Word Level Confidence Annotation using Combinations of Features

Rong Zhang and Alexander I. Rudnicky

School of Computer Science, Carnegie Mellon University
5000 Forbes Avenue, Pittsburgh, PA 15213

{rongz,air}@cs.cmu.edu

ABSTRACT

This paper describes the development of a word-level confidence metric suitable for use in a dialog system. Two aspects of the problems are investigated: the identification of useful features and the selection of an effective classifier. We find that two parse-level features, Parsing-Mode and Slot-Backoff-Mode, provide annotation accuracy comparable to that observed for decoder-level features. However, both decoder-level and parse-level features independently contribute to confidence annotation accuracy. In comparing different classification techniques, we found that Support Vector Machines (SVMs) appear to provide the best accuracy. Overall we achieve 39.7% reduction in annotation uncertainty for a binary confidence decision in a travel-planning domain.

1. INTRODUCTION

Accurate confidence annotation is a key capability for dialog systems, since acting upon a misunderstood input may incur a high cost to the user (either though an undesired side-effect or through time wasted on correction). Current confidence annotation approaches lack the desired level of accuracy. We approach the problem by increasing the variety of information sources used for annotation and by improving classification accuracy.

Chase [1] proposed the following framework for incorporating a confidence metric into a recognition system: (1) At what level should the confidence annotation be made; (2) What is the right way to define what is an error and what isn't; (3) What features are useful and how useful; (4) How to build a model combining the various features to create a confidence annotation; (5) How to measure the goodness of the feature and model.

The answers to these questions depend on the particular application that incorporates the confidence annotator. For the CMU Communicator system [2], a telephone based dialog system that supports planning in travel domain, both the utterance level and word level confidence annotation are used. The former measures the confidence of the whole utterance, and the latter supplies the reliability description of each single word. In this paper we focus on word-level confidence annotation and more specifically on a binary tagging of word hypotheses, as *correct* or *incorrect*, according to whether the system believes it is a correct decoding result or not.

The key problem in confidence annotation is the selection of effective features [1][3][4][5][6][7] and a variety of features have been proposed. These features can be assigned to one of four categories depending on the information source: acoustic, language model, N-best list or word lattice and other. All are based on information from the decoder and moreover have the disadvantage that they unavoidably overlap in the information that they use, as is apparent in the common observation that the performance achieved by all the features together isn't much better than that with only the best feature. Secondly, these features are for the most part redundant with the information used to generate hypotheses in the first place and so contribute little new information, particularly at the acoustic level.

Given the limitations of purely decoder-based features, other information source can be considered. Syntactic constraints and grammar rules have proved to be helpful for selecting the best hypothesis from an N-best list [8][9]. It's reasonable to apply this knowledge to the word-level confidence annotation because we know that the correct recognition result tends to be grammatical while an incorrect hypothesis is often (but not always) ungrammatical. For large-vocabulary speech transcription, the absence of a covering grammar limits the use of this knowledge source. For limited domains (as typically is the case for a dialog system) a high-coverage grammar is available, since parsing is a necessary processing step. In this paper, we describe two parser-level features, Parsing-Mode and Slot-Backoff-Mode.

Confidence annotation can be treated as a problem in pattern classification and a large variety of classification approaches can be considered for our problem. A recent technique, Support Vector Machine [10][11], has been shown to achieve better performance for long-standing problems than the traditional approaches such as Neural Networks and Decision Tree. We assess the suitability of SVMs for the confidence annotation problem.

The data set used in our experiments was taken from the speech and log files produced by the CMU Communicator system (see e.g. [2]). A total of 1781 utterances were used, all of which were collected from an evaluation during July 2000. As a result of the evaluation design, all data is from first-time users of the system. The first 1000 utterances were used as the training set and the remaining as the test set. All experiments were performed using the Sphinx-2 [12] decoder.

2. FEATURES

In this section, we describe the decoder-based features and introduce two parse-level features.

2.1 Decoder-based Features

We investigated nine decoder-based features, in four categories. Most of these have been previously described in the literature and have been shown to be promising. All were re-implemented for the purpose of this study, based on their published descriptions.

Acoustic Features

1. *PercPhAll-Frame* [1]: The percentage of frames in the hypothesized word which base phones match the base phones in the phone-only decoding.

2. *PercPhAll-Phone* [1]: Similar to the previous feature except the percentage is computed for phones rather than frames.

3. *Normalized-Acoustic-Score* [1]: The ratio between the acoustic scores from the normal decoding and phone-only decoding.

Language Model Features

4. *LM-Backoff-Mode* [13]: The back-off mode of the trigram language model. This feature is computed over a four-word window that includes the current word, as well as one preceding word and two succeeding words. For each word in the window we note whether the 1, 2 or 3-gram value was used to compute the language model score. For each such pattern in the training set we compute the error probability of the current word and threshold this value for the confidence decision.

