A RECURRENT NETWORK THAT LEARNS TO PRONOUNCE ENGLISH TEXT

M.J. Adamson and R.I. Damper

Image, Speech and Intelligent Systems (ISIS) Research Group,
Department of Electronics and Computer Science,
University of Southampton,
Southampton SO17 1BJ, UK.

Email: {mja95r|rid}@uk.ac.soton.ecs

ABSTRACT

Previous attempts to derive connectionist models for text-to-phoneme conversion – such as NETtalk and NETspeak – have generally used pre-aligned training data and purely feedforward networks, both of which represent simplifications of the problem. In this work, we explore the potential of recurrent networks to perform the conversion task when trained on non-aligned data. Initially, our use of a single recurrent network produced disappointing results. This led to the definition of a two-phase model in which the hidden-unit representation of an auto-associative network was fed forward to a recurrent network. Although this model currently does not perform as well as NETspeak, it is solving a harder problem. Also, we propose several possible avenues for improvement.

1. INTRODUCTION

NETtalk [1] is a well-known multi-layer perceptron (MLP) trained on error-back propagation to convert text to phonemes. Subsequently McCulloch *et al* [2] produced a network called NETspeak. Intended as a re-implementation, it extended NETtalk in several ways. However, both NETtalk and NETspeak have been criticized on two grounds.

First, the task of text-to-phoneme conversion is one which requires processing of sequences over time. It has been argued by authors such as Damper [3] that a recurrent (temporal) network provides a more appropriate solution to such problems. Second, the data used to train these networks was pre-processed such that the text and corresponding phonemes were aligned. In this way, the difficult task of automatically aligning words to their phonemic equivalents was avoided [3,4].

In this paper, following the ideas of Jordan [5,6] Elman [7,8], and Pineda [9], we address these criticisms through use of a recurrent network. The alignment problem is tackled via a two-phase model which splits the text-to-phoneme task into two broad procedures – that of learning words, then associating a sequence of phonemes with each word – which themselves are trained in a gradual manner.

Currently, the network performance is poorer than NETspeak. However, several improvements are suggested which we believe have the potential to increase accuracy.

2. BACKGROUND

The ability of neural networks to learn complex non-linear mappings between data makes them attractive for applications such as text-to-phoneme conversion. In particular, back-propagation learning has led to several implementations including NETtalk [1] and NETspeak [2]. These achieved a reasonable level of performance at around 85% phonemes correct on both seen and unseen data, a figure which is somewhat below that achieved by the best data-driven systems. One possibility would be simply to develop the basic NETspeak model so as to gain incremental improvements on this figure. Instead, however, we prefer to focus on the two fundamental deficiencies detailed in the previous section – namely the non-temporal nature of NETspeak and the requirement for pre-aligned training data. (In the remainder of this paper, when referring to "NETspeak" we also implicitly include NETtalk.)

NETspeak can be described as non-temporal in that its input/output mapping is "fixed". That is, a seven-character window provides input, where the central letter in this sequence is associated with the current output pattern. This gives rise to an unrealistic one-to-one mapping of character to phoneme (where surrounding letters are used for context). Furthermore, it leads to the necessity for padded pre-aligned data, a further criticism. This pre-alignment of letters with phonemes can be argued to be inappropriate on two grounds. First, part of the learning is achieved *a priori*: the difficult task of learning which groups of letters align to which phonemes is avoided. Second, the use of "null" and "extra" phonemes for padding (to make letter and phoneme strings the same length) is an obvious patch, having nothing to do with any reasonable description of English orthography and sound structure.

In our opinion, reading text is a task which involves acquiring data (letters) in a sequential manner, where time is significant. The important issue is that the representation of time should be inherent within the model. Hence, a recurrent network, as proposed by Jordan [5,6] and Elman [7,8], may provide a more appropriate solution. Rather than NETspeak's problematic one-

to-one mapping requiring pre-aligned training data, we believe that the network should develop a holistic concept of "words". We attempt to do this using an auto-associative network as the first phase in a two-phase model. This auto-associator constructs internal representations of the words in its training data-set, which are passed on to the second phase – a recurrent network – for translation.

