Automatic Speech Recognition Using Recurrent Neural Networks

L. J .M. Rothkrantz and D. Nollen
Delft University of Technology, Knowledge Based Systems,
Zuidplantsoen 4, 2628 BZ Delft, The Netherlands
Phone +312787504 Fax +312787141
L.J.M.Rothkrantz@cs.tudelft.nl

Abstract

The main topic of this paper was to research the use of recurrent neural networks in the process of automatic speech processing. The starting point was a modified version of RECNET as developed by Robinson [4]. This phoneme recognizer is based on a RNN. But post-processing is based on Hidden Markov Models. A parallel version of RECNET was implemented on a parallel computer (nCUBE2) and Elman RNN was used as postprocessor. Word segmentation is also realized using an Elman RNN. The network models and results of testing are reported in this paper.

1 Introduction

Since the late 1970's speech technology has been dominated by the technique of Hidden Markov Models (HMM). In the late 1980's Neural Networks (NNs) became very popular. Recurrent NN are used to model temporal variation in speech [1, 2]. A better modeling of time variation was achieved by Time Delay NNs. But it turned out that the neural approach could not compete with HMM. From the early 1990's on hybrid speech recognition systems have been developed which combine the advantages of HMMs and NNs. A successful example is the RECNET system [3]. However today's speech recognition technology is still far from being competitive with human skills. A Markov process does not accurately model speech. A model of information processing in the human brain differs completely from HMM models and is more similar to NN models. Inspired by the working of the human brain, we developed an automatic speech recognizer, which only uses NNs. The preliminary results were reported in [5]

2 RECNET

The system RECNET is a recurrent neural network, developed at Cambridge University by T. Robinson [3, 4]. The network has two layers, an input and

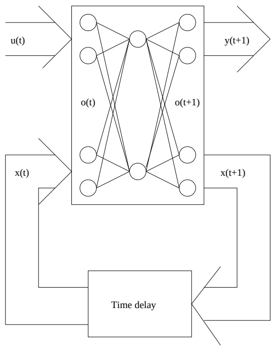

Figure 1: Architecture of RECNET

an output layer (see fig 1). Both the input and the output layer are split into an external part and a state part. The external part of the input vector is fed with the external information. The state input is an identical copy of the state output of the output vector one timeslot ago. The external part of the output vector is the pattern produced by the recognizer. The state part is fed back to the input, delayed one timeslot.

RECNET has been used as a phoneme recognizer. The acoustic speech signal is first split into frames. Then Hamming windows are used because they have less spectral leakage than block windows. The windows have a duration of 32 ms and shift 16 ms every new frame. From every frame a 23 valued feature vector has to be extracted. The components are the cube root powers in the first 20 channels derived from the Fast Fourier Transformation, the zero crossing rate, the fundamental frequency and the power of the speech signal.

The external input of RECNET is fed with the 23 channels from the preprocessor. The external output contains the probability distribution of the 61 phonemes available in the phoneme alphabet. The internal part of the network contains 192 nodes, both at the input layer and at the output layer. They form the context of the RNN. Every new frame, the complete input vector is formed by the output of the preprocessor and the state of the internal output layer one timeslot ago (see Fig. 1).

The system was trained, using a variant of the backpropagation through time algorithm (BTT)[3]. The training of RECNET is a computationally heavy task. The network contains a total of 56.026 weights. The network has to forward a frame every 16 ms, which makes more than 3.500.000 multiply and accumulate operations per second of processed speech. To speed up the process of training we implemented the RECNET system on a parallel computer (nCUBE2) with 32 nodes. The BTT algorithm has been made parallel by dividing the complete number of frames between two weight adjustments over the 32 nodes. Because the weights are fixed during the forward pass, the nodes don't have to communicate with each other during the local forward and backward pass. After calculating the average partial derivative, it is collected from all nodes by the

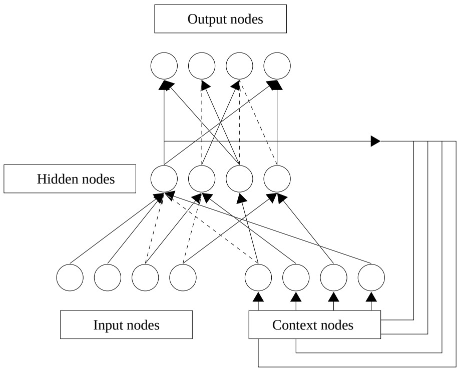

Figure 2: Architecture of the Elman RNN

master node. The master node averages the partial derivative of all nodes and adjusts the weights. After adjusting the weights a new cycle has been started.