Word Lattice Features

5. *Lattice-PWP-Acoustic* [6]: The log posterior word probability computed from the word lattice by summing and normalizing the scores of paths passing through the hypothesized word. It is computed using only the acoustic score.

6. *Lattice-PWP-LM* [6]: The log posterior word probability computed from the word lattice but using only the language model score.

7. *Lattice-PWP* [6]: *Lattice-PWP-Acoustic + Lattice-PWP-LM*.

N-Best List Features

8. *N-Best-Homogeneity* [1]: The ratio between the score of the paths containing the hypothesized word to the total path score of N-Best list.

9. *N-Best-Word-Rate* [1]: The ratio between the number of the paths containing the hypothesized word to the total number of paths in the N-Best list.

We use the Annotation Error Rate to measure the performance of each feature:

$$Annotation\ Error\ Rate = \frac{Number\ of\ incorrectly\ assigned\ tags}{Total\ number\ of\ tags}$$
(2.1)

The annotation error rate is often compared with the revised Word Error Rate, which is different to its namesake for speech recognition:

$$Word\ Error\ Rate = \frac{Substitutions + Insertions}{Total\ number\ of\ words\ in\ hypotheses}$$
(2.2)

Working only on the hypotheses, the confidence annotator is unable to tell whether something is missing or not. So the revised word error rate doesn't consider the deletion errors and changes the denominator to the number of words in hypotheses rather than that in references. If no confidence annotation is made to the decoding result, namely, we simply label every word hypothesis as correct; the annotation error rate would be equal to the revised word error rate. The revised word error rate of the test set is 30.2%.

Feature parameters, except LM-Backoff-Mode, were trained using a standard maximum likelihood procedure, minimizing the number of mislabels in the training set. For LM-Backoff-Mode, all of its possible combination modes are labeled according to the distribution of correct and incorrect hypotheses belonging to that mode and classification is done by table lookup. Table 2.1 shows the annotation error rate and the corresponding false alarm and missing rate of each feature (the false alarm rate and missing rate are normalized by the total number of words in hypotheses).

Feature	Annot. Error	False Alarm	Missing
Lattice-PWP	20.8%	6.0%	14.8%
Lattice-PWP-LM	24.3%	6.5%	17.8%
LM-Backoff-Mode	26.1%	5.7%	20.4%
N-Best-Homogeneity	26.8%	2.3%	24.5%
Normalized-Acoustic-Score	29.4%	2.6%	26.8%
PercPhAll-Frame	30.2%	0%	30.2%
PercPhAll-Phone	30.2%	0%	30.2%
Lattice-PWP-Acoustic	30.2%	0%	30.2%
N-Best-Word-Rate	30.2%	0%	30.2%

Table 2.1 Performance of decoder-based single feature

The result in Table 2.1 suggests that the language model features outperform acoustic features. One source of evidence comes from the comparison between LM-Backoff-Mode and Normalized-Acoustic-Score, and the comparison between Lattice-PWP-LM and Lattice-PWP-Acoustic. Obviously the former performs much better than the latter. More evidence is obtained from the analysis of Lattice-PWP, the best feature in Table 2.1. In computing this feature, the weight for language model is increased to six times the value used in normal decoding. Therefore a reasonable conclusion is that the success of Lattice-PWP should be mainly due to the language model score.

2.2 Parser-based Features

Good performance for language model features suggests the importance of linguistic, syntactic and semantic knowledge. Besides the language model, features of the parse of the decoding hypothesis suggest themselves as a new information source. Given the assumption that the speaker is cooperative, the correct recognition results would be more grammatical than the incorrect recognition result. Analysis of our data set shows that the ratio between parsed and unparsed words for the class of correct recognition result is 16:1 while the ratio is reduced to 1.8:1 for the class of incorrect recognition result.

For a large-vocabulary speech recognition system, the absence of accurate parsing makes the incorporation of syntactic

knowledge difficult. However, this isn't a problem for dialog system in which parsing is a necessary stage. Our Communicator system uses the Phoenix semantic parser [14]. Its outcome isn't a complete parse tree but a sequence of slots that bracket the concepts extracted from the utterance. This approach typically produces a robust parse of the input and is therefore suitable as a source of information about word-level confidence. We therefore considered two additional features based on the parse result.

10. *Parsing-Mode*: This feature indicates if a word is parsed by the grammar (i.e., bracketed as part of a slot). For a parsed word, it further indicates the position of the word within the slot, either edge or middle.

11. *Slot-Backoff-Mode*: Using a bigram language model for the slots, each parsed word is assigned the back-off mode of the slot that it belongs to. This feature is computed on a two-word window that contains both the current word and the next word (see LM-Backoff-Mode, above).

