REAL-TIME WORD CONFIDENCE SCORING USING LOCAL POSTERIOR PROBABILITIES ON TREE TRELLIS SEARCH

Akinobu LEE, Kiyohiro SHIKANO

Nara Institute of Science and Technology
Ikoma, Nara 630-0192, Japan

Tatsuya KAWAHARA

Kyoto University
Sakyo-ku, Kyoto 606-8501, Japan

ABSTRACT

Confidence scoring based on word posterior probability is usually performed as a post process of speech recognition decoding, and also needs a large number of word hypotheses to get enough confidence quality. We propose a simple way of computing the word confidence using estimated posterior probability while decoding. At the word expansion of stack decoding search, the local sentence likelihoods that contains heuristic scores of unreached segment are directly used to compute the posterior probabilities. Experimental result showed that, although the likelihoods are not optimal, it can provide slightly better confidence measures compared with N-best lists, while the computation is faster than 100-best method because no N-best decoding is required.

1. INTRODUCTION

The recent development of speech recognition techniques has raised a wide variety of speech application areas. However, the accuracy of current recognition system is still far from perfect in practical applications. In this context, the demand for annotating recognition results with some confidence scores, indicating how certain the system is about each word hypothesis, also increases. The confidence can provide an additional information about the recognized words. For example, in a spoken dialogue system, it allow a dialogue manager to reject uncertain words to avoid unnecessary interactions for utterance verification.

Confidence scoring based on the posterior probabilities of words is one of the popular method to annotate such confidences to the speech recognition results. A recognition decoder typically outputs the results as a form of N-best list or word graph[1], which contains a large number of competing word hypotheses and their associated likelihoods. Then, the posterior probability of each word hypothesis is computed on them by parsing the hypotheses. As the posterior probability reflects the relative distribution of word likelihoods among competing alternatives, it works well as a confidence measure[2].

A large number of word hypotheses are necessary to get enough confidence quality. The recognition decoder should output a large number of hypotheses even if the application itself requires only the best one (on automatic transcription task, etc), and finding much hypotheses results in the increase of computational cost. Also, as this confidence scoring should be done at the post processing of recognition, the processing time directly affects the turn-around time of the recognition system, which may suffer user interaction in spoken dialogue applications.

In this paper, we propose a new method to compute the posterior probability based confidence measure at the decoding time with very small amount of processing cost. Some works has been focused on assessing word confidence at decoding time[3][4], but [3] is not based on posterior probability and [4] is also a rescoring approach on the resulting word graph. In our method, instead of parsing through the whole lattice after decoding, the local likelihoods of partial sentence hypotheses of expanded words in the way of stack decoding are used to compute the word posterior probability. With tree-trellis search, this method enables rapid confidence scoring without generating N-best lists or word graphs.

2. CONFIDENCE SCORING USING WORD POSTERIOR PROBABILITY

First, the computation procedure of word confidence measures based on word posterior probability is briefly described. Let τ denote the starting time and t the ending time of word w. $W_{[w;\tau,t]}$ denotes all sentences that contain the hypothesis $[w;\tau,t]$. Given a N-best list or word graph from recognition decoder, the posterior probability $p([w;\tau,t]|X)$ of a specific word hypothesis $[w;\tau,t]$ over the acoustic observations X can be computed by summing up the posterior probabilities of all paths which contain the hypothesis $[w;\tau,t]$

$$p([w;\tau,t]|X) = \sum_{W\in W_{[w;\tau,t]}} \frac{p(X|W)p(W)}{p(X)} = \sum_{W\in W_{[w;\tau,t]}} \frac{e^{g(W)}}{p(X)} \quad (1)$$

where $g(W)$ denotes the log likelihood of a sentence hypothesis W derived from the recognition decoder, i.e.,

$$g(W) = \log p(X|W)p(W). \tag{2}$$

$p(X)$ is approximated by the sum over all paths through the lattice. This word posterior probability can be used directly as the confidence score of the word hypothesis

$$C([w;\tau,t]) = \sum_{W \in W_{[w;\tau,t]}} \frac{e^{\alpha \cdot g(W)}}{p(X)} \tag{3}$$

where α is a scaling parameter ($\alpha < 1$) that is necessary to avoid only a few words to dominate the sums in these equations because of the large dynamic range of acoustic likelihoods[2].

From the equations above, it is apparent that both the sum of probabilities of paths $W_{[w;\tau,t]}$ and of probabilities of all paths for $p(X)$ should be computed to estimate a word confidence score of $[w;\tau,t]$. If the hypotheses are given as N-best list, the computation process is a simple summation of sentence scores that contains $[w;\tau,t]$. On word graph, forward-backward algorithm is normally applied.