A further criticism of NETspeak is that it attempts to learn the entire task "in one go", rather than taking a gradual, or incremental, approach. It can be argued (e.g. [10]) that infants acquire a grasp of "words" long before they are capable of sophisticated utterances (i.e. simple speech). This can be seen by an infant's appropriate reaction to different verbal (and visual) stimuli. The route towards mastering basic language skills (such as the full phoneme inventory) is gradual. This is obvious through observing an infant, at first babbling before progressing onto one- and two-word utterances. The use of an auto-associator as the first part of a two-phase model allows us very naturally to teach the network in a gradual fashion (although this has not been fully exploited).

In short, NETspeak is considered to over-simplify the text-to-phoneme problem, which contributes towards its 85% level of accuracy. However, this simplification reduces the model's biological and psychological plausibility, and may detract from its ultimate achievable performance level. Hence, this defines our motivation for the model which follows.

3. RECURRENT IMPLEMENTATION: PRELIMINARY WORK

Before the two-phase model was developed, several attempts were made to implement a recurrent network which could map letters to phonemes in a single stage. In this section, we outline this work and its shortcomings – which led us to the use of an auto-associator plus recurrent net. Specific details of input/output codings and the training data, which were the same for both the single-phase and two-phase models, are given when we deal with the latter in section 5.

The single-stage net is based on the work of Jordan [5,6] and Elman [7]. First, a recurrent net was constructed by copying the output activations back to the input, in the form of a "state layer". Training at this stage was on pre-aligned data. All feedback connections from output to input had unit weight so that error back-propagation learning could be used. The three-letter left context of NETspeak's input window was discarded (but the three-letter right context retained) in the expectation that the state layer would retain this information. The resulting drop in performance relative to NETspeak was from about 85% "best guess" phonemes correct to approximately 80.5% on both seen and unseen data. However, NETspeak itself with this reduced input window did not perform very much worse at about 78%, showing that the additional state layer and feedback actually achieve very little. Worse still, if the right context was removed also, leaving just a one-character window, performance dropped disastrously to just over 16%.

The above attempts were repeated with feedback to the state layer taken from the hidden units. Contrary to our intuition (we favour feeding back the hidden-unit representation), this led to only a marginal improvement from 16% to 20%.

We believe the goal of associating non-aligned data is probably unattainable in this manner. The network effectively tries to learn incompatible and inconsistent 1-to-*n* input/output mappings and unavoidably fails. It was clear that simply extending the model was inappropriate, and a different implementation was required.

4. TWO-PHASE MODEL ARCHITECTURE

We have now defined the need for a two-phase model (see Figure 1). The task of the first phase is to acquire the concept of "words" derived from the hidden unit representation. This is to be achieved through a simple auto-associative network, i.e. trained to reproduce its input as its output. After experimentation, the configuration was based on an 18-character input window, capable of supporting the longest word. Input words were left-justified in the window and the space to the right padded with nulls. To avoid these null patterns dominating training, the network only updates the weights where there are letters. In this way, the network is presented words letter by letter gradually, and learns to associate specific weights in accordance with word size as well as specific letter positions.

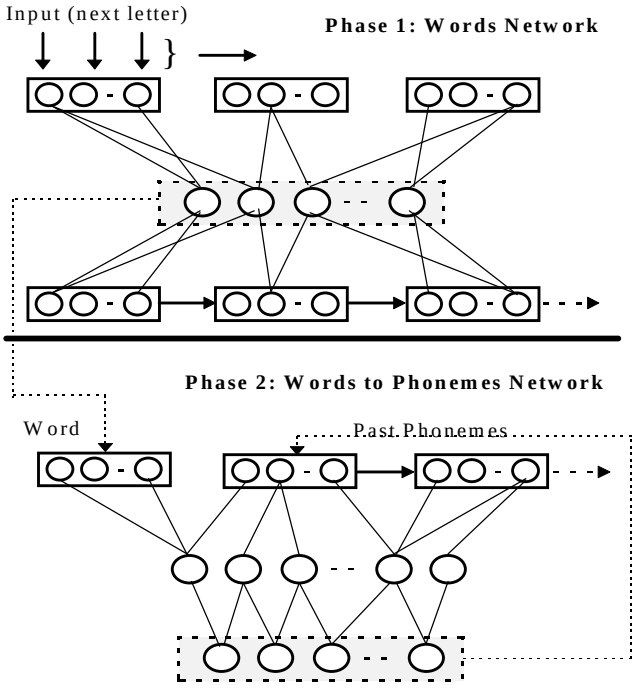

Figure 1: Two-phase text-to-phoneme model.