The simplicity of these features may lead one to doubt their effectiveness. However our experiments shows they are useful features for confidence annotation. Table 2.2 gives their annotation error rates on the test set. Compared with the features displayed in Table 2.1, the performance of parse-level features exceeds all but Lattice-PWP. Their individual performance is encouraging. The question of whether they can work with other features is addressed in the next section.

Features	Annot. Error	False Alarm	Missing
Parsing-Mode	23.8%	6.5%	17.3%
Slot-Backoff-Mode	23.6%	6.3%	17.3%

Table 2.2 Performance of parse-level single feature

3. CLASSIFIERS

In this section, we describe our experimental results for SVM as well as two traditional classifiers, Decision Tree and Neural Nets. The experiments were conducted on the same data set used in the previous experiments for decoder-level features. We use the same annotation error rate (formula 2.1) as the metric.

3.1 Experiments using Decision Trees and Neural Nets

We first describe classification accuracy using two widely used approaches, Decision Trees and Neural Nets. For the decision Tree classifier we further investigated the impact of the choice of objective function on accuracy. We compared two criteria for choosing the next feature when building the Decision Tree: Decision Tree (1) uses a standard information gain criterion and Decision Tree (2) uses the word error rate. We used a single BP Network, with one hidden layer containing 50 nodes.

Table 3.1 presents the performance of the Decision Tree and Neural Network classifiers using only decoder-level features. Recalling the error rate achieved by the best feature, Lattice-PWP, which is 20.8%, there is only a small improvement gained by adding other decoder-based features (and no improvement for Decision Tree (1) at all). The reason for this phenomenon has been mentioned before. All of these features come from the decoder and they overlap each other on the information that they contain. The feature Lattice-PWP has

information from both acoustic model and language model while other features only represent one aspect. So the importance of other decoder based features is weakened.

Classifier	Without Parser-based Features		
	Annot. Error	False Alarm	Missing
Decision Tree (1)	21.4%	7.8%	13.6%
Decision Tree (2)	20.6%	8.3%	12.3%
Neural Network	20.4%	5.4%	15.0%

Table 3.1 Results of NN and DT without parse-level features

Classifier	With Parser-based Features		
	Annot. Error	False Alarm	Missing
Decision Tree (1)	20.3%	5.9%	14.4%
Decision Tree (2)	20.1%	8.0%	12.1%
Neural Network	19.3%	5.5%	13.8%

Table 3.2 Results of NN and DT using parse-level features

It's perhaps surprising to see that the annotation error rate for Decision Tree (1) is higher than the error rate achieved by the single feature Lattice-PWP. There are two reasons that would explain this anomalous result. The first one is concerned with a known limitation of Decision Tree classifiers, which is that they can easily overfit the training data. The other possible reason is the mismatch between the criterion to build the Decision Tree and the metrics to evaluate it. Decision Tree (1) is constructed on the basis of the information gain, but its performance is measured by the error rate. So in Decision Tree (2) we used the error rate directly as the criterion to choose feature and got a better result. However, it doesn't mean the error rate is superior to the information gain as the criterion to construct the tree. If an entropy related metric is used to evaluate the performance, we may get a totally different result. We also observed that Decision Tree (2) tends to create fewer nodes than Decision Tree (1).

Table 3.2 shows the performance of the annotator incorporated with the parse-level features. With the help of parse-level features, each of the three approaches produced some improvement on the annotation error rate. The relative reductions of error rate for these three approaches are 5.1%, 2.4% and 5.4% respectively. However, the missing rate of Decision Tree (1) doesn't decrease with its error rate.

3.2 Experiments on SVM

The basic idea of SVM is to seek an optimal hyper-plane that separates two classes with maximum margin. SVM solves the problem that in some cases there isn't a linear separating hyper-plane in an ingenious way: mapping the samples to a higher dimension space using a *kernel function*, and seeking a hyper-plane in that space. For the detail of SVM see [11], which has an excellent introduction to this approach. We evaluated five different kernel functions [15] for confidence annotation:

Dot: $$K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i \bullet \mathbf{x}_j \qquad (3.1)$$

Poly: $K(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i \bullet \mathbf{x}_j + 1)^d$ (3.2)

Radial: $K(\mathbf{x}_i, \mathbf{x}_j) = exp(-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2)$ (3.3)

Neural: $K(\mathbf{x}_i, \mathbf{x}_j) = tanh(a\mathbf{x}_i \bullet \mathbf{x}_j + b)$ (3.4)

ANOVA: $K(\mathbf{x}_i, \mathbf{x}_j) = (\sum_p exp(-\gamma(\mathbf{x}_{ip} - \mathbf{x}_{jp})^2))^d$

