COMBINATION OF MULTIPLE PREDICTORS TO IMPROVE CONFIDENCE MEASURE BASED ON LOCAL POSTERIOR PROBABILITIES

Yuewen Fu , Limin Du

Center for Speech Interaction Technology Research
Institute of Acoustics, Chinese Academy of Sciences
Beijing 100080, China

ABSTRACT

Recently local word posterior probabilities computed from word expansion during stack decoding search was proposed to be a confidence measure under real-time condition. However, much approximation in its computation limits its quality. In this paper, we intend to improve its performance by using decision tree to combine it with other real-time predictors. A series of other predictors are constructed and the experiments on different combination of predictors using decision tree are carried out. The experimental results show that confidence measure based on local word posterior probability can be improved significantly (18.9% confidence error rate improvement relatively in our experiments) by combining with other real-time predictors. The experiments also show that local posterior probabilities of adjacent words are relatively effective predictors.

1. INTRODUCTION

The development of speech recognition techniques has increased the demand for the ability to spot erroneous words from recognition results. Confidence measures can be used for annotating each recognition word with either 'correct' or 'incorrect', providing useful information about words for application systems. In a spoken dialogue system, e.g. a ticket reservation system, confidence measures can be used to avoid unnecessary interactions and dialogue duration can be shortened.

The word posterior probability is one of the popular confidence measures in recent years. Much work has been done on this subject [1][2]. The word posterior probability is often computed from word graph or n-best list, and the estimation of posterior word probabilities on word graphs yields better results than the estimation on n-best list. Furthermore, the result from word graph is considered very efficient even if used as confidence measure singly and a further improvement is more difficult [3].

To get good confidence estimation, a large number of hypotheses are needed. However, under some cases, especially for real-time occasion, owing to the heavy cost of computation time, word graph or n-best list is sometimes not computed or their size is not large enough to get good confidence results.

Recently a new method was proposed by A. Lee et al [4] to solve this issue. Their speech recognition decoder is based on a tree-trellis search [5][6], a typical two pass search based on tree lexicon. Instead of using n-best list or word graph after decoding, their method uses the local word graph made of partial sentence hypotheses and expanded words during the stack decoding process to compute local word posterior probabilities. The authors come to a conclusion that the method can produce confidence scores without searching for n-best while keeping its quality.

Considering much approximation in the computation of the local word posterior probability (LWPP) in the above method, in this paper we intend to improve the LWPP based confidence measure by combining it with some other predictors that can be computed real-time. We constructed a series of predictors and use decision tree to combine them with LWPP.

For comparison, the similar experiments are also carried out with word posterior probability from n-best list.

In section 2, the construction of predictors and decision tree are described. In section 3, the experiments on different combination of predictors are done. The conclusion is given in section 4.

2. CONSTRUCTION OF PREDICTORS AND DECISION TREE

This paper constructs 12 predictors, which are listed in Table 1.

There are two kinds of predictors for word posterior probability. One is local word posterior probability (LWPP) that is computed during stack decoding. The other, NBWPP, is computed from n-best list.

Table 1: Predictors for confidence measure

	Predictor	meaning
1	WLength	Number of syllables of word
2	WDuration	Duration of word
3	SpeakingR	WLength/WDuration
4	FrameAC	Acoustic score per frame of word
5	LMScore	Language score of word
6	LMType	Language model type of word (bigram or trigram, backoff information)
7	LWPP	Local word posterior probability computed during stack decoding
8	LPLeft	LWPP of last word
9	LPRight	LWPP of next word
10	NBWPP	Word posterior probability from n-best list
11	NBPLeft	NBWPP of last word
12	NBPRight	NBWPP of next word

2.1. Computation of LWPP

Given an acoustic feature vector X, then the posterior probability of a word w starting with time t_a and ending with time t_e can be formulated by

$$p(w,t_a,t_e \mid X) = \sum_{W \in W_{w(t_a,t_e)}} \frac{p(X \mid W)p(W)}{p(X)} = \sum_{W \in W_{w(t_a,t_e)}} \frac{e^{f(W)}}{p(X)} \quad (1)$$

where $f(W)$ denotes the likelihood of sentence hypothesis W that contains (w,t_a,t_e) .

