AN EFFICIENT ALGORITHM FOR USING WORD TRIGRAM MODELS FOR CONTINUOUS SPEECH RECOGNITION

Jin'ichi Murakami and Shigeki Sagayama

ATR Interpreting Telephony Research Laboratories

ABSTRACT - This paper describes an efficient algorithm for using word trigram models in continuous speech recognition. The algorithm reduces the memory requirements and computational cost by employing two techniques: beam search and an improved method for training the Viterbi path. It was tested on continuous speech recognition experiment.

INTRODUCTION

Many continuous speech recognition systems have recently been reported. These systems are given language information to improve the recognition rate because the syllable recognition rates are too low. Many different language models exist. These include a word trigram model for isolated word recognition, e.g., in an IBM system (Averbuch, 1986). Bigram models are used in CMU's SPHINX continuous speech recognition(Lee, 1988). ATR's continuous speech recognition system(Kita, 1989) uses a context-free grammar. Other language models such as network grammars or unification grammars have also been used. Among these language models, the word bigram model(Lee, 1988) seems to be the most widely used.

Word bigram models are effective models in spite of their simplicity. These models, however sometimes generate grammatical incorrect sentences. The reason for this is that word bigram models are too simple to express natural language. This can be improved by using word trigram models. There are two problems with using word trigram models in continuous speech recognition systems. One is the reliability of trigram probabilities. Since there are many parameters in word trigram models, e.g., for a 1,000 word vocabulary has 1,000,000,000 parameters, a very large amount of text data. The other problem is the large amount of memory and computational cost required for speech recognition. For these reasons, very few studies on continuous speech recognition systems using only word trigram models have been reported.

One possible method avoiding these problems is to use categories. If we use categories instead of words, the number of parameters is reduced. The memory and computational cost are reduced in speech recognition. Therefore, this method solves both problems together. Shikano(Shikano, 1987) and Murase(Murase, 1990) reported this model and showed experimental results. However, a word trigram grammar has a lower perplexity than a category trigram model. So the need for developing efficient implementations of the word trigram model remains.

In this paper, we first introduce a continuous speech recognition algorithm using word trigram models based on Viterbi search. Next, we show that the algorithm reduces memory requirements and computational cost. We explain how to use the beam search and how to calculate the Viterbi path. Finally, we present the experimental results obtained using this algorithm. The experiments show that the recognition rates obtained using word trigram models are the same as those obtained with word bigram models for text-open data. However, better results are obtained for text-closed data.

In this section, we describe a continuous speech algorithm using word trigram models based on Viterbi search. Well-known algorithms for continuous speech recognition include two-level DP matching, level building or Viterbi search (one-pass DP). Among them, the Viterbi search (one-pass DP) algorithm is well suited for Markov models with a language model such as bigram or trigram. To compute the Viterbi path to the n'th recognized word w_n, at time t, 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.

We define $w(t)$ the word uttered at time t, $w_{\mathrm{Prev}}(t)$ the word uttered previous $w(t)$, $G_t(w_1, w_0, i)$ = Prob$(O_1, O_2, ..., O_{t-1} \bigwedge s(t) = i, w(t) = w_0, w_{\mathrm{Prev}}(t) = w_1)$. We can calculate $G_t(w_1, w_0, i)$ recursively using the following algorithm.

Table 1: A Viterbi Algorithm Using Word Trigram Models

```
[ Definition ]
$l_w$ : the 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 $t$
$P(w_0|w_2, w_1)$ : trigram probability of word $w_0$ after $w_2, w_1$ have appeared.
$Q$ : vocabulary
$T$ : the number of input frames
[ Initialization ]
execute step1 under $w_0 = 0, ..., Q-1$
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) $G_t(w_1, w_0, i) = \max(G_{t-1}(w_1, w_0, i) \times a^{w_0}_{i,i} \times b^{w_0}_i(O_t), G_{t-1}(w_1, w_0, i-1) \times a^{w_0}_{i-1,i} \times b^{w_0}_{i-1}(O_t))$
[ Calculate word boundaries ]
6) execute step7 for $w_1 = 0, 1, ..., Q-1$
  7) execute step8 for $w_0 = 0, 1, ..., Q-1$
    8) $\Delta = \max_{0 \leq w_2 \leq Q-1}(G_{t-1}(w_2, w_1, l_{w_1} - 2) \times a^{w_1}_{l_{w_1}-2, l_{w_1}-1} \times b^{w_1}_{l_{w_1}-1}(O_t) \times P(w_0|w_2, w_1)$
    if $\Delta \geq G_t(w_1, w_0, 0)$ then $G_t(w_1, w_0, 0) = \Delta$
```

RECOGNITION ALGORITHM REDUCING THE MEMORY REQUIREMENTS AND COMPUTATIONAL COST

As one can see, this algorithm is of complexity $O(Q^2)$ as opposed to $O(Q)$ for a bigram grammar. This entails a large memory and computational cost. We will now concentrate on overcoming these problems.

