results in 20k-word JNAS task are listed in Table 7. The delayed LM rescoring method improved word accuracy by 2.7% on the first pass and by 0.4% on the final pass over one-best approximation. Additionally, it achieved the same word accuracy as 10-best word-pair approximation in less time. This result shows that the delayed LM rescoring method also works well on a 20k-word task.

5. Conclusion

The performance of Julius in over-million LVCSR was evaluated and the algorithms were investigated. Experiments showed that Julius achieved a word accuracy of about 91% and a real time factor of about 2 in over-million large vocabulary CSR for Japanese newspaper speech transcription. Additionally, by using delayed LM rescoring, an improvement of 0.8% was achieved. Delayed LM rescoring method was as accurate as word-pair approximation and performed 14 times faster. Future work will be dedicated to more fast decoding using inter-frame cache and memory saving using quantize N-gram to realize an over-million LVCSR with low computational resources.

6. Acknowledgements

This research was partially supported by Grant-in-Aid for Scientific Research (21300066). The authors would like to thank the Speech and Acoustics Processing Laboratory of NAIST, Japan for providing us with the dictionary for Google N-gram.

Table 6: Number of hypotheses per frame

Beam width	Number of hypotheses per frame
3000	184.94
10000	683.01

Table 7: Comparison of approximations on 20k-word task

Algorithm Configuration	Word Accuracy (%)		xRT
	Final pass	First pass	
Default	95.42	87.30	1.32
10-best-wp	95.84	92.26	3.29
Delayed	**95.84**	90.04	2.06

(LM: JNAS 3-gram, beam width: 3,000)

7. References

[1] 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, Vol. 15, pp. 1352-1365, 2007.

[2] J. Schalkwyk, D. Beeferman, F. Beaufays, B. Byrne, C. Chelba, M. Cohen, M. Garrett, B. Strope, "Google Search by Voice: A Case Study," Advances in Speech Recognition: Mobile Environments, Call Centers, and Clinics, Amy Neustein (Ed.), Springer-Verlag 2010.

[3] H. Ney and X. Aubert, "A Word Graph Algorithm for Large Vocabulary Continuous Speech Recognition," Proc. of ICSLP, pp. 1355-1358, 1994.

[4] F.K. Soong and Eng-Fong Huang, "A Tree-Trellis Based Fast Search for Finding the N-Best Sentence Hypotheses in Continuous Speech Recognition," Proc. of IEEE-ICASSP, pp. 705-708, 1991.

[5] A. Lee and T. Kawahara, "Recent Development of Open-Source Speech Recognition Engine Julius," Proc. of APSIPA, pp. 131-137, 2009.

[6] M. Woszczyna and M. Finke, "Minimizing search errors due to delayed bigrams in real-time speech recognition systems," Proc. of ICASSP, pp. 137-140, 1996.

[7] T. Kudo and H. Kazawa, "Web Japanese N-gram Version 1," published by Gengo Shigen Kyokai, 2007.

[8] F. Marcello, B. Nicola, C. Mauro, "IRSTLM: an open source toolkit for handling large scale language models," Proc. of INTERSPEECH, pp. 1618-1621, 2008.

[9] T. Kudo, K. Yamamoto and Y. Matsumoto, "Applying Conditional Random Fields to Japanese Morphological Analysis," Proc. of the 2004 Conference on Empirical Methods in Natural Language Processing, pp. 230-237, 2004.

[10] J. Miyake, S. Takeuchi, H. Kawanami, H. Saruwatari, K. Shikano, "Automatic Reading Annotation to Japanese Trendy Words based on Parentheses Expression," Proc. of Oriental COCOSDA, pp. 81-86, 2008.