

Learning of Word Boundaries In Continuous Speech Using Time Delay Neural Networks

Colin Keng-Yan TAN

Laboratory for Computational Linguistics,
Department of Computer Science,
School of Computing,
National University of Singapore
ctank@comp.nus.edu.sg

Kim-Teng LUA

Laboratory for Computational Linguistics
Department of Computer Science,
School of Computing,
National University of Singapore
luakt@comp.nus.edu.sg

Abstract

This paper presents early research results for a method for training neural networks to segment previously unseen continuous speech data into words, without the use of a lexicon or a speech recognition engine, or any other forms of supervision. The initial word segmentation is derived through a simple and naïve method, and our method then leverages the time-invariant properties of the TDNN to derive better hypotheses, which is then used to recursively re-train the neural network.

1 Introduction

The ability to automatically learn word and phoneme boundary information from continuous speech input without guidance is particularly appealing. Word and phoneme segmentations are important for training acoustic models in speech recognition, but manually segmenting a speech database is a time consuming and expensive process. The ability to automatically segment speech information using minimal data is thus very useful. In addition it is interesting to study how computer systems can learn word boundaries spontaneously, as it would allow us to build systems that mimic the human ability to automatically induce new words from speech.

Various methods have been explored to solve this problem. Toledano et. al. (1998) present a method using a combination of Hidden Markov Models and fuzzy methods to segment phonemes for use in text-to-speech systems. Ramana and Srichand (1996) use pitch variances found in Hindi and German to segment words. More recently Laureys et. al. (2002) present a method that uses a forward-backward segmentation. A confidence interval is computed, and this is used to remove biases in the segmentation. Huk and Lee present a method to segment speech using a multi-layer perceptrons.

This paper explores the use of a Time-Delay Neural Network and simple speech features to recognize word boundaries. Since it is tedious and expensive to manually segment speech data, we will assume that only limited training data is available to initially train the TDNN. To support this, the TDNN is first trained on a small set of speech data. A pruning algorithm is applied to the trained neural network to reduce over-fitting caused by limited data.

Segmentation of new speech data is achieved through a bootstrapping algorithm. A naïve method is used to derive an initial word segmentation, which is then used to train the TDNN, together with the initial set of training data. This method then relies on the TDNN's ability to learn actual token (in our case, word) boundaries (See Waibel et. al. (1989)) to provide improved boundary hypotheses.

2 Method Description

2.1 Neural Network Architecture and Input Feature Selection

Standard multi-layer perceptrons (MLPs) assume that each input-output example pair in the training set is independent, and hence make minimal attempts to associate input-output pairs across time. This

assumption unfortunately does not hold as far as speech inputs are concerned. The Time-Delay Neural Network (TDNN) was introduced to make-up for the sometimes incorrect independence assumption made by MLPs.

In TDNNs training is not only confined to a particular input-output example pair, but is performed over examples in time. Input (and sometimes hidden) nodes have shift registers attached to them. By shifting previous inputs through the shift registers and inserting new inputs, the TDNN is able to generalize over time. All the weights connecting the input delays to the hidden layer (and perhaps the hidden-layer delays to the output layer) are tied together, meaning that they will always be updated with the same delta values during the backward propagation phase of the training algorithm, leading to time-invariance across the delayed inputs. This idea is illustrated in Fig. 1 below.

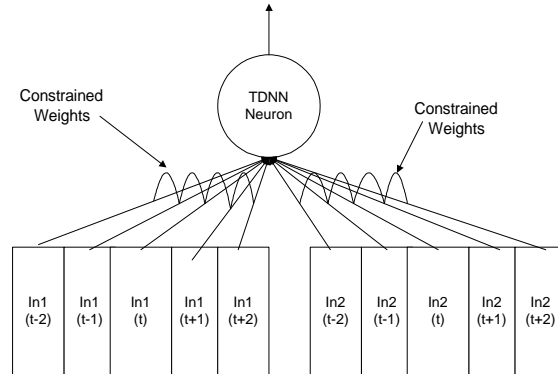


Fig 1. Time Delay Neural Network

The TDNN architecture chosen for the experiments in this paper is shown in Table 1.:

Number of Input Units:	27 (13 dcep, 13 ddcep, 1 power)
Number of Delays	5, on input layer only.
Number of Hidden Nodes:	96 (Chosen arbitrarily, to be pruned later)
Number of Output Nodes:	2 (Word Start Score, Word End Score)

Table 1. Network Architecture

The network outputs its predictions on the start and end of each word in the speech input. Various speech features like the fundamental frequency F_0 and other harmonics and zero-crossing could be used to train the TDNN, but we have chosen to use the first and second order cepstral differences (delta-cep and delta-delta-cep), largely because these are trivial and very efficient to compute once the cepstral vectors themselves are computed. The inputs are normalized to zero-mean and unit variance before being fed to the TDNN to improve convergence.

2.2 Network Pruning

Due to the difficulty in obtaining manually segmented data it is assumed that the amount of data available to train the initial TDNN is limited. There is a high risk of over-fitting, which will manifest itself in very good results on the known training data, but very poor results on unseen data. Choosing the correct network size to prevent this is time-consuming and tedious. Instead we have opted to prune the TDNN once it has been trained. The algorithm used in this paper is a simplification of Yann (1990).

