

Learning Word Pronunciations Using a Recurrent Neural Network

Matthew J. Radio¹, James A. Reggia^{1,2}, and Rita S. Berndt²

radio@wam.umd.edu, reggia@cs.umd.edu, rberndt@umaryland.edu

¹Department of Computer Science, University of Maryland, College Park, MD 20742

²Department of Neurology, University of Maryland School of Medicine, Baltimore, MD 21201

Abstract

Segmentation is the process of dividing a printed character string into graphemes, each of which is associated with one (or rarely more) output phonemes. The purpose of this study was to investigate what internal representation of the segmentation process and character-to-phoneme correspondences would be learned by a recurrent neural network as it was trained to produce the correct temporal sequence of phonemes for printed words held fixed on its input nodes. The resilient recurrent backpropagation network learned very effectively to generate the correct pronunciation for 150 words. Some interesting rules of pronunciation discovered by the network were extracted despite the network's distributed representation.

1 Introduction

The acquisition of literacy depends on learning to associate strings of printed characters with their pronunciations. In English this task is complicated by the facts that (1) many individual characters have more than one possible pronunciation, and (2) characters may be combined before being associated with a single sound (e.g., PH→/f/). The second of these factors suggests that learning to read requires segmentation of strings into pronounceable units. Since characters and phonemes are not in a one-to-one correspondence in many words, it is not obvious that pronunciation information constrains character-string segmentation. Few characters are always associated with a single sound, and many characters are pronounced in several ways [1]. Thus, there is no unambiguous map from the set of words of the English language to the set of phoneme sequences.

The purpose of this study was to investigate the internal representation of the segmentation process and grapheme-to-phoneme correspondences that would be

learned/discovered by a recurrent neural network as it was trained to produce the correct temporal sequence of phonemes to pronounce printed words. This internal representation gives clues as to how the network distinguishes particular word features and creates rules to segment and pronounce words correctly. The internal representation can also give clues as to whether or not the network simply memorizes whole input patterns to associate with temporal phoneme sequences. These opposing methods form the basis for the dual-route print-to-sound model of word pronunciation [4].

A second goal of this study was to critically evaluate the effectiveness of some specific formulations of recurrent backpropagation (RBP) for this task. To our knowledge, this is the first use of a recurrent neural network of this kind to generate correct pronunciations of written words. Unlike some past systems, such as NetTalk [6], there is no sliding window used to designate which of the current characters to pronounce, i.e., the network must learn to segment the characters as well as to pronounce them. In other words, the network must learn the difficult task of generating the correct sequence of output phonemes from a *fixed* representation of the input word, a very challenging task. We evaluated how basic backpropagation [3], backpropagation with momentum, backpropagation with an adaptive learning rate [2], and resilient backpropagation [5] all performed on this problem.

2 Methods

2.1 The Model

Error backpropagation learning rules designed for training recurrent neural networks were used. The model is trained to accept a word as input and to produce the correct temporal sequence of phonemes for the pronunciation of that word as output. The model is also trained to produce an “end of word” signal when it is finished pronouncing the

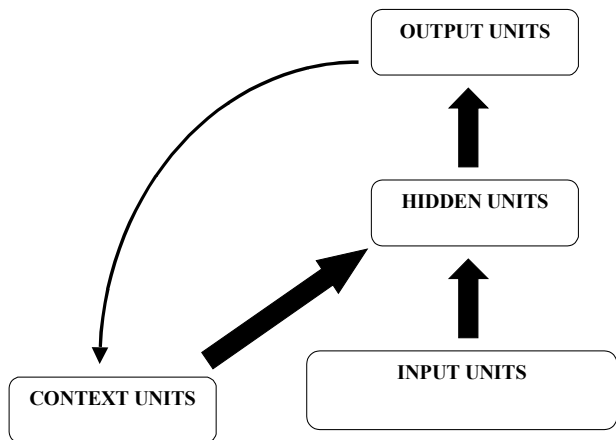


Figure 1: Basic Network Architecture.

word. The data for this study were taken from the NetTalk corpus [6]. A set of 150 randomly chosen words was selected from this corpus, restricted to words of length six letters or less. Words of length 2, 3, 4, 5, and 6 were represented equally (30 of each). The spelling and correct output phoneme sequence for each word was available to the model during training. For example, for the word *coffee*, the six letters C O F F E E served as a fixed input pattern and the correct/target output temporal sequence was /k/, /aw/, /f/, and /ee/, followed by the end of word signal.