Here, the computational cost of computing the word confidence scoring is discussed on the basis of total recognition performance, that includes not only the cost of the confidence scoring but also the recognition cost to generate the hypotheses. Since a sufficient number of word hypotheses is required to estimate the confidence, a large number of N-best hypotheses should be searched on the recognition decoder. They should be computed even if the final application requires only the best one. Actually, several hundreds of sentence hypotheses are needed to get a sufficient result for N-best list based confidence scoring on large vocabulary continuous speech recognition. Obtaining such large number of hypotheses at the preliminary recognition process result in a much growth of processing time.

Moreover, since the computation of posterior probability for any word hypothesis requires overall likelihoods for the whole utterance, the scoring process cannot be started before all the word hypotheses are generated by the decoder. Such post-processing may cause an increase of turn-around time for a user application.

3. CONFIDENCE SCORING WHILE DECODING

We propose a new confidence scoring method that is applicable directly in the way of speech recognition procedure. The aim is to compute a word posterior based confidence measures efficiently and faster without generating much hypotheses and with small computational cost. The basic approach is to use the local likelihoods of partial sentence hypothesis at the point of word expansion in the decoding procedure to get approximated word posterior probability.

We assume in this paper that the speech recognition algorithm is based on a tree-trellis search[5][6], a typical two pass heuristic search based on tree lexicon. The first pass performs tree-lexicon search to generate intermediate results in word trellis form, that consists of all the survived word hypotheses with their boundary times and accumulated likelihoods from the beginning of the utterance. Then the second pass performs stack decoding in the reverse direction with more precise models, using the word trellis as the estimated heuristics of unreached part. Actually, when connecting a word w_n to a partial sentence hypothesis $w_1^{n-1} = w_1, w_2, ..., w_{n-1}$ on the second pass, the decoder objective function of the second pass $f(w_1^n)$ is computed as

$$\begin{aligned} f(w_1^{n-1}, [w_n;\tau,t]) &= g(w_1^{n-1}, t) + \hat{h}(w_n, t) && (4) \\ f(w_1^n) &= \max_{0 \le t < T} f(w_1^{n-1}, [w_n;t]) && (5) \end{aligned}$$

where $g(w_1^{n-1}, t)$ denotes the likelihood at the connecting edge of the partial sentence hypothesis w_1^{n-1} on time t, and $\hat{h}(w,t)$ denotes the likelihood of the connected word w at time t on the word trellis.

Let us consider deriving approximated word posterior probabilities from the likelihood $f(w_1^n)$. As it contains both likelihood of most likely path for the searched segment and heuristic likelihood of unsearched segment that are derived from the previous pass, it is possible to use the score directly instead of precise probability. It means approximation of $p(w_n, X)$ by using only the maximum likelihood path and use heuristic likelihood for the unsearched segment. For the estimation of $p(X)$, the summation of $f(w_1^n)$ for all words that exists around the target word can be used.

Finally, the approximated posterior probability $\hat{p}(w_n|X)$ of a word w_n can be obtained from $f(w_1)$,

$$\begin{aligned} W_c &= [w;\tau,t] : \tau \le t_n \le t && (6) \\ \hat{p}(w_n|X) &= \frac{e^{f(w_1^n)}}{\sum_{W_c} e^{f(w_1^{n-1},[w;\tau,t])}} && (7) \end{aligned}$$

where t_n is the maximum argument of equation (5).

Introducing this approximation enables the posterior probability based confidence scoring at search process, on the word expansion stage. The stack decoding algorithm integrated with the proposed confidence scoring is shown below.

1. Set initial hypotheses into the sentence hypothesis stack.
2. Repeat the following steps until requested number of sentences are obtained.
    (a) Pop the best hypothesis from the stack.
    (b) if the popped hypothesis has been reached at end of utterance, output it and exit.

(c) Determining next words: estimate the end time of popped hypothesis, and look for words in the trellis to pick up words whose end node exist at the estimated end time. They are the next word candidates (equation (6)).

(d) Generate new hypotheses by connecting those words, and compute their likelihoods by finding the optimal connection time for each (equation (4) and (5)).

(e) *Confidence scoring*: compute the word posterior probabilities of those word candidates using the likelihoods (equation (7)). Store the confidence score to each hypothesis.

(f) Push them to the hypothesis stack and go to (a).

As shown from the above algorithm, it is obvious that the computation cost needed for the confidence scoring is substantially small, as compared with parsing all the N-best lists or word graph. Moreover, it is not necessary to search for a large number of word hypotheses only for getting the confidence scores. These feature realizes fast speech recognition with confidence scoring.