By dividing the frames over several nodes, which operate independently, we introduce a small error. In a single processor environment the state vector of the RNN will contain some kind of context information, which is dependent of the previous frames. This means if we want to divide the frames into smaller pieces, we also need to know the initial values of the state vector. The difficulty is that this value isn't known until all previous frames have been forwarded but then we don't speed up the process. By choosing the number of frames big enough, this effect is not significant. We used a number of 2048 frames between two weight adjustments. This means that every node has to calculate the partial derivatives of 64 frames. This number of frames is big enough to make the introduced error insignificantly small.

The system is trained and tested with the TIMIT database. This set contains 5966 utterances in the training set. After preprocessing the complete training set contains 700.130 frames. The preprocessed frames are labeled with the correct phonemes. Per training cycle over 78 billion floating-point calculations have to be realized. The training time on a SUN 14 MHz was 700 hours and on the nCUBE2 70 hours. After approximately 70 training cycles with decreasing stepsize we found that 64.7% of the frames were labeled correctly. Similar results were reported by Robinson [3].

3 Phoneme post processing

The output of the RECNET phoneme recognizer is a vector with the probability distribution of phonemes of every speech sample. One or more probability vectors represent a phoneme, depending of the duration of a phoneme. Successive samples show a lot of variation due to errors in speech recognition and variation in speech. To convert this stream of probability vectors into a stream of single phonemes without repetition different post-processing techniques have

Table 1: Scoring without post-processing

Number of phonemes	51,681	
Correctly labeled	41,290	79.9 %
Insertion errors	23,839	46.1 %
Substitution errors	9,072	17.6 %
Deletion errors	1,319	2.6 %
Total number of errors	**34,230**	**66.2 %**

Table 2: Scoring of the HMM model

Number of phonemes	51,681	
Correctly labeled	37,644	72.8 %
Insertion errors	1,097	3.7 %
Substitution errors	10,974	21.2 %
Deletion errors	3,063	5.9 %
Total number of errors	**15,934**	**30.8 %**

been used. To test the performance of these techniques the complete test set of the TIMIT database was used. This set contains 321.680 frames with 51.681 phonemes.

After applying a postprocessor, a stream of phonemes or words is obtained. To score the recognizer one has to compare the desired outcome with the real outcome. There are three different kinds of errors a recognizer can make:

- Substitution, a phoneme is recognized as a different phoneme
- Deletion, a phoneme is not recognized at all
- Insertion, a phoneme is recognized which wasn't spoken

To match the reference sentence to the hypothesis sentence the errors can be determined. This matching is known as symbol string alignment. The technique used for determining the best alignment is based on Dynamic Time Warping.

First we scored the output of the RECNET recognizer directly; without any post-processing. The repetitions in the phoneme stream are deleted and this stream of single phonemes is optimally aligned with the stream of correct phonemes. Next the numbers of errors were computed (Table 1). From the table we can see that the number of insertion errors is very high. Every error made by the RECNET recognizer disturbed the repetition of phoneme labels during a phoneme and will insert one or two incorrect phonemes.

In [3] the post-processing is realized by using Hidden Markov Models. The total error rate decreases significantly (Table 2). As expected the improvement is made on the insertion errors. The small irregularities are smoothened out by the HMM.

4 RNN phoneme postprocessor

A different approach in correcting the irregularities in the output of RECNET recognizer is the use of a second Recurrent Neural Network. The idea is that the network is trained to perform a similar function as the HMM. The architecture used for the Phoneme Postprocessor RNN is the Elman architecture as displayed in Fig. 2. The input of the network contains 61 nodes, corresponding to the probabilities of all phonemes. The output of the network also contains 61 nodes, corresponding to the probabilities of the 61 phonemes after postprocessing the phonemes. A context layer of 150 nodes has been used. The goal of the Elman RNN is to transform its input (the output of RECNET) in the correct

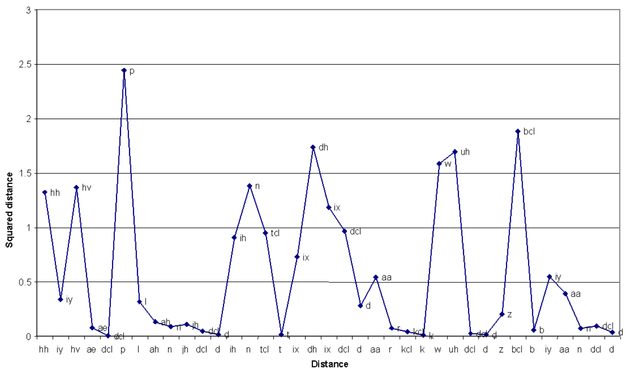

Figure 3: Output of the phoneme predicting RNN

phoneme, in which the correct phoneme has the highest probability. The recurrent connection and the context layer enable the network to include information of previous frames.