However, because of the left-justification of the window, the same substring appearing at different places in different words is not treated the same. Despite this "invariance" problem, the 18-

character window is still favoured at this time over a single-character window (plus state units) as described in the previous section. We observed through various tests that the state "memory" provided by the latter model was limited (usefully) to only a few letters. The result was that the network only regarded words as similar if they ended the same. Using the 18-character window, this problem is avoided since the current input is retained directly for subsequent processing. However, we believe this is unsatisfactory in the long term and may not be the most appropriate system. Hidden-layer feedback, as proposed by Elman [7], allows for use of the network's own (developing) coding system during learning. By extending the ideas of Elman, it may be possible to provide a long-term memory, and also tackle the invariance problem at the same time.

The second phase associates a sequence of phonemes with the learnt words based on the extracted hidden layer representation for each of the words from the stable phase 1 auto-associator. The second-phase model is a simple recurrent network. The input this time is split into two parts, one static for the duration of each word (containing the current word code derived from the auto-associator), and the other a scrolling window where phonemes previously processed are passed back. The task is therefore one of learning valid phonemic sequences. For each iteration, the network approaches the problem by processing: "for the current word, given what phonemes have passed before, the next one in the sequence is...".

The concept of starting small when training neural networks is discussed by Elman [8], who points out how sometimes tasks may only be learnt incrementally. However, Elman also shows that the opposite is sometimes true. Choosing the most appropriate training method is down to trial and error. Such ideas are restricted in our model at this time to a two-phase process in which weights are only updated for the input/output patterns that contain values. Although this was primarily to solve the padding problem, it also serves as a way for explicitly making the learning process gradual.

5. TRAINING THE NETWORK

The model as described above is trained using the standard error back-propagation method. Only feed-forward connections are updated (feedback connections have fixed, unit weight values), and where data is passed back, it is done so unaltered. What follows is a brief description of specific training issues.

5.1. Input/Output Coding

As a starting point, the coding scheme adopted by NETspeak [2] was re-created by hand. However, because we lacked documentation detailing the stress information, it was excluded at this time. This was not seen as an over-simplification of the problem, as stress was used by NETspeak purely for synthesis purposes, and its omission should not affect the overall network performance. This was confirmed when we re-implemented NETspeak with our modified output patterns, and achieved very similar results.

For specific information of the NETspeak input/output pattern coding scheme, refer to McCulloch *et al* [2].

5.2. Training Data Set

The dictionary used for training and testing NETspeak was also adopted for our model. It comprised 16,280 words from *The Teacher's Word Book* [11]. Of this, the first 15,080 were used for training, where 1,200 words from the learnt data set and the remaining 1,200 words never seen during training were used to test the network's performance.

5.3. Network Size

Through a process of trial and error, the following network sizes (number of units) were derived as suitable.

Phase 1 :	**Phase 2 :**
Input Layer = 18×11	Input Layer = 18×19+45
Hidden Layer = 45	Hidden Layer = 175
Output Layer = 18×11	Output Layer = 19

In both phases, the learning rate was 0.05, the momentum was 0.95, and errors only back-propagated when greater than 0.01.

5.4. Training Procedure

As previously discussed, network learning (and recall) is a two-phase process. To start, the first (auto-associative) network establishes the notion of words. When the network has achieved acceptable levels of performance at this task, the hidden-layer representation for each word is extracted. This is then used as a contribution to the input of the second (recurrent) network during training.

In phase 2, the word is first presented in isolation, with the task of the network being to derive the first phoneme for this word. For the next iteration, the phoneme is then passed back to join the previous input (in this instance, just the word) in order that the second phoneme can be derived. This process continues until all phonemes for the current word are exhausted. The task therefore is one of learning valid phonemic sequences.

6. RESULTS

Several flavours of the model described above were implemented. However, we report here the best results from our findings to date.

It can be seen from the table of results below that allowing the network more knowledge of past phonemes (memory) yields better results. The performance was observed to level out at around 70% best guess. This result was unexpectedly low as it was anticipated that by using a smaller scrolled memory, the network would be better placed to learn regularly-occurring (short) phonemic sequences. The explanation for this is believed to lie in training data biasing as discussed below.

Although the model presented in this paper tackles essentially the same problem as NETspeak, the way the training data is used is

completely different. Consequently, although the NETspeak dictionary could be considered adequate for its associated implementation, the same is not true for ours.