The architecture of the model is summarized in Figure 1. The input layer consists of 157 nodes. The first node is a bias unit. The remaining 156 are divided into six groups of 26 nodes. Each group corresponds to the input character position in a word, and the nodes in each group correspond to the 26 letters in the alphabet. Each input node is activated at a level of 1.0 if the node is “on” or 0.0 if the node is “off.” The hidden layer consists of a number of nodes (this number was varied in different simulations), which are fully connected to the input layer, output layer, and context unit layers (state units) as illustrated. The output layer consists of 45 nodes representing the 44 phonemes present in the NetTalk corpus and one “end of word” (EOW) node. A context layer also consists of 45 nodes, which give feedback to the hidden units via recurrent connections from the output, similar to those used in [3]. The activation pattern over the context units represents the previous state of the network. The nodes corresponding to the most recently pronounced phonemes are given the most activation.

During simulations, the activation pattern representing each written word is clamped to the input layer and held fixed while the network generates a temporal sequence of phonemes as output. Initially, activation propagates

through the hidden layer of nodes and produces an output. This output pattern then activates the context layers to give input to the hidden layers to be used to generate the next output. The network recomputes its output, this time generating the second phoneme in the word using activation from both the input units and context units. This is repeated until the EOW signal is produced or for a specified maximum number of iterations occurs. The logistic transfer function is used at both the hidden and output layers.

This network is trained using various error backpropagation algorithms. In this study, several variants of backpropagation learning were adapted for use to attempt to determine the best method. The methods implemented include basic backpropagation [3], basic backpropagation with momentum, backpropagation with an adaptive learning rate algorithm [2], and resilient backpropagation or RPROP [5]. A penalty term increasing the error when a node’s activation is less than its target activation was used, as this was found empirically to improve learning effectiveness.

2.2 Performance Measures

Each simulation’s performance was evaluated using two independent measures: root mean squared error (RMSE), and the percentage of phonemes that was generated completely correctly (PCT). Completely correct output is defined for this study as having the single correct phoneme activated at greater than or equal to 0.9, and all other output phonemes activated at less than or equal to 0.1. Training consists of repeated passes through the data in either an incremental or batch learning mode, specified prior to each simulation. Training continues until either the RMSE reaches a certain level (0.01), the RMSE does not increase nor decrease for 1000 training epochs, or after 10,000 passes through the data. After training, the hidden unit activation values and the weight vectors associated with the network were examined to determine any internal representations learned by the network.

3 Results

The best-trained network was a simulation using 45 hidden units in the network, the RPROP learning algorithm, and a low output penalty term of 40. The latter means that the scaled error was multiplied by 40 when an inactive output node was supposed to be activated. Two context layers were used to represent the previous two states of the network. This network ran for 1500 training epochs, with the performance tested every 5 epochs. The best set of weights was saved throughout training to be used to examine the network’s internal representation. This network achieved a RMSE of 0.01 and a PCT of 99.55% after only 160 epochs. The results below are based on this network.

3.1 Generalization

Generalization is one of the key components of any neural network. The idea that a network simply memorizes the correspondence between its explicit input and output seriously limits a network's usefulness in new problems. To examine this issue, a test data set of 150 words was developed. This set had a three-word overlap with the training set, simply due to random selections from the NetTalk corpus. The network performed very poorly on this set. The performance was a mere 14% PCT for correct phoneme production. Only five words were produced entirely correct by the network and three of them were the words from the training set. This low score is due to the fact that when one phoneme is pronounced incorrectly, the network receives incorrect feedback from its context units and therefore tends to produce incorrect phonemes for the rest of the word.

Another test set was developed by hand, containing words and "non-words" similar to those in the training set. There were 21 words in this second test set, three of which were also in the original training set. For example, the words *lame* and *ink* were in the original training set, so *came*, *pame*, and *ilk* were placed in the test set. The network produced 35.4% of the phonemes correctly in this test set. By inspection of the output generated for each word, it was seen that the network in general learned to pronounce the first phoneme correctly for new words, and then the activation pattern for the rest of the word is a combination of the correct pattern and the correct pattern for a similar word from the training set. For example, the second phoneme for *came* is activated at 0.78 for the correct phoneme AY, and at 0.25 for the phoneme L, due to the input pattern "C L" that it had seen in the word *clay* from the training set.

3.2 Hidden Unit Activation Patterns

The key to understanding the internal representation learned by the network lies in the hidden units. The analysis of the hidden units leads to an understanding of whether or not the network simply "memorized" the words in the training set or if it learned something more meaningful. By inspecting both the hidden unit activation patterns of the network for each word in the training set and the weight vectors associated with each hidden unit, it is possible to find rules that the network learned during training.

Several patterns were found after careful examination of the hidden unit activation patterns for the words in the training set. This was accomplished by looking at each hidden unit individually, and comparing its weights from the input and context units, and its weights to the output units, with its own activation level and the activation level of the output units. In this way, it is possible to see which letters in the input and which states in the context units cause the

particular hidden units to be activated. Likewise, the hidden unit can activate or inhibit each output unit (phoneme) based on its corresponding weight to that output unit.

A first observation was the distinction of the short vowel vs. long vowel sound produced for a vