A SPONTANEOUS SPEECH RECOGNITION ALGORITHM USING WORD TRIGRAM MODELS AND FILLED-PAUSE PROCEDURE

JiN'ichi Murakami and Shoichi Matsunaga

ATR Interpreting Telecommunications Research Labs., 2-2 Hikaridai Seika-cho Soraku-gun Kyoto 619-02 Japan

ABSTRACT

This paper describes an effective recognition algorithm that uses word trigram models directly and a procedure for dealing with filled-pauses in spontaneous speech. This recognition algorithm greatly reduces the memory requirements and computational costs by employing two techniques: beam search and an improved Viterbi search. With these methods, execution can be performed in a 15M byte space for about a 1500-word vocabulary. The filled-pause procedure, capable of handing many filled-pauses in spontaneous speech, is then examined for spontaneous speech recognition. Even though the proposed algorithm employs a simple procedure, a 42.0% sentence recognition rate is obtained for spontaneous speech. Including the semantically correct sentences, the sentence recognition rate is about 75%.

1. INTRODUCTION

This paper describes a spontaneous speech recognition algorithm using word trigram models and a procedure for dealing with filled-pauses.

Spontaneous speech would be ideal for human-machine interfacing, if it were not for the many serious technical problems associated with its automatic recognition: filled-pauses, hesitations, self-corrections, out-of-vocabulary words, etc. These conditions make the acoustic modeling difficult. Therefore, lower perplexity language modeling has been prefered for spontaneous speech recognition.

Among the many language models for speech recognition, word bigram models[1] seem to be the most widely used because of their effectiveness and simplicity. However, for spontaneous speech recognition, word trigram models[2] should be used because they have been shown to produce a relatively lower perplexity over other language models (for example, network grammar, context free grammar, etc.). Yet, word trigram models pose some problems of their own when applied to speech recognition systems. One of the biggest problems is the large memory requirements and computational costs required for Viterbi decoding[3].

One possible way of avoiding these problems is to use an N-best paradigm: First, N-best candidate lists are generated using a word bigram model. Next, the candidates on these lists are rescored using a word trigram model and the best candidate is selected as the recognition result. This method has been used in a BBN recognition system[4]. Unfortunately, this algorithm's disadvantage is that the correct sentence may be out of the N-best lists with the word bigram model; consequently, this algorithm might decrease the performance of the word trigram model. Therefore, the need remains for developing efficient implementations of word trigram models directly.

In addressing these problems, we have devised a new algorithm that combines a word trigram model and a procedure for dealing with filled-pauses as a first step toward spontaneous, large-vocabulary, speaker-independent speech recognition. Firstly, we modified a frame synchronous Viterbi recognition algorithm using the word trigram model to greatly reduce the memory requirement and computational cost. Secondly, we addressed the problem of handling pauses and filled-pauses in language modeling by ignoring them in the calculation of the word trigram probabilities.

Speaker-independent speech recognition experiments were carried out using spontaneous 261-sentence data. On recording the speech data, each speaker remembered the intention of the utterance, and spoke freely while including many kinds of filled-pauses. The recognition task was a conference registration task, and the vocabulary size and word perplexity were 1567 and 4, respectively. The sentence recognition rate was greatly increased from 25.3% to 42.0% for the top-1 candidate and from 39.5% to 63.4% for the top-8 candidates. These results show the effectiveness of combining a word trigram model and filled-pause procedure.

2. RECOGNITION ALGORITHM USING WORD TRIGRAM MODELS

In this section, we describe a frame synchronous speech recognition algorithm using word trigram models based on Viterbi search. Among the many speech recognition algorithms, the Viterbi search (or one-pass DP) is well suited for Markov models used as language models like bigrams or trigrams.

To compute the Viterbi path to the n'th recognized word w_n, at time t for the bigram algorithm, we need to know the Viterbi paths emerging from each w_{n-1} word candidate, at time $t-1$. However, for the trigram algorithm, for each previous word candidate w_{n-1}, we need to know not only the Viterbi paths emerging from words at time $t-1$, but also the most likely paths passing through all possible w_{n-2}, w_{n-1} word pair combinations. This means that this algorithm essentially requires a lot of memory and high computational costs.

We define $w(t)$ as the word uttered at time t, $w_{\text{Prev}}(t)$ as the word uttered prior to $w(t)$, and $G_t(w_1, w_0, i) = \text{Prob}(O_0, O_1, ..., O_{t-1} \bigwedge s(t) = i, w(t) = w_0, w_{\text{Prev}}(t) = w_1)$. We can calculate $G_t(w_1, w_0, i)$ recursively using the following algorithm (Table 1).

Table 1. Recognition algorithm using word trigram models (Viterbi search)

[Definition]
l_w:number of states of word w a^w_{ij}:transition probability in word w from state s_i to state s_j $b^w_j(O(t))$:symbol output probability in word w at state s_j for observation vector at frame t $P(w_c\|w_a, w_b)$:trigram probability of word w_c after w_a, w_b have appeared. Q:vocabulary size T:number of input frames α:weight between trigram probability and HMM likelihood
[Initialization]
execute step1 for $w_0 = 0, ..., Q-1$ 1) $G_0(start, w_0, 0) = P(w_0\|start, start)$ ($start$ means sentence head)
[Viterbi search]
execute step2 and step6 for $t = 0, 1, .., T-1$ 2) execute step3 for $w_1 = 0, ..., Q-1$ 3) execute step4 for $w_0 = 0, ..., Q-1$ 4) execute step5 for $i = 0, 1, ..., l_{w_0} - 2$ 5) if $i = 0$ $G_t(w_1, w_0, i) = G_{t-1}(w_1, w_0, i) \times a^{w_0}_{i,i} \times b^{w_0}_i(O_t)$ else $G_t(w_1, w_0, i) =$ $\max(G_{t-1}(w_1, w_0, i) \times a^{w_0}_{i,i} \times b^{w_0}_i(O_t),$ $G_{t-1}(w_1, w_0, i-1) \times a^{w_0}_{i-1,i} \times b^{w_0}_{i-1}(O_t))$
[Viterbi search (word boundaries)]
6) execute step7 for $w_1 = 0, 1, ..., Q-1$ 7) execute step8 for $w_0 = 0, 1, ..., Q-1$ 8) $\Delta = \max_{0 \le w_2 \le Q-1}(G_{t-1}(w_2, w_1, l_{w_1} - 2)$ $\times a^{w_1}_{l_{w_1}-2, l_{w_1}-1} \times b^{w_2}_{l_{w_1}-2}(O_t) \times P(w_0\|w_2, w_1)^\alpha)$ if $\Delta \ge G_t(w_1, w_0, 0)$ then $G_t(w_1, w_0, 0) = \Delta$

3. MEMORY AND COMPUTATIONAL COST REDUCTION

This algorithm is of complexity $O(Q^2 \cdot l_w \cdot T)$ as opposed to $O(Q \cdot l_w \cdot T)$ for a bigram grammar. This entails a large memory and huge computational cost. To overcome these problems, the following methods are adopted.

3.1. Beam search

Beam search is employed to decrease the computational cost[5]. In Viterbi search, at each time t, we compute the probability of all possible combinations. In comparison, with the beam search algorithm, we only compute the probability of the most likely word combinations within a beam width B. Assuming a large enough beam width, the globally optimal Viterbi path will exist within that beam width. The complexity of the beam search algorithm is $O(B \cdot T)$. So the memory requirement and the calculation cost are reduced by $OB/(Q^2 \cdot l_w)$.

3.2. Pruning of beam width

There are two methods to prune the beam width.

1. Select the beam width based on some threshold.
2. Select a fixed beam width.

In most case, a method that selects the beam width with some threshold is used. Such a method is computationally cheap. But the threshold must be determined before recognition[1]. Therefore, under some recognition conditions, the output is faulty. A method that selects a fixed beam width, on the other hand, requires $G_t(w_1, w_0, i)$ to be sorted for each frame. This method requires a lot of computation. However, $G_t(w_1, w_0, i)$ need not be fully sorted, if only the values of the best B likelihoods are calculated, and the other likelihood values are subsequently pruned. Concretely, if the maximum beam width is N, this calculation is only of order $O(log_2 N)$, which is not too costly.

3.3. The path calculation for beam search

For normal Viterbi search, a traceback is needed to recover the word string after a likelihood calculation. For this reason, the memory must contain information as to which word is selected for each frame. This requires $B \times T$ memory cells. A capacity of $B \times H$ is however sufficient, if we store likelihoods and word strings as $G_t(w_1, w_0, i)$ together, where H is the number of words in a sentence. In this case, no traceback is needed. In most cases, H, the number of words in a sentence, is smaller than T, the number of input frames, so the memory is reduced but the calculation cost is slightly increased.

3.4. N-best search

The N-best-algorithm can be implemented in a two-pass strategy; e.g. forward-backward search or A* search. However, we choose another technique. We prepare N candidates for each $G_t(w_1, w_0, i)$ and store the N-th candidate for step 8 in Table 1. Therefore, this proposed algorithm is a one-pass strategy. If beam search is not used, this method generates a complete N-best list. However, if beam search is used, only an approximate N-best list is obtained.

With these techniques, we can recognize for a 1500-word vocabulary in a 15M byte space at about 1 or 2 minutes per sentence for word trigram models using HP 9000/730.

In this algorithm, the computational cost depends on the beam width and does not depend on the language model. Additionally, this algorithm can be used with any left-to-right parsing algorithm such as CYK. Similarly, if we use a high order Markov model, the recognition rate can be raised for text-closed data.

4. SPEECH RECOGNITION EXPERIMENTS

4.1. Experimental conditions

Experiments are performed for speaker dependent (SD) and speaker independent (SI) sentence speech recognition. For comparison, a word bigram model experiment is also carried out under the same conditions.

The test data were spoken by a male broadcast announcer. The training text data employed to estimate word trigram probabilities consisted of about 15,000 sentences with 190,000 words from the ATR Dialog Database, and an additional 261 test speech sentences. The task perplexity was 4.0 for the word trigram. The word HMMs were made by connected phone HMMs. A summary of these experimental conditions is shown in Table 2 .

Table 2. Experimental conditions

Algorithm	continuous mixture HMM + beam search + word trigram models (One-pass DP)
Phone model	4-state 3-loop left-to-right model
Acoustic parameters	16th order LPC cepstrum + power + Δ power + 16th order Δ cepstrum
Frame period	5 mS
Training data (SD)	2620-word utterance
Training data (SI)	12 male speakers, 736-word utterance
Vocabulary	1,567
Beam width	4,096
Test data	261 sentences
Speaking style	read

4.2. Results of experiments

We obtained a 66.7% sentence recognition rate for the top-1 candidate and 75.1% for the top-8 candidates for the speaker dependent (SD) case with the word trigram models. However, no sentence was recognized for the speaker independent (SI) case because of word insertion. (For example, "Yes" being recognized as " Oh yes ".)

4.3. Pause skip

In our survey of erroneous results (section 4.2.), we found that the wrong sentences included */pause/* in the speech data. We concluded that such */pause/* causes errors because acoustic parameters and word trigram models do not correspond. To counter this, we modified the recognition algorithm, based on the following techniques. As */pause/* is considered to be one word, it can be recognized, and thus, the trigram probability can be calculated to skip it. For example, when we see that w1 = */pause/* in our calculation of $P(w0|w2, w1)$, we calculate $P(w0|w3, w2)$ instead. And if we calculate $P(w0|w2, w1)$, and $w0 =$ */pause/*, we set $P(w0|w2, w1)$ to 1.

This algorithm raised the sentence recognition rate to 71.6% as opposed to 66.7% with no such procedure for the SD case. Furthermore, a great improvement was found for the SI case (61.7% as opposed to 0.0% with no such procedure).

4.4. Pause adaptation

Even though a pause skip technique is employed, the insertion error in the pause period did not always disappear. So, we retrained the */pause/* HMM for the test data with the Baum-Welch algorithm using some */pause/* units at the beginning of the test data. Because of this procedure, the sentence recognition rate was 83.9% as opposed to 61.7% with no such procedure for the SI case. This recognition rate was very close to the SD case's. A summary of these experiments is shown in Table 3 .

Table 3. Results of read speech recognition

sentence recognition rate(%)

language		bigram	bigram	trigram	trigram
procedure	order	SD	SI	SD	SI
original	1	42.5%	0.0%	66.7%	0.0%
word trigram	∼8	51.3%	0.0%	75.1%	0.0%
pause skip	1	49.4%	31.4%	71.6%	61.7%
	∼8	60.2%	44.4%	80.0%	76.7%
pause skip &	1	60.5%	44.8%	90.4%	83.9%
adaptation	∼8	76.2%	55.6%	97.7%	96.6%

text-closed; beam width: 4096; α: 1;

5. SPONTANEOUS SPEECH RECOGNITION

Spontaneous speech would be ideal for human-machine interfacing, if it were not for the many serious technical problems associated with its automatic recognition: filled-pauses, hesitations, retractions, out-of-vocabulary words, etc. Among these problems, the filled-pauses constitute the most remarkable phenomenon in that they appear in about 42% of spontaneous sentences. We therefore require a more accurate speech recognition algorithm that takes this phenomenon into account.

5.1. Filled-pauses procedure

In section 4.3., we described a procedure to counter */pause/* in speech data. This technique can also be used for the procedure for dealing with filled-pauses in spontaneous speech. We therefore propose two techniques for this procedure.

1. Filled-pauses Skip
   Filled-pauses are already known as phone strings (e.g. "well" is /W/ /EH/ /L/), and recorded in a word dictionary. And this method skips filled-pauses like a */pause/*. Additionally, we use the unigram probabilities of filled-pauses as the penalty.
   For example, let's assume that the speech data is as follows.
   " Do you already have well a registration form ? "
   In this case, "well" is a filled-pause. The trigram probabilities are calculated as follows.
   $P(\text{“do”}) \times P(\text{“you”}|\text{“do”}) \times P(\text{“already”}|\text{“do”},\text{“you”}) \times P(\text{“well”}) \times P(\text{“have”}|\text{“you”},\text{“already”}) \times P(\text{“a”}|\text{“already”},\text{“have”}) \times P(\text{“registration”}|\text{“have”},\text{“a”})$.

In this equation, $P(\text{“well”})$ is the penalty and $P(\text{“have”}|\text{“you”},\text{“already”})$ means to skip filled-pauses.

2. Phone-strings Skip

   As filled-pauses are considered to be sequences of phones, this technique skips phones strings as filled-pauses, we also use a penalty for the phone trigram probabilities. This technique is also used for the procedure for dealing with hesitations, retractions and out-of-vocabulary words .

   For example, let's assume the speech data is as follows.

   “*Do you already have well a registration form?*” (*/D/UW/ /Y/UW/ /AO/L/R/EH/D/IY/ /H/AE/V/ /W/EH/L/ /AH/ /R/EH/J/IH/S/T/R/EY/SH/UN/ /F/OH/M?*)

   The trigram probabilities are calculated as follows.

   $P(\text{“do”}) \times P(\text{“you”}|\text{“do”}) \times P(\text{“already”}|\text{“do”},\text{“you”}) \times P(/W/|/E/,/V/) \times P(/EH/|/V/,/W/) \times P(/L/|/W/,/EH/) \times P(\text{“have”}|\text{“you”},\text{“already”}) \times P(\text{“a”}|\text{“already”},\text{“have”}) \times P(\text{“registration”}|\text{“have”},\text{“a”})$.