Beam search

Beam search is employed to decrease the computational cost. In Viterbi search, at each time t, we compute the probability of all possible combinations. In comparison with beam search algorithm, at time t, we only compute the probability of most likely word combinations within a beam width B. Assuming a large enough beam, we can be confident that the globally optimal Viterbi path almost always exists within that beam. So the beam search algorithm reduces computation while still providing globally optimal solutions. In step5 of the algorithm, the best B value of $G_t(w_1, w_0, i)$ for each frame is calculated. Beam search has the following advantages.

1. Reduction of Memory and Computational Cost
   For Viterbi search, $Q \times Q \times l_w$ memory cells are needed to store $G_t(w_1, w_0, i)$. The beam search, however only needs B. Therefore, the memory needed to use this algorithm is significantly reduced. The computational cost is also reduced by the same ratio.
2. Reduction of the Access Count to Obtain Trigram Probability
   The trigram probabilities when using the beam search are best stored in a list structure to save memory. Therefore, to obtain the trigram probabilities, the indices needed to be calculated. As a result, with beam search, the total number of indices to obtain trigram probabilities is reduced. As a result, the computational cost is reduced.

The path calculation for beam search

For normal beam search, a traceback is needed to recover the word string after the Viterbi search. For this reason, the memory must contain information which word was 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. Usually, 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.

The method determining the beam width

There are two methods to determine the beam width.

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

Methods to select the beam width with some criterion are usually used. These methods are computationally cheap. However, the criterion must be determined before recognition. Therefore, under some recognition conditions, the output is faulty. The method to select a fixed beam width requires $G_t(w_1, w_0, i)$ to be sorted for each frame. This requires a lot of computation. But, $G_t(w_1, w_0, i)$ need not be fully sorted, i.e., only the value of the best B likelihoods are needed. Concretely, if the maximum beam width is N, this calculation is only of order $O(logN)$, which is not too costly.

In this algorithm, the computational cost depends on the beam width and does not depend on the language model. Word trigram models have the same cost as word bigram models. This algorithm can used with all left-to-right parsing algorithms such as CYK. If we use a high order Markov model, the recognition rate can be raised for text-closed data.

EXPERIMENTAL RESULTS OF CONTINUOUS SPEECH RECOGNITION USING WORD TRIGRAM MODELS

In this section, we describe the experimental results obtained using this algorithm. This experiment is a speaker dependent continuous speech recognition and the test sentences are spoken by an broadcast announcer. The flooring probability was set to e^{-1000}. The experimental conditions are shown in Table 2.

Table 2: Experimental conditions of experience for continuous speech recognition using word trigram models

algorithm	continuous mixture HMM + beam search + word trigram models
mixture number	max 14 (valid for each syllable)
state number	3-state 4-loop left-to-right model
acoustic parameter	16th order LPC cepstrum + power +delta power + 16th order delta cepstrum
frame window	20 ms
frame period	5 ms
training voice	word speech (5,240 words)
syllable category count	52 syllables
vocabulary	1,567
beam width	4,096
duration control	no
language information	word trigram models
unit of recognition	sentence
test sentence count	38 sentences
speaking style	read speech
speech content	international conference task (model conversation)

The probabilities of the word trigram models for language information are calculated as follows.

1. Test data is not included in the training data which is calculated word trigram probabilities. (text-open data)
2. Test data is included in the training data which is calculated word trigram probabilities. (test-closed data)

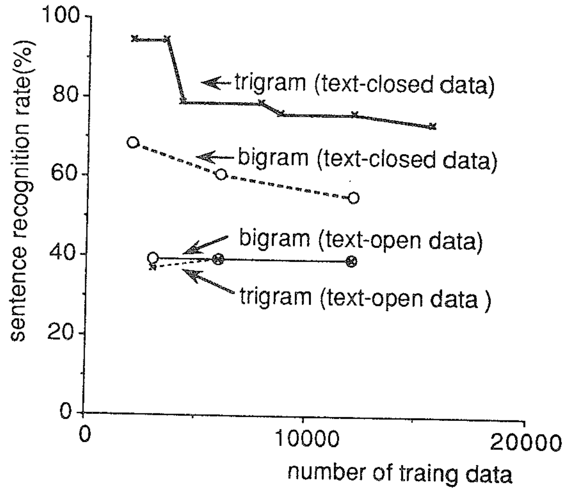
The training data includes about 15,000 sentences with 190,000 words of the ATR Dialog Database. Table 3 shows the task entropy. The experimental results are shown in Figure 1. Results from word bigram models are shown together for comparison. We obtained a 78% sentence recognition rate for text-closed data and 40% for test-open data. Figure 4 shows erroneous output results for the text-closed data. These results show that many sentences are semantically correct. Only four sentences are completely wrong. The sentence recognition rate is 89%. These completely wrong sentences include the pause in speech data.