In order to train the Elman RNN we had to create a training set. As noted the output of RECNET contains many errors. So if we take the phoneme with the highest probability as the assumed correct output, we make a lot of errors. We can distinguish two categories of classifications in the output of RECNET. The first category contains a probability distribution, in which the correct phoneme shows the highest probability and the second category in which an incorrect phoneme shows the highest probability. By definition we assume that the correct output vectors of the RECNET phoneme recognizer contain a one for the correct phoneme and zero for all others.

In order to correct classification of category two, we used some characteristics of speech. Almost all phonemes last more than one frame (16 ms). This repetitive character, embodied in the context layer, can correct category two classifications. In case of a HMM postprocessor the repetitive character is included in the fact that the transition of one phoneme to itself contains the highest probability for every phoneme (=state). This prevents the output of the phoneme processor to bounce from phoneme to phoneme very new frame.

In a first approach we simply feed the output of the RECNET phoneme recognizer to the network, and the correct output is fed backward (correct phoneme probability 1 and all other probabilities 0). The probabilities provided by RECNET are normalized. After training the PPRNN reduced the error rate to 60.8%. This was a small decrease of errors if we compare the performance to the output of RECNET without any phoneme postprocessing. The errors corrected by the PPRNN were mostly insertion errors. It proved that the network was fed during the training phase with probability distributions in which 65% of the frames was labeled incorrectly. This means that the network wasn't trained to include the repetitive character of the phoneme labels.

In order to improve the performance, the training set needs to be adjusted.

Table 3: Scoring of the Phoneme Post-processing RNN

Number of phonemes	51,681	
Correctly labeled	38,812	75.1 %
Insertion errors	11,318	21.9 %
Substitution errors	10,905	21.1 %
Deletion errors	1,964	3.8 %
Total number of errors	**24,187**	**46.8 %**

The problem in the first experiment was that the network has never been trained on the correct data, which means it cannot use this information in the phoneme postprocessing process. On the other hand if we simply feed the perfect output of RECNET into the PPRNN (correct phoneme probability 1 and all other probabilities 0) and use the same vector as the desired output, the network will train itself to exclude the context. The correct output is already available in the input so the context is not needed.

We designed our training set as a linear combination of the "RECNET output" and the "correct output".

$$\text{Mixed output} = \alpha \text{ Normalized RECNET output} + (1-\alpha) \text{ Correct\_output} \quad (1)$$

We got the following results:

- If we feed the output of the RECNET phoneme recognizer to the network and the correct output is fed back ($\alpha = 0$) the performance was bad. The error rate was reduced to 60.8%. The input has too many errors (35%). So the network wasn't trained to include the repetitive character of the phoneme labels.
- If we simply feedback the correct output of RECNET and use the same vector as the desired output, the network will train itself to exclude the context.
- If we feedback a combination of the "RECNET output" and the "correct output", we found better results. In Table 3 we display the results ($\alpha = 0.5$).

Although the performance of the RNN phoneme postprocessor is significantly increased by the mixing methods, a lot of improvements have to be made to compete with the HMM postprocessor. We can explain the big difference in performance between both postprocessors as follows:

The output of the RECNET recognizer contains still a lot of irregularities. The correct phoneme shows a high probability most of the time, but other phonemes can also show a high probability, sometimes even higher than the probability of the correct phoneme. The main difference between the HMM and the RNN phoneme postprocessor is that the HMM determines the most likely phoneme by calculating the most likely complete path of the desired length. This means that although the transitions with a HMM are only dependent of the preceding phoneme, the most likely path is computed on the basis of the complete phoneme stream. The RNN postprocessor on the other hand has to correct the irregularities after receiving the second frame, not knowing what the most likely phoneme of the next phoneme is. This lack of information about the future is the main disadvantage of the RNN postprocessor.

