# New Word-Level and Sentence-Level Confidence Scoring Using Graph Theory Calculus and its Evaluation on Speech Understanding

*Javier Ferreiros, Rubén San-Segundo, Fernando Fernández, Luis-Fernando D'Haro, Valentín Sama, Roberto Barra and Pedro Mellén*

Speech Technology Group. Dept of Electronic Engineering. Universidad Politécnica de Madrid
E.T.S.I. Telecomunicación. Ciudad Universitaria s/n, 28040-Madrid, Spain
{jfl, lapiz, efhes, lfdharo, vsama, barra, mellen}@die.upm.es

## Abstract

A lot of work has been devoted to the estimation of confidence measures for speech recognizers. In the quite extended case where a word-graph speech recognizer is in use, we will present new confidence measures employing the graph theory that shows us how to estimate some interesting characteristics about the different paths through the graph that constitute the recognition solutions, without the need of expanding them all.

We will take advantage of some of these features to generate confidence scores both at the word and sentence level. We will also compare this new confidence scoring to more traditional ones and will find similar behavior with less computational load and with an increase in the simplicity of the approach that will lead to more generalization power of the confidence estimation to different applications of the recognizer.

## 1. Introduction

Some good studies exist on sentence and word level confidence scoring for speech recognition and its applications to speech understanding systems [1], [3]. The search for effective confidence measures usually goes through the integration of several features more or less closely related to the idea of confidence as some kind of combination, usually a neural network [4], [5]. These combinations suffer from the need to tune the combination parameters to new applications.

In this paper we are presenting a new confidence scoring methodology based on some basic graph calculus [2] that exhibits the nice property of eluding the presence of tuning parameters.

In the experimental work we will show similar behavior of this new scoring to more complex traditional combinations of parameters in the task of predicting word and sentence error rates. We also show similar performance when this scoring is used in a speech understanding task as confidence filters to refine the interpretation of the sentences.

## 2. Some graph theory basics revisited

The recognizer generates, as the best match to the acoustic sequence belonging to a sentence, a word graph composed of a set of arcs that connect two special nodes (in the sense that they are not related to any existing word in the vocabulary): the START node and the END node. From the START to the END, a map of paths following arcs that contain words from the vocabulary represents the solution obtained by the recognition algorithm. This kind of graphs is called simple directed graphs as they do not contain loops and each arc has a specific direction from one node to another one. These characteristics will be fundamental in the discussion about the mathematics we will employ.

We need now to define the adjacency matrix, let call it A. We will simplify the definition saying that the adjacency matrix A for a graph is a binary-valued matrix which element (i,j) is 1 if it there exists a connection between node i and j, otherwise it is 0. It is straightforward to reach the conclusion that if the graph has no loops, the diagonal will be composed of 0s.

Also, as the graphs obtained as the result of the recognizer are constituted by arcs going from words appearing before in time to other following words and we will order the numbering following this criterion, we notice that, in this case, the adjacency matrices will have nonzero elements only in the upper-triangular part (with a null diagonal). Another important detail is that these adjacency matrices will be quite sparse (indeed, as more sparse as better is the recognizer or the acoustic sequence is nearer to usual examples of the application learnt by the system) and the more common appearance for these matrices is some 1s disposed near a diagonal of 0s.

By its definition, we can see that the adjacency matrix let us know if there is a simple connection between two nodes. Now we will make some reflections about the meaning of the successive powers of the adjacency matrix. The square of the adjacency matrix would be composed by the number of two-step paths form one node to another one. Following this reasoning, we will reach to the conclusion that the number of paths with *s* steps length from node i to node j will be the element in the position (i,j) of the matrix $A^s$, i.e., the s-th power of the adjacency matrix.