$P(X)$ should be computed by summing up probabilities of all possible sentence hypotheses. For n-best list, the result is a simple summation of probabilities of all n-best sentences [3]. For word graph, forward-backward algorithm [1] is normally applied to get the summation of probabilities of all sentences that the word graph contains.

In this paper, we use a small word graph generated during stack decoding to compute word posterior probability. The method was first proposed recently by A. Lee et al [4].

The decoder in this paper is based on Julius 3.4.2 [5] with some changes made by us, which is a two-pass search based on tree lexicon [6]. 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. In stack decoding, when connecting a word $w_{(n,j)}$ $(j=1,2,...,m)$ to a existed partial sentence hypothesis $w_1^{n-1} = w_1, w_2, ..., w_{n-1}$, the likelihood of a potential sentence hypothesis is calculated as a decoder objective function by

$$f(w_1^{n-1},[w_{(n,j)};\tau_{(n,j)},t]) = g(w_1^{n-1},t) + \hat{h}(w_{(n,j)},t) \quad (2)$$

$$f(w_1^{(n,j)}) = \max_{0 \le t_{(n,j)} < T} f(w_1^{n-1},[w_{(n,j)};t_{(n,j)}]), \quad (j=1,2,...,m) \quad (3)$$

where $w_{(n,j)}$ $(j=1,2,...,m)$ denotes the connected word. t denotes the connecting time between the partial sentence hypothesis and the connected words. $g(w_1^{n-1},t)$ denotes the likelihood at the connecting edge of the partial sentence hypothesis on time t, and $\hat{h}(w_{(n,j)},t)$ denotes the heuristics likelihood of the connected word $w_{(n,j)}$ $(j=1,2,...,m)$ at time t on the word trellis. After the computation of formula (3), t is split into different value $t_{(n,j)}$ $(j=1,2,...,m)$ for responding connected word $w_{(n,j)}$ $(j=1,2,...,m)$.

After the above computation a small local word graph as Figure 1 is generated.

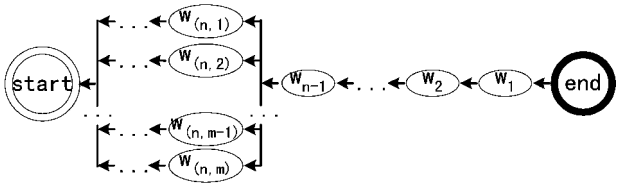

Figure 1 Local word graph during stack decoding

The word posterior probability of each $w_{(n,j)}$ $(j=1,2,\ldots,m)$ can be calculated from this small local word graph by

$$p(w_{(n,j)},\tau_{(n,j)},t_{(n,j)} \mid X) = \sum_{W \in W_{(w(n,j),\tau(n,j),t(n,j))}} \frac{p(X \mid W)p(W)}{p(X)} = \frac{e^{f(w_1^{(n,j)})}}{p(X)} = \frac{e^{g(w_1^{(n-1,j)},t_{(n,j)})+h(w_{(n,j)},t_{(n,j)})}}{\sum_{k=1}^{m} e^{g(w_1^{(n-1,k)},t_{n,k})+h(w_{n,k},t_{n,k})}}, (j=1,...m) \quad (4)$$

where $P(X)$ is computed by summing up probabilities of all heuristic paths in the local word graph. Compared with word graph in [1], this graph is very small.

2.2. Computation of other predictors

Word posterior probability from n-best list (NBWPP) is computed in traditional way with n equal to 100 [3].

Predictor 8, 9, 11, 12 are LWPP or NBWPP of adjacent words.

The other predictors are empirical ones that can be obtained at nearly no cost of time.

2.3 Construction of decision tree

The decision trees are made using a data-mining tool called C4.5R8 [7]. The trees are used to combine different predictors to decide if a word has been correctly recognized.

A word is a case and the predictors are attributes of the case in C4.5. The training cases are split at each tree node under a predefined criterion by a node question about a certain attribute. In our experiments, gain ratio criterion is employed, which is default in C4.5.

$$gainRatio = \frac{gain(X)}{splitInfo(X)} \quad (5)$$

where *gain(X)* denotes the information gain by partitioning test set in accordance with X (a question about a particular attribute). s*plitInfo(X)* denotes the quantity of information when splitting test set into subsets with X.