Pruning Algorithm

For each hidden node H_j :

- i) Compute the mean activation \bar{h}_j value of H_j over all inputs I_i . H_{ji} is the activation value of hidden node H_j for input I_i , and n is the number of input patterns.

$$\mathbf{m}_j = \frac{\sum_{i=1}^n H_{ji}}{n}$$

- ii) Compute the significance of H_j . w_{jk} is the weight connecting hidden node N_j to output node O_k .

$$\mathbf{m}_{j(\max)} = \max_k \mathbf{m}_j w_{jk}$$

$\bar{h}_j w_{jk}$ can be understood to be the expected contribution of H_j to the output unit O_k , and hence $\bar{h}_{j(\max)}$ can be understood to be the largest contribution that node N_j is expected to make to the TDNN outputs.

- iii) Compute the variance σ_j of N_j for all inputs I_i .

$$\mathbf{s}_j = \frac{\sum_{i=1}^n (H_{ji} - \mathbf{m}_j)^2}{n-1}$$

This measures how node H_j changes over all the inputs I_i , and can thus be used to measure how active node H_j is.

- iv) Compute the pair-wise node distance \bar{e}_{ji} of node H_j and all other hidden nodes H_k , for all inputs I_i .

$$\mathbf{q}_{ji} = \frac{\sum_{i=1}^n |H_{ji} - H_{ki}|}{n}$$

A small \bar{e}_{ji} indicates that the activation for node H_j closely shadows that of node H_k .

- v) Prune the TDNN

A hidden node H_j is removed if either $\bar{h}_{j(\max)}$, σ_j or \bar{e}_{ji} fall below specific thresholds $\hat{\sigma}_j$, $\hat{\sigma}_\sigma$ and $\hat{\sigma}_e$. This corresponds to a node that is either contributing insignificantly to the TDNN output, or does not change significantly over all inputs, or shadows another hidden node closely. The thresholds are chosen empirically, and it is found that taken 105% of the minimum values of $\bar{h}_{j(\max)}$, σ_j and \bar{e}_{ji} gives the best compromise between network size and accuracy. Smaller threshold values do not reduce network size appreciably, if at all, and larger

threshold sizes impair the TDNN's ability to learn boundary information.

Using 105% of the minimum of $\hat{i}_{j(\max)}$, \hat{o}_j and \hat{e}_j results in 6 hidden nodes being eliminated.

This is a reduction of $27 \times 6 \times 2$ or 324 weights. The tied-weights in the input delays are considered to be single weights. If they are considered separately, then the reduction is $27 \times 5 \times 6 \times 2$ or 1,620 weights.

Improvements in recognition of outside data as a result of this pruning is discussed in the next section.

2.3 Prediction of Word Boundaries in Unseen Speech

Experiments were conducted to examine the feasibility of unsupervised training of the TDNN on new unseen speech data. A bootstrapping approach was used:

i) Generate Initial Estimate of Word Boundaries

The initial estimate can be generated in a number of ways. We take the naïve approach of taking the total number of frames in the utterance and dividing them equally between the expected number of words in the utterance.

Deriving the expected number of words is a much easier task than actually segmenting the speech, and is hence reasonable to use this in our segmentation.

20% of the frames allocated to each word is reserved to allow for spacing between words. This spacing is important for the relabeling program, which uses scores generated by the TDNN to move the boundaries.

ii) Re-train TDNN

The new speech utterance with its hypothetical boundaries is combined with existing utterances to re-train the TDNN.

iii) Re-align Boundaries

It is assumed that the actual boundaries lie somewhere close by, and a simple algorithm moves the boundaries to where the TDNN score is highest.

These three steps are repeated until insignificant changes are observed in the word boundaries.

3 Experiment Details and Results

3.1 Collection of Training Data

The initial TDNN is trained on speech samples collected from six subjects (3 male and 3 female) in a simulated train-scheduling task similar to the CISD Trains System described in Ferguson (1996). We had intentionally avoided ideal laboratory conditions by performing the recordings in a room with background chatter from other occupants, and even occasional noise from renovations above the room. This is to allow us to evaluate how the TDNNs perform under realistic, less than ideal conditions. An ordinary, non noise-cancelling microphone was used.

Since we have stated from the beginning that it is too expensive and tedious to generate large amounts of hand-labelled data, we have opted to restrict the number of input examples to just about 20 minutes of recording, or about 120,000 cepstral vectors.

3.2 Performance of initial TDNN

Table 2 compares the performance of the TDNN relative to hand-labeled samples, for both inside data used in training and outside, unseen data. Predictably the small data size has resulted in over-fitting, with inside Precision at 66.88% and outside Precision at 56.22%, or 10 percentage points difference. The F-Number, which presents Precision and Recall in a single metric, shows a less significant 4 percentage point difference.