Now, consider the output of a word graph recognizer. If we write down the adjacency matrix A, we will obtain the connections between words with which we could redraw the whole graph. We can follow by calculating the successive powers of the adjacency matrix $A^s$ and eventually will reach a certain $s=s_{min}$ where the element $A^s$(START,END) becomes different to zero (and not necessarily 1, but any positive number). This represents the first occasion in which there exist one or more connections from the START to the END and we may save this $s_{min}$ number as the length of the shortest solution paths, the minimum length of a recognized hypothesis. If we continue the calculation of successive powers of the adjacency matrix, we will reach another $s=(s_{max}+1)$ where every element of the $A^s$ matrix becomes zero. The meaning is that there is no path with length $(s_{max}+1)$ or, what it is the same, the largest paths have a length of $s_{max}$ steps. One may think that this could be a path between any two internal nodes, but this can not be the case: since the graph is acyclic and directed from START to END, the largest paths HAVE to be between START and END and must actually be members of the solution family.

In the following descriptions we will also use another matrix computed as the sum of all powers of the adjacency matrix:

$$S = \sum_s A^s \tag{1}$$

The elements of this matrix will keep the total number of paths (of any length) existing between two any nodes.

## 3. Keeping computational demands under reasonable limits

We have just shown that we need to accumulate the successive powers of the adjacency matrix for a graph. Some may think that this will lead to computational limitations easily. In this section we show some optimizations and recommendations to obtain the calculations without computational explosions.

First of all, there are pathological cases where the graph may have an enormous size, like when the speaker utters a sentence completely out of the supposed domain. For these cases, a simple limit in the number of nodes of the maximum size graph we will accept to be processed is the right solution and we can assume directly that the confidence is low enough to reject the input that caused this giant graph.

Now, for the graphs we do process, there are some advises we can mention.

We just reviewed that the adjacency matrix is very sparse in the sense that only a few elements near the diagonal (and excluding the diagonal) will have non zero values. So, the wise way to manage this matrix is not as a bi-dimensional array of values but as a uni-dimensional array of concatenated lists of actually existing values. This coding scheme also applies to the successive powers matrix.

To obtain the successive powers of the adjacency matrix, we have to multiply the last power obtained by the adjacency matrix: $A^{s+1}=A^s \cdot A$.

Without any other consideration, we would tend to code this multiplication as two loops, one for the row index and another one for the column index controlling a third loop that would run trough the values with which we obtain one element of the product.

Again, there is a better way to code this part: we can run trough the uni-dimensional array of lists we just mentioned. Then, for any non empty list, we can run through the existing elements in the last calculated power. For each of them, we look again in the corresponding list to make the products by the existing elements, but in the adjacency matrix this time. We have to note that, because the adjacency matrix is binary valued, this product has not to be done, but we have only to accumulate the value read from the existing element on the last calculated power for the position existing in the adjacency matrix. With this trick, we can process rather big graphs without computational problems.

## 4. Graph features related to confidence

### 4.1. Word level confidence

We estimate word level confidence as the "purity" of each of the words. This purity is defined as the relative number of hypothesis in the graph that contain this word in the same position.

For the calculation of this parameter, we will make use of the accumulated-powers matrix S as defined in equation (1). If we index the element $S(START,w_i)$, we will get the number of paths of any length in the solution family coming from the START to the word $w_i$ in question. In the other hand, the element $S(w_i,END)$ is the number of paths from $w_i$ to the END. Thus, if we take the product $S(START,w_i) \cdot S(w_i,END)$ we will obtain an estimation of the total number of paths in the solution family going through $w_i$.

We need to keep in mind that here $w_i$ means a word in a particular position (for example, another $w_j$ may in fact be the same word, but in a different position). To obtain the purity of each word in the principal hypothesis, we need to know the total number of possible hypothesis that the graph comprises. This factor is easily obtained from the element S(START,END) and leads us to the estimation of the purity of each word in the principal hypothesis considering the graph as:

$$P_i = \frac{S(START, w_i) \bullet S(w_i, END)}{S(START, END)} \tag{2}$$

This purity of each word acts as a confidence measure because, when a word is correctly recognized, it would appear in all the expanded hypothesis from the graph in the same position. In the case where the recognizer would be less confident about this word, the number of hypothesis compiled in the graph divides between the possibility that in this position would appear word $w_i$ and the possibilities for other words in the same position.