However, computing confidence measure from such approximated posterior probability may suffer the quality of confidence scores. The likelihood used in the proposed method reflects only the best path, and it also contains heuristic scores.

4. EXPERIMENTAL RESULT

4.1. Conditions

The proposed method was evaluated through a recognition test. The proposed method is implemented on our open-source real-time large vocabulary recognition engine Julius[6], version 3.4[1]. It applies word 2-gram on the first pass and word 3-gram on the second pass. For comparison, the conventional N-best rescoring is also implemented.

The task domain is an unconstrained, natural question and answer dialogue with a software agent, located at the public civil hall. The task of the agent is guidance of the hall, some information query in and around the hall, greetings and so on. 500 utterances of adult's moderate speech are extracted as a test set. The average length of spoken utterance is 2.1 seconds. Acoustic model is a phonetic tied-mixture triphone model, and language model is a task-dependent word 3-gram. The dictionary consists of 41,248 words, and test set perplexity is 11.4. Other details about this task are in [7]. The scaling parameter α is set to 0.04 for all experiment.

On N-best method, the number of recognized sentence to be obtained is set to 100.[2] 10 are also tested for comparison.

[1]Available at `http://julius.sourceforge.jp/en/`

4.2. Evaluation measures

To assess the quality of confidence scoring, a threshold θ is defined to label the recognized words. Each word will be labeled as "*correct*" if the confidence score is equal or above the threshold, and "*false*" if below the threshold.

Several evaluation measure was introduced. To see the labeling accuracy of *correct* and *false*, confidence error rate (CER) was computed as the number of incorrectly assigned labels divided by the total number of recognized words. Detection-error-tradeoff (DET) curve was also drawn by plotting false acceptance rate over false rejection rate.

Moreover, another criteria was introduced in this experiment to evaluate the confidence scores in the domain of spoken dialogue system: failure of acceptance (FA) and slot error (SErr):

$$\mathrm{FA} = 1 - \frac{\text{num. of correct words labeled as } \textit{correct}}{\text{num. of words labeled as } \textit{correct}}, \quad (8)$$

$$\mathrm{SErr} = 1 - \frac{\text{num. of correct words labeled as } \textit{correct}}{\text{num. of reference words}}. \quad (9)$$

From the dialogue management's point of view, accepting wrong words has critical impact, since the wrongly accepted word will cause further confusion of discourse. Considering this, the weighed summation of the FA and SErr is also computed using weighing parameter λ:

$$(\mathrm{FA} \cdot \lambda + \mathrm{SErr}) \times 2/(\lambda + 1). \quad (10)$$

4.3. Results

The FA, SErr and FA+SErr with several weights for various threshold θ are plotted in Figure 1 and Figure 2 for N-best list method and proposed method, respectively. It is shown that although approximations has been introduced, the proposed method can provide almost the same confidence quality as the conventional N-best method. It was found that the proposed method can especially reduce the FA. This indicates that, while the N-best method can deal with only the limited variety of hypotheses, while the proposed method compute the confidence scoring directly from the word candidates that will later be dropped from search. On the other hand, the increase of Slot error is observed. It is supposed to occur by the approximation of likelihoods.

DET curves are also plotted in Figure 3. It is shown that our proposed method (labeled as "search" in the figure) has substantial performance over the N-best methods.

Finally, the minimal error rate of FA+SErr, CER, and average processing time is shown in Table 1. Both FA+SErr

[2]Since most of the test set utterance are short, search space become exhausted given a large N value. Actually, all 100 sentence results could not be obtained for 253 out of 500 utterances.

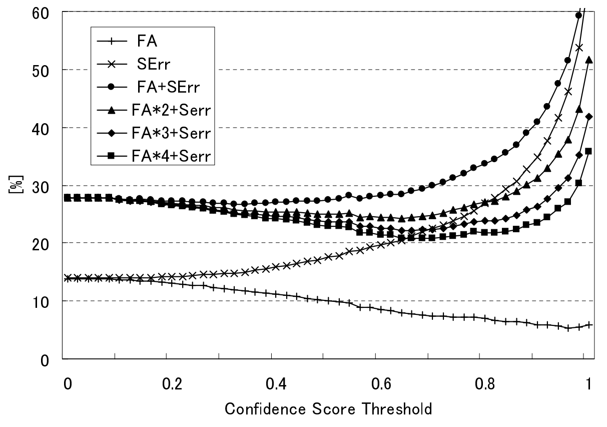

Fig. 1. FA and SErr by N-best method (N=100).