Inside Test		Outside Test	
Precision:	66.88%	Precision:	56.22%
Recall:	67.33%	Recall:	76.69%
F-Number:	67.07%	F-Number:	64.88%

Table 2. Performance of TDNN Before Pruning

Table 3 shows the outcome after pruning the network. By eliminating 6 hidden nodes and re-training the neural network for a further 200 cycles, the difference in Precision, Recall and F-Number has decreased significantly, indicating that the model is no longer over-fitted.

Unfortunately the pruning has reduced the Precision and Recall of the boundary hypotheses, but this is to be expected when nodes and their corresponding weights are removed from a neural network.

Inside Test		Outside Test	
Precision:	66.03%	Precision:	57.10%
Recall:	61.41%	Recall:	72.16%
F-Number:	63.61%	F-Number:	61.71%

Table 3. Performance of TDNN After Pruning

The results shown above are promising, especially when it is remembered that the data used is noisy. The incorrectly inserted boundaries were often caused by background noise being mistaken for words, and the omitted boundaries often corresponded to utterances that were uttered quickly, or masked by background noise.

3.3 Word Boundary Prediction

Four previously unseen utterances were chosen to test the TDNN's ability to correct word boundary hypotheses. On each iteration the TDNN was re-trained for 100 cycles. The result of the first 5 iterations (excluding iteration 0) is shown in Table 4.

Iteration	Start Time Standard Deviation	End Time Standard Deviation
0	0.685	0.710
1	0.586	0.693
2	0.582	0.687
3	0.580	0.687
4	0.579	0.685
5	0.577	0.684

Table 4. Performance of Re-Labeling Method

The deviations shown in Table 4 are taken relative to word start and end times found by manual segmentation. The decreasing deviations show that the hypothesized boundaries converge towards the actual boundaries of the word. It is thus feasible to use TDNNs to automatically learn word

boundaries, though a more sophisticated way of finding the initial segmentations is required to allow the word boundary hypotheses to converge faster.

4 Discussion

Our experiments show that even though we had made several simplifying assumptions, promising results can still be obtained. More significantly we have shown that it is possible to use TDNNs to induce the actual word boundaries from hypothetical boundaries. Our method of generating initial boundary hypotheses is naïve but simple, and it is expected that better results may be obtained using more sophisticated estimation methods to derive the initial segmentation.

5 Conclusion and Future Work

This paper discusses some early research results in the use of TDNN to automatically learn word boundaries without supervision through producing an initial hypothesis on where word boundaries might exist, and then repeatedly refining that hypothesis to produce a better segmentation.

The results shown in section 3 are promising, especially considering the poor quality of the training data used. However this paper's main aim is to establish the feasibility of using TDNNs to induce word boundary information, and thus no attempt was made to compare the performance of the TDNNs to other word segmentation methods. This would certainly be an interesting extension to this research.

Future work will focus on combining the predictions of the TDNN with other methods like EM clustering, and in using either method to improve the performance of the other.

Preliminary work on phoneme-based word induction using multigram modelling show good results, and work is currently underway to integrate both the TDNN word boundary prediction approach presented in this paper and the multigram approach. Multigram modelling is introduced by Deligne and Bimbot (1997) and extended by Deligne and Yoshinori (2000).

References

- Toledano T, Crespo M.A.R and Sardina J.G.E., 1998. Trying to mimic human segmentation of speech using HMM and fuzzy logic correction rules. *Third ESCA/COCOSDA Workshop for Speech Synthesis, Jenolan Caves (Australia), Nov. 1998.*
- Ramana R.G.V. and Srichand J., 1996. Word Boundary Detection using Pitch Variations, *International Conference on Spoken Language Processing, Philadelphia, pp813-816, Oct 1996.*
- Laureys T, Demuyneck K, Duchateau J and Wambacq P, 2002. An Improved Algorithm for the Automatic Segmentation of Speech Corpora, *Proc. 3rd International Conference on Language Resources and Evaluation, volume V, pp 1564-1567, Las Palmas, Canary Islands, May 2002.*
- Waibel A, Hanazawa T., Hinton G., Shikano K., Lang K., 1989. *Phoneme Recognition using Time-Delay Neural Networks, IEEE. Transactions on Acoustic, Speech and Signal Processing, Vol 37 No. 3, pp 328-339, March 1989.*
- Yann L. C., Denker J. S., Solla S. A., 1990. *Optimal Brain Damage, AT&T Bell Laboratories, Holmdel, NJ 1990.*
- Ferguson G. M., Allen J. F., Miller B. W., Ringer E. K., 1996. *The Design and Implementation of the TRAINS-96 System: A Prototype Mixed-Initiative Planning Assistant, TN96-5, Computer Science Dept., U. Rochester, Oct. 1996.*
- Suh Y. J., Lee Y. J., *Phoneme Segmentation of Continuous Speech Using Multi-layer Perceptron, International Conference on Speech and Language Processing Vol 3.*
- Deligne S., Bimbot F., 1997. *Inference of Variable-Length Linguistic and Acoustic Units By Multigrams, Speech Communications Vol. 23 (1997) pp 223-241, 1997*
- Deligne S., Yoshinori S., 2000. *Statistical Language Modeling With a Class-Based n-Multigram Model, Computer Speech and Language Vol. 14, pp 261-279, 2000.*