Epochs	Memory	%Bits	%Guess	%Exact
10	4	82.58	59.23	23.84
10	7	83.04	60.89	24.92
10	18	84.55	62.29	27.88
25	18	86.01	66.79	31.87

Performance on previously seen data

10	4	82.65	59.29	23.94
10	7	82.99	60.39	24.23
10	18	84.55	61.87	27.71
25	18	86.22	66.85	31.93

Performance on unseen data

Epochs : No. of times training data presented to the network.
Memory : Passed-back phonemes memory size. When the maximum is reached, the oldest phoneme is disregarded.
%bits : Percentage of individual output values which were within 0.1 of their target values.
%guess : Percentage of phonemes which were closest (in Euclidean distance) to the target phoneme.
%exact : Percentage of output vectors where every output value was within 0.1 of the correct value.

Table 1: Final network performance on seen and unseen data

When the training data was examined, it was seen to contain many occurrences of common phonemic sequences. Hence, during learning, the network could be biased to the most common because there were insufficient occurrences of other possibilities for the network to generalize to an alternative. In fact, we feel the results as a whole will be greatly improved when a more appropriate training dictionary is used – one where the network is shown valid phonemic combinations in more equal proportions.

7. CONCLUSIONS

It can be seen from the results achieved at present that the model's performance falls some way behind NETspeak. The justification for using our model currently is its avoidance of pre-aligned training data. It could also be argued as showing far more potential (eventually) for reaching adequate levels of performance.

As the task attempted by our model is consequently much harder, its performance is arguably quite good. When results are compared with our NETspeak re-implementation trained without pre-aligned data – which only achieved around 20% – our model's performance at this more difficult task seems less pessimistic. It is believed, despite early results, that a model which considers sequences and is taught incrementally is the appropriate route to follow. With further improvements to exploit the model presented here, the current level of performance should improve.

Moves are being undertaken to produce a training data-set which does not suffer from biasing. We also intend to improve the auto-associative "word" model such that invariance is properly considered. In this way, the phase 2 (recurrent) network can draw more on the subtle similarities of words irrespective of where they occur.

Other areas of the model to be addressed (at a later stage) will include the internal organization of the data by the network. Our motivation for this lies with humans who are believed to rely on spatial organization and classification of data in order to process language.

8. REFERENCES

1. Sejnowski T., and Rosenberg C. (1987), "Parallel networks that learn to pronounce English text", *Complex Systems*, Vol. 1, No. 1, 145-168.
2. McCulloch N., Bedworth M, and Bridle J. (1987), "NETspeak – A re-implementation of NETtalk", *Computer Speech and Language*, Vol. 2, 284-301.
3. Damper R.I. (1995), "Self-learning and connectionist approaches to text-to-phoneme conversion", in *Connectionist Models of Memory and Language*, Levy J., Bairaktaris J., Bullinaria J., and Cairns P. (eds.), UCL Press, London, 117-144.
4. Bullinaria J. (1995), "Neural network models of reading: solving the alignment problem without Wickelfeatures", in *Connectionist Models of Memory and Language*, Levy J., Bairaktaris J., Bullinaria J., and Cairns P. (eds.), UCL Press, London, 161-178.
5. Jordan M.I. (1986), "Attractor dynamics and parallelism in a connectionist sequential machine", *Proceedings of Eighth Conference of the Cognitive Science Society*, Amherst, MA, 531-546.
6. Jordan M.I. (1988), "Supervised learning and systems with excess degrees of freedom", *Proceedings of the 1988 Connectionist Models Summer School.*
7. Elman J.L. (1990), "Finding structure in time", *Cognitive Science*, Vol. 14, No. 2, 179-211.
8. Elman J.L. (1993), "Learning and development in neural networks: the importance of starting small", *Cognition*, Vol. 48, 71-99.
9. Pineda F. (1995), "Recurrent back-propagation networks", in *Backpropagation: Theory, Architectures and Applications*, Chauvin Y. and Rumelhart D.E. (eds.), Lawrence Erlbaum, Hillsdale, NJ, 99-135.
10. Slobin D.I. (1979), *Psycholinguistics (Second Edition)*, Scott, Foresman and Company, Glenview, IL.
11. Thorndike E., and Lorge I. (1944), *The Teacher's Word Book of 30 000 Words*, Teachers' College, Columbia University, NY.