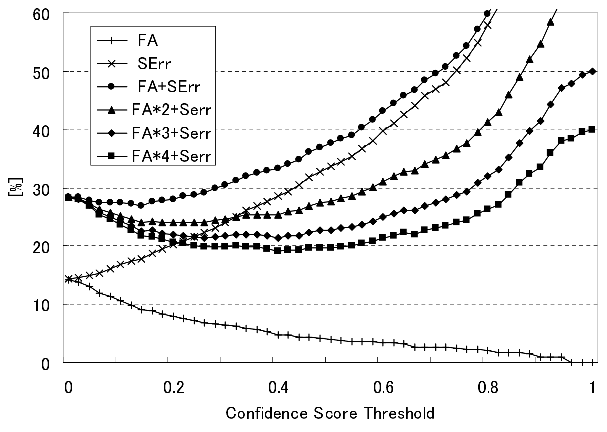

Fig. 2. FA and SErr by proposed method.

and CER was slightly improved by the proposed method. This result suspects that our approximation method not only contributes to reduction of computation, but also improves the quality of confidence.

The computation cost of confidence scoring is relatively small as compared with the 100-best result. Our method does not need N-best decoding and getting only the best sentence candidate is enough.

5. CONCLUSION

A rapid and efficient method to compute a word confidence measures based on word posterior probability has been introduced. Our method computes confidence scores at recognition decoder while recognition, and can produce confidence scores without searching for N-best results while keeping its quality. It can be applied to other search algorithm than the tree trellis search. Future work will be dedicated to more evaluation at wide variety of tasks.

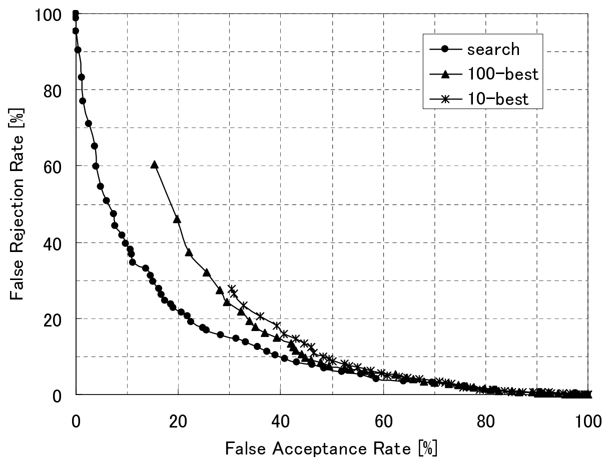

Fig. 3. Detection-error tradeoff curves.

Table 1. Minimal error rate [%] and average process time

	FA +SErr	FA*4 +SErr	CER	avg. time (sec)
10-best	27.1	21.9	12.3	2.0
100-best	26.7	20.8	12.3	2.4
search	26.9	19.1	11.8	2.1

CPU: Intel Xeon 2.4GHz, baseline CER = 14.1

6. REFERENCES

[1] T. Kemp and T. Schaaf, “Estimating confidence using word lattices,” in *Proc. EUROSPEECH*, September 1997, vol. 2, pp. 827–830.

[2] F. Wessel, R. Schluter, K. Macherey, and H. Ney, “Confidence measures for large vocabulary continuous speech recognition,” *IEEE Trans. Speech & Audio Process.*, vol. 9, no. 3, pp. 288–298, March 2001.

[3] C. Neti, S. Roukos, and E. Eide, “Word-based confidence measures as a guide for stack search in speech recognition,” in *Proc. ICASSP*, 1997, vol. 2, pp. 883–886.

[4] G. Evermann and P. Woodland, “Large vocabulary decoding and confidence estimation using word posterior probabilities,” in *Proc. ICASSP*, 2000, vol. III, pp. 1655–1658.

[5] F. K. Soong and E.-F. Huang, “A tree-trellis based fast search for finding the N best sentence hypotheses in continuous speech recognition,” in *Proc. ICASSP*, 1991, vol. 1, pp. 705–708.

[6] A. Lee, T. Kawahara, and K. Shikano, “Julius — an open source real-time large vocabulary recognition engine,” in *Proc. EUROSPEECH*, September 2001, pp. 1691–1694.

[7] R. Nisimura, Y. Nishihara, R. Tsurumi, A. Lee, H. Saruwatari, and K. Shikano, “Takemaru-kun: Speech-oriented information system for real world research platform,” in *Proc. First International Workshop on Language Understanding and Agents for Real World Interaction*, July 2003, pp. 70–78.