One nice property of this confidence measure is that it has no tuning parameter and should behave the same even changing the application been worked out by the recognizer.

Basically, this confidence estimation follows the fact that the complexity of the graph goes in inverse proportion to the confidence of the recognizer in its delivered solution. In our case, we have estimated the "partial complexity" of the graph for each word as the proportion of alternatives to the word under inspection, as compiled in the graph, and the purity or confidence for this word as the complementary measure.

### 4.2. Sentence level confidence

For the sentence level we are describing first some features extracted from the graph calculus that will be used as inputs to a neural network confidence sentence scoring generator and in the next sections we will propose and compare this approach to the simple and more general approach of using the last feature alone, the average of word confidence scores generated as described in 4.1, as a confidence score for the whole sentence.

Next, we are describing some features computed from the graph that will be used by a neural network to give a confidence score to a sentence.

- Total number of possible hypothesis. The number of different hypothesis that could be obtained from the graph. It is simply obtained from the element S(START,END). The rationale behind this feature, once again, is that the number of different solutions in the graph (its complexity) will increase in inverse relation to the confidence.
- N-best computation reduction. This feature is used when we activate the generation of the n-best list of solutions (usually we extract 10 different solution) and is the ratio between the number of expansions needed to find these 10 different solutions and the total number of hypothesis in the graph. If this feature gives numbers near 1, it means that almost all hypothesis have been needed to be expanded to obtain de 10-best list (or maybe, the algorithm even could not find 10) and this intuitively is an indication of closely competing hypothesis and low confidence.
- Hypothesis length dynamic range. This is the ratio between the length of the largest hypothesis and the length of the shortest hypothesis. As we analyzed before, this can be obtained as smax / smin. If a sentence is badly recognized, a lot of different solutions with very varied lengths are obtained. In high confidence cases, this feature delivers values near 1.
- Percentage of words with high purity. Calculated as the number of words with purity values over 0.5.
- Average purity. The average of purity values for all words in the principal hypothesis. Although we present here this feature as input to the neural network, we will use also this one by itself as a confidence measure for the whole sentence.

## 5. Our graph-based confidence scoring proposal

As a summary, our proposal is to use the purity of each word in the principal hypothesis defined in 4.1 to estimate the confidence score for each word and the average of them all as the global sentence score. We will now proceed to evaluate this proposal against more classical neural net integration of several confidence related features to obtain both word level and sentence level confidence scores.

## 6. Experimental work

We have used our speech recognizer that is prepared to generate graphs and n-best lists as the result of recognition. The task is a spontaneous speech task of Air Traffic Control in radio channels limited to less than 4KHz bandwidth.

The experiments with confidence measures where two-fold. In first place, following the direction of analyzing the effects of using these confidence scores as predictors of the word and sentence level recognition errors and then in the direction of evaluating the effects of using the confidence scores in a speech understanding module to generate confidence scores for the slots in the generated frame and to refine them with filters based on the confidence of individual items.

### 6.1. Evaluation of confidence measures as error predictors

First, we are comparing the word purity as a confidence score to a neural network integration of several features (word score, average word score, score variance, score of the worst frame, score of the best frame, average difference to the best score, N-best purity, N-best scoring and also the graph purity). The performance of the purity alone compared to the neural network integration (although slightly worst) is not very different and make us think that most of the confidence information resides already in the graph purity. This can be seen in Table 1 where we present this comparison and also the recognition performance for this task.