5 Phoneme prediction

The outcome of the Phoneme Postprocessing RNN of the last section is a continuous stream of phonemes. A next step in automatic speech processing is the segmentation of this stream of phonemes. This means the beginning and ending of separate words need to be determined. The segmented stream of phonemes can be converted into a stream of words or wordgraph. In [1, 2], J. Elman used RNN in a study on character prediction. He used a RNN as displayed in Fig. 2. We made a similar experiment for phoneme prediction. The idea is to feed a RNN with a sequence of phonemes one by one. The output of the network predicts its next input, the next phoneme. The context layer provides the network with some kind of memory. Using the context generated by the previous phonemes, the network will try to predict the next phoneme. Once we have a trained RNN we can use this RNN again to detect and correct errors in the stream of phonemes. The network we used in our experiment has an input and an output layer of 6 nodes. After some experimentation we found out that 100 nodes in the hidden layer is a good choice. The input and the output of the network are the current phoneme and the next phoneme in a coded format. The 63 possible phonemes are coded in a 6 bit binary code.

To create a vocabulary 10 sentences are chosen randomly from the TIMIT database. The 10 sentences form a vocabulary of 105 words. The TIMIT database contains for every sentence the correct phonetic transcription. Because we want a segmentation corresponding to words, the network has to be trained on prediction within a word, not between two words. To avoid these between-word relationships, the word sequences are chosen randomly. The training set is created by sampling 2000 words randomly from the vocabulary of 105 words. In this way we had a continuous stream of 9361 phonemes. The squared distance between the desired phoneme code and the network output was used as the error function.

After 60 training cycles the average squared distance limits to 0.3. After training the network was tested on the 10 sentences. One of the test sentences with phonetic transcription was: "He had plunged into the dark woods beyond."

```
(hh iy hv ae dcl p l ah n jh dcl d ih n tcl t ix dh ix
dcl d aa r kcl k w uh dcl d z bcl b iy aa n dcl d).
```

The phonemes can be converted to the correct English words. The previous sentence should be parsed into the following sentence:

```
he        hh iy
had       hv ae dcl
plunged   p l ah n jh dcl d
Into      ih n tcl t ix
the       dh ix
dark      dcl d aa r kcl k
woods     w uh dcl d z
beyond    bcl b iy aa n dcl d
```

From Fig. 3. it can be seen that most of the initial phonemes show a local maximum in the error. Not every maximum is the beginning of a new word, and not every first phoneme is a local maximum. In case of words with equal parts we can have local maxima not related to the beginning of new words. But especially within longer words, the squared distance decreases for every next phoneme. It proves that the increase of the squared distance (threshold > 0.5), in combination with the value of the squared distance (threshold > 1.0) determined whether a local maximum is an indicator of a new word. So we can use this phoneme prediction network to segment a stream of phonemes. It proved that in 80% of the situations the parser indicates the right segmentation point.

6 Conclusions

The main goal of our research was to design and implement an Automatic Speech Recognizer based on NN. A prototype is successfully implemented. We implemented a parallel version of the RECNET phoneme recognizer. It proved that 65% of the phonemes were correctly classified. A RNN phoneme postprocessor was implemented to smooth the RECNET output. The error rate was reduced with 20%. A HMM postprocessor reduced the error with 36%. So our RNN is still not optimal. But an advantage is that it performs real time. We used RNN to parse the stream of phonemes. On a vocabulary of 105 words 80% of a phoneme stream was segmented correctly.

References

[1] Elman, J.L.: Finding structure in time, Cognitive Science, 14, (1990), 179-211

[2] Elman, J.L., E.A. Bates, M.H. Jonhson (eds.): Rethinking Innateness, The MIT Press, Cambridge, Massachussets, (1996)

[3] Robinson, A. J.,F. Fallside: A dynamic connectionist model for phoneme recognition, Proceedings of nEuro '88, pages 541-550, I.D.S.E.T., Paris, (1989)

[4] Robinson, A. J.,F. Fallside: A recurrent error propagation network speech recognition system, Computer speech and language, Volume 5, Number 3, (1991)

[5] Rothkrantz, L. J. M., E. Nollen: Speech Recognition using Elman Neural Networks, Proceedings of the Second International Workshop TSD'99, Plzen, (1999)

[6] Tebelski, J.: Speech recognition using Neural Networks, PhD Thesis, Carnegie Mellon University, Pittsburgh, USA, (1995)