We think that the pause causes the error because acoustic parameters and word trigram models do not correspond. However, the recognition rate for text-open data is the same for word bigram models and word trigram models. This is due to the small flooring probability value and the small amount of text data. Yet, in statistical language models like word trigram models, the recognition rate for text-open data depends on the coverage between the training data and test data. Therefore, we believe that the reliability of the recognition rate for the text-open data is very low.

Table 3: Task perplexity

ngram	Entropy	Perplexity
unigram	8.33	321.8
bigram	3.77	13.6
trigram	2.01	4.0

Figure 1: Experimental Result



CONCLUSION

In this paper, we described a speech recognition algorithm using word trigram models that reduces the memory and computational cost, and reported the obtained experimental results. A recognition rate of 40% for text-open data and of 78% for text-closed data was obtained.

REFERENCES

Averbuch, A. (1986) "An IBM PC based large-vocabulary isolated-utterance speech recognizer," ICASSP, 53-56.

Kita, K. (1989) " HMM continuous speech recognition using predictive LR parsing," ICASSP, 703-706.

Lee, Kai-Fu. (1988) "Large-Vocabulary speaker independent continuous speech recognition: the SPHINX system," 15213 CMU-CS-88-148.

Murase, T. & Nakagawa, S. (1990) "Sentence recognition method using word cooccurrence probability and its evaluation," ICSLP90,1217-1220.

Shikano, K. (1987) "Improvement of word recognition results by trigram model," ICASSP,1261-1262.

Table 4: Sentence Error in the Experiment (Text-Closed, Training Data 15713)

correct → output
kaiginoshukuhakushisetsunitsuiteotazuneshitainodesuga
→ kaiginoshukuhakushisetsunitsuiteotazuneshitaiNdesuga
会議の宿泊施設についてお尋ねしたいのですが
→ 会議の宿泊施設についてお尋ねしたい ん ですが
kyoutopuriNsuhoterugakaigizyounihachikainodesuga
→ kyoutopuriNsuhoterugakaigizyounihachikaiNdesuga
京都プリンスホテルが会議場には近いんですが
→ 京都プリンスホテルが会場には近い ん ですが
soredehakyoutopuriNsuhoteruoyoyakushitainodesuga
→ soredehakyoutopuriNsuhoteruoyoyakushitaiNdesuga
それでは京都プリンスホテルを予約したいのですが
→ それでは京都プリンスホテルを予約したい ん ですが
hoterunotehaimoshiteitadakerunodesuka
→ hoterunotehaimoshiteitadakeruNdesuka
ホテルの手配もしていただけるのですか
→ ホテルの手配もしていただける ん ですか
dehaonamaetogozyuushooonegaishimasu
→ gohaQpyouninarukatanogozyuushooonegaishimasu
ではお名前とご住所をお願いします
→ ご発表になる方の ご住所をお願いします
zyuushohatoukyoutominatokushiNbashiiQchoumeichibaNchisaNgoudesu
→ zyuushohanechoQtokyoutonokaiginihatourokunikaNshimashitekyounoseQshoNnoichibaNsaNgoudesu
住所は東京都港区新橋１丁目１番３号です
→ 住所は ねちょっと京都の会議には登録に関しまして今日のセッションの １番３号です
deNwabaNgoumoonegaishimasu
→ deNwabaNgouonegaishimasu
電話番号もお願いします
→ 電話番号お願いします
deNwabaNgouhasaNsaNichinoniigoniiichidesu
→ roNbuNnohaQpyouhagozeNchuunokuzinikaizyounichikaidesu
電話番号は３３１の２５２１です
→ 論文の発表は午前中の９時に会場に近いです
kyoutopuriNsuhoterunihachigatsuyoQkakarayoukamadehitoribeyaootorishimashita
→ kyoutopuriNsuhoterunihachigatsuyoQkakarayoukamadenihahaQpyoushaootorishimashita
京都プリンスホテルに８月４日から８日まで一人部屋をお取りしました
→ 京都プリンスホテルに８月４日から８日まで には発表者 にお送りしました。