   In this equation, $P(/W/|/E/,/V/)$, $P(/EH/|/V/,/W/)$ and $P(/L/|/W/,/EH/)$ mean the penalty and $P(\text{“have”}|\text{“you”},\text{“already”})$ means to skip phone strings.

5.2. Experiments for spontaneous speech

We perform experiments with word trigram models and for the SD case. The test data were spoken as follows.

1. Read speech
   Speakers uttered text data in read style.
2. Semi spontaneous speech
   Speakers uttered text data that contained symbols indicating filled-pauses. Except for the filled-pauses, the above utterances were the same as the read speech.
3. Spontaneous speech
   Speakers remembered the intention of the text data, and spoke freely. The contents of this utterances differed from the read speech. Therefore, this test data included filled-pauses and out-of-vocabulary words.

The context of the text data was the same as in section 4.4.; the speakers, however, differed and were not broadcast announcer.

5.3. Results of spontaneous speech

In the filled-pause skip experiments, the number of filled-pauses in the vocabulary was 109. The experimental results are shown in Table 4 .

As can be seen, the phone-strings skip method obtains a higher performance than the filled-pauses skip method. In addition, the sentence recognition rate for spontaneous speech is 42.0% as opposed to 25.3% with no such procedure; including the semantically correct sentences, the sentence recognition rate was about 75%. This means that this algorithm is effective in spontaneous speech recognition.

Table 4. Results of spontaneous speech recognition

sentence recognition rate(%)

speaking style	order	original word trigram	Filled-pauses skip	Phone-strings skip
read	1	90.5%	82.1%	92.7%
	~ 8	98.5%	95.4%	98.9%
semi spontaneous	1	42.0%	63.7%	65.6%
	~ 8	43.9%	80.1%	72.9%
spontaneous	1	25.3%	42.7%	42.0%
	~ 8	39.5%	61.1%	63.4%

speaker-independent; word trigram; beam width: 16384; α: 16;

6. CONCLUSION

In this paper, we described an effective speech recognition algorithm that uses word trigram models and a filled-pause procedure to reduce the memory and computational requirements. With these methods, we can perform execution in a 15M byte space for about a 1500-word vocabulary. Using this algorithm, a sentence recognition rate of 66.7% was obtained for speaker dependent recognition. We also described a procedure to skip */pause/* in speech data. Using this procedure, a sentence recognition rate of 83.9% was obtained as opposed to 0.0% with no such procedure for speaker independent recognition. This procedure was then extended to the procedure for dealing with filled-pauses in spontaneous speech. In experiments for spontaneous speech, we obtained a 42.0% sentence recognition rate; including the semantically correct sentences, the sentence recognition rate was about 75%. In the future, we will try to raise the sentence recognition rate for spontaneous speech by controlling the duration of filled-pauses.

Acknowledgments

We would like to thank Dr. Yamazaki, President, ATR Interpreting Telecommunications Research Laboratories and Dr. Sagisaka for their continuous support of this work. We are also grateful to all of the members of Department 1 for their advice and encouragement.

REFERENCES

[1] Kai-Fu Lee, “Large-Vocabulary Speaker Independent Continuous Speech Recognition: The SPHINX System,” 15213 CMU-CS-88-148 (April 18, 1988).

[2] A. Averbuch, et al., “An IBM PC Based Large-Vocabulary Isolated-Utterance Speech Recognizer,” Vol.1, 2.4.1 pp.53-56 ICASSP (1986).

[3] Kiyoshi Shikano, “Improvement of Word Recognition Results by Trigram Model,” Vol.3, 29.2.1, pp.1261-1262 ICASSP (1987).

[4] Long Nguyen, et al., “Search Algorithms for Software Only Real Time Recognition with Very Large Vocabularies,” DARPA Human Language Technology Workshop (1993).

[5] H.Ney, et al., “A Data Driven Organization of The Dynamic Programming Beam Search for Continous Speech Recognition,” Proc. ICASSP87, pp.833-836 (May 1987).