*Table 1*: Word level confidence evaluation

| Experiment | % Performance | Band |
|---|---|---|
| Recognizer accuracy | 83.3 | ±0.51 |
| Graph purity | 86.88 | ±0.46 |
| NN integration | 88.31 | ±0.41 |

Something similar is observed for the sentence level confidence scoring. As it can be seen in Table 2, there is not a great gap between the performance of just using the average purity or a network integration of the graph features we presented for that case.

*Table 2*: Sentence level confidence evaluation

| Experiment | % Error |
|---|---|
| Average Graph purity | 10.66 |
| NN integration | 10.10 |

### 6.2. Evaluation of confidence measures for speech understanding

We now comment some results obtained when using these confidence scores in a speech understanding module. Our speech understanding module is based on context dependent rules applied on the principal hypothesis delivered by the recognizer. The words in this hypothesis are labeled with syntactic-semantic tags and now we accompany them with the word confidence scores and the global sentence score. The global sentence score is used simply as a multiplier to all the word scores in these experiments to give it some modulating capacity.

Speech understanding rules are written making use of some primitive functions in a dedicated library. In this primitives, we introduce the following simple confidence elaborating mechanism: For the items generated by the rule, a new confidence score is generated as the average confidence of the terms on which the primitive relies to do its job.

Executing the understanding module, we eventually arrive to a frame with a variable number of slots, each one qualified now with a confidence score. Then we have applied filters related to these confidences (basically removing interpretations with low confidence scores or with confidence scores far (and below) from their neighbors') and obtained the results in Table 3, where we have consigned the concept accuracy (paralleling the definition of word accuracy, but for the application concepts in the sentence).

*Table 3*: Speech understanding evaluation

| Experiment | % Concept accuracy |
|---|---|
| Without confidence | 76.57 |
| With NN confidence | 79.74 |
| With Graph confidence | 80.52 |

In this table 3 we may note that using confidences generated with graph calculus as we have introduced them, we obtain similar results than using more complicated scores coming form NN integration. And we would like to remark again that NN confidence means the use of a neural network, i.e., the tuning of the net weights to a particular application whereas the graph confidences have no tuning parameter to the application.

## 7. Conclusions

We have presented a new word and sentence level confidence scoring estimated as features extracted from a word graph recognizer through the application of some basic graph calculus.

The nice property is that these features elude the presence of tuning parameters and lead us in conclusion to confidence scores independent of the application and circumstances of the speech recognizer.

The experiments that we have carried out using this new confidence scoring had two aims.

- In the first place, to study the predicting power of the recognizer errors.
- In a second place, to study the help of the confidence measures when used in filters to refine the interpretation of the sentences in a speech understanding system.

For both cases, graph based confidence scoring revealed similar performance compared to a classical complex combination of features with neural networks.

## 8. Acknowledgements

This work has been partially funded by the Spanish Ministry of Science and Technology under contracts DPI2001-3652-C02-02 (URBANO-IVANHOE) and DPI2004-07908-C02-02 (ROBINT). The authors wish to thank all the rest of the Speech Technology Group for the continuous and fruitful discussion on these and many other topics.

## 9. References

[1] Timothy J. Hazen, Stephanie Seneff, and Joseph Polifroni., "Recognition confidence scoring and its use in speech understanding systems", *Computer speech and language, 16:49-67, 2002.*

[2] Kennet H. Rosen., "Discrete mathematics and its applications.", *Mathematics & statistics series. MacGraw-Hill International Editions, 1999.*

[3] Manhung Siu and Herbert Gish., "Evaluation of word confidence for speech recognition systems.", *Computer speecg and language, 4(13):299-319, 1999.*

[4] Bernd Souvignier and Andreas Wendemuth., "Combination of confidence measures for phrases.", *Proc. Eur. Conf. Speech Communication and Technology, pages 217-220, 1999.*

[5] Rong Zhang and Alexander I. Rudnicky., "Word level confidence annotation using combinations of features.", *Eursopeech, 2001.*