(3.5)

Kernel Function	Without Parser-based Features		
	Annot. Error	False Alarm	Missing
Dot	21.9%	4.0%	17.9%
Polynomial	21.4%	4.1%	17.3%
Radial	19.9%	6.3%	13.6%
Neural	22.3%	3.9%	18.4%
ANOVA	19.7%	6.8%	12.9%

Table 3.3 Results for SVM without parse-level features

Kernel Function	With Parser-based Features		
	Annot. Error	False Alarm	Missing
Dot	19.8%	4.2%	15.6%
Polynomial	19.2%	5.1%	14.1%
Radial	19.0%	7.2%	11.8%
Neural	19.9%	3.7%	16.2%
ANOVA	18.2%	5.5%	12.7%

Table 3.4 Results for SVM using parse-level features

Table 3.3 and 3.4 present the experiment results for these five kernel functions. By comparing Table 3.1 with Table 3.3, and Table 3.2 with Table 3.4, one can note that SVM using the kernel functions ANOVA and Radial perform better than Decision Tree and Neural Network measured on annotation error rate. At the same time their corresponding missing rates are relatively low. The other three kernel functions, although they didn't work very well in the experiment, also provide acceptable performance. The experimental results validate the utility of the parse-level features. For each kernel function, the annotation error rate and the corresponding missing rate are identically reduced by using these features. For ANOVA, the relative reduction of error rate is 7.6%.

As is the case with Neural Networks, tuning the parameters of an SVM classifier is a non-trivial task. It should also be borne in mind that some kernel functions are not very robust and that a small change in parameters may produce a noticeably different result.

4. SUMMARY

We investigated the use of various features for confidence annotation in a dialog domain. We described the application of parse-level knowledge for confidence annotation. Given their simple form and easy computation, the performance of the

parse-level features is very encouraging. Such results indicate that the information from knowledge sources other than the decoder could benefit confidence annotation. We also provided the experiment results for SVM-based annotation. The performances of the kernel function ANOVA and Radial demonstrate that SVM is a reasonable choice of classifier for confidence annotation.

REFERENCES

[1] Lin Chase, "Error-Responsive Feedback Mechanisms for Speech Recognition". Ph.D. Thesis, Carnegie Mellon University, April 1997.

[2] Alex Rudnicky, Eric Thayer, et al., "Creating Natural Dialogs in the Carnegie Mellon Communicator System". Proc. EuroSpeech-99, Vol. 4, pp. 1531-1534, 1999.

[3] Mitch Weintraub, Francoise Beaufays, et al., "Neural Network Based Measures of Confidence for Word Recognition". Proc. ICASSP-97, Vol. 2, pp. 887-890, 1997.

[4] Thomas Schaaf and Thomas Kemp, "Confidence Measures for Spontaneous Speech Recognition". Proc. ICASSP-97, Vol. 2, pp. 875-878, 1997.

[5] D. Bansal and M. K. Ravishankar, "New Features for Confidence Annotation". Proc. ICSLP-98, No. 829, 1998.

[6] Frank Wessel, Klaus Macherey and Ralf Schluter, "Using Word Probabilities as Confidence Measures". Proc. ICASSP-98, Vol. 1, pp. 225-228, 1998.

[7] Ruben San-Segundo, Bryan Pellom and Wayne Ward, "Confidence Measures for Dialogue Management in the CU Communicator System". Proc. ICASSP-00, Vol. 2, pp. 1237-1240, 2000.

[8] Manny Rayner, David Carter, Vassilios Digalakis and Patti Price, "Combining Knowledge Sources to Recorder N-Best Speech Hypothesis Lists". Proc. ARPA HLT Meeting, 1994.

[9] E. Brill, R. Florian, J. C. Henderson, and L. Mangu, "Beyond N-grams: Can Linguistic Sophistication Improve Language Modeling". Proc. COLING-ACL 1998.

[10] V. Vapnik, "The Nature of Statistical Learning Theory". Springer Verlag, 1995.

[11] Christopher Burges, "A Tutorial on Support Vector Machines for Pattern Recognition". Data Mining and Knowledge Discovery, 2(2), 1998.

[12] K. Lee, "Large Vocabulary Speaker-Independent Continuous Speech Recognition: The SPHINX System". Ph.D. Thesis, Carnegie Mellon University, April 1988.

[13] D. Bansal and M. K. Ravishankar implemented this feature in the CMU Communicator system, 1998.

[14] Sunil Issar and Wayne Ward, "Recent Improvement in the CMU Spoken Language Understanding System". Proc. ARPA HLT Workshop, pp. 213-216, 1994.

[15] Stefan Ruping, "mySVM Manual". University of Dortmund, October, 2000.