Cross validation and pruning methods are used to optimize the tree.

A word can be annotated as " hit" or "els" (which mean "correctly recognized" or "insertion &substitution error " respectively in our experiments) when reaching a tree leaf via the constructed decision tree.

3. EXPERIMENTS AND DISCUSSIONS

In this section, we use decision tree method to combine predictors. The experiments of different combination are done for comparison.

3.1 Experimental condition

Speech signals are digitized with 16kHz sampling and 16bit quantization. Feature vectors have 39 elements consisting of 12 MFCC, 0'th cepstral parameter and their delta and acceleration coefficients.

Acoustic model is triphone HMMs with 3 states per model trained with HTK3.0 [8]. The language models are made by using CMU SLM Toolkit v2.0, with ten years "*renmin daily*" corpus as training data. The vocabulary size is 60k, constructed based on the frequency of words appearing in the training data. The decoder is based on julius3.4.2 (some changes made by us). The language model weights and the insertion penalties are chosen to maximize the recognition accuracy.

The training cases for the decision tree come from the recognition result for 8 speakers (4 females: f65, f66, f67, f67, f68; 4 males: m65, m66, m67, m68; about 600 sentences per speaker) in "863" speech recognition database [9](a database widely used in Chinese speech recognition), 37226 words in all. The test cases are from another 8 speakers (4 females: f93, f94, f95, f96; 4 males: m93, m94, m95, m96) in "863" database, 37573 words in all.

3.2 Evaluation criterion

Each word is annotated as "hit" or "els" by constructed decision tree, which denotes "correctly recognized" or "insertion&substitution error" respectively. Word deletions are not considered.

Confidence error rate (CER) is computed as the evaluation criterion of confidence measures.

A baseline CER is given by the number of insertions and substitutions, divided by the number of recognized words (obtained by tagging all words as *hit* when recognition correct rate is larger than 50%).

$$baseline\ CER = \frac{\#\ insertions + \#\ substitutions}{\#\ recognized\ words} \quad (6)$$

Two kinds of decision error, false acceptance and false rejection are used to give confidence error rate:

$$CER = \frac{\#\ false\ acceptances + \#\ false\ rejections}{\#\ recognized\ words} \quad (7)$$

3.3 LWPP based experiments

When predictor LWPP is used as confidence measure singly, the value of LWPP that minimize the CER of training set is used as the threshold to split words in test set to calculate the CER for test set.

Then LWPP and other predictors (predictor 1,2,3,4,5,6,8,9 in Table 1), 9 predictors in all, are used to construct a decision tree. After pruning, the best decision tree contains 5 of the 9, which are, by order of importance: LWPP, WLength, LPLeft, LPRight, LMScore. The tree is shown as Figure 2. Compared with the result of using only LWPP as confidence measure (T3 in Table 2), the CER of the decision tree (T5 in Table 2) obtains an improvement of 18.9% relatively.

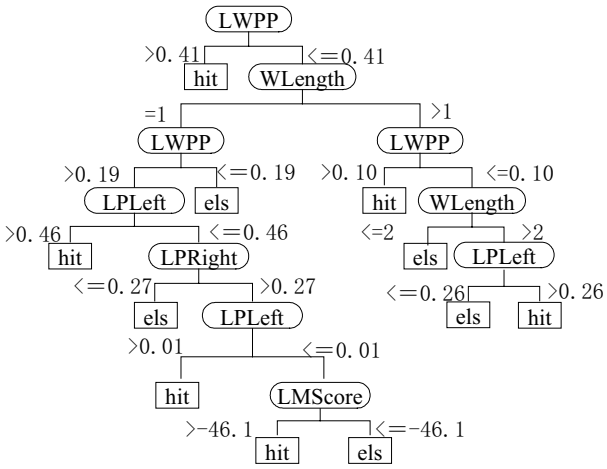

Figure 2 Decision tree for LWPP based combination

3.4 N-best list based experiments

For comparison, the similar experiments are done with predictor NBWPP as with LWPP. First the experiment of using NBWPP singly as confidence measure is done. Then NBWPP and predictors (1,2,3,4,5,6,11,12 in Table 1), 9 predictors in all, are used to construct a decision tree. The best tree contains 6 of the 9, which are, by order of importance: NBWPP, WLength, LMScore, SpeakingR, NBPLeft, NBPRight.

Compared with the result of using only NBWPP as confidence measure (T2 in Table 2), the CER of the decision tree (T4 in Table 2) obtains an improvement of 14.1% relatively.

3.5 Experiment on combination of all predictors

All predictors in Table 1 are used to construct the decision tree. The best tree contains 7 of the 12, which are, by order of importance: LWPP, WLength, NBWPP, LPLeft, LPRight, WDuration, NBPLeft.

The combination (shown in T6 of Table 2) does not provide significant improvement compared with LWPP based tree (T5).

Table 2 and Figure 3 show the comparison of confidence performance of different predictor combinations (base: baseline *CER*; T2: NBWPP singly; T3: LWPP singly; T4: NBWPP + predictor 1,5,3,11,12; T5: LWPP + predictor 1,8,9,5; T6: LWPP + predictor 2 + NBWPP + predictor 8,9,2,11).

Table 2: Confidence error rates in [%] for different combination of predictors

Speaker	base	T2	T3	T4	T5	T6
F93	32.9	24.6	23.1	21.0	18.8	18.8
F94	37.3	28.2	23.3	21.1	17.4	17.4
F95	22.9	16.8	18.4	17.3	14.8	14.4
F96	37.1	27.2	24.8	22.8	19.7	19.2
M93	26.6	19.7	19.0	18.3	15.7	15.0
M94	29.1	22.8	20.9	20.2	17.7	17.4
M95	25.7	19.0	18.8	17.6	15.3	14.6
M96	44.5	33.2	28.7	26.6	23.9	23.5
Total	32.2	24.1	22.2	20.7	18.0	17.6

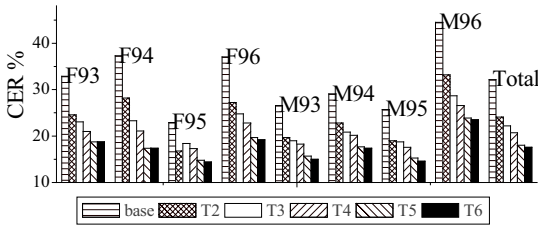

Figure 3 Comparison of confidence performance of different predictor combinations

4. CONCLUSION

The experimental results show that local word posterior probability (LWPP) computed from word expansion during stack decoding can obtain a significant improvement in confidence performance by using decision tree to combine some other real-time predictors, the relatively important ones among which are word length and LWPPs of adjacent words.

The experimental results also show that, compared with word posterior probability from n-best list (NBWPP), LWPP remains better after combining with other predictors.

The combination of LWPP and NBWPP with other predictors does not produce significant improvement.

5. REFERENCES

[1] F. Wessel, K. Macherey, and R. Schlueter. "Using word probabilities as confidence measures," Proc. ICASSP, Seattle, pp. 225–228, 1998.

[2] F. Wessel, "Word Posterior Probabilities for Large Vocabulary Continuous Speech Recognition," PhD thesis, Aachen, Germany, January 2002.

[3] F. Wessel, K.Macherey, and H. Ney, "A comparison of word graph and N-best list based confidence measures," Proc. ICASSP, Istambul, pp. 1587–1590, 2000.

[4] Akinobu Lee, Kiyohiso Shikano, Tatsuya Kawahara, "Real-Time Word Confidence Scoring using Local Posterior probabilities on Tree Trellis Search," Proc. ICASSP, Quebec, pp. 793-796, 2004.

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

[6] 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, Toronto, pp. 705–708, 1991.

[7] J. R. Quinlan, C4.5: Programs for Machine Learning, Morgan Kaufmann, California, 1996.

[8] http://htk.eng.cam.ac.uk/

[9]http://www.cass.net.cn/chinese/s18_yys/yuyin/product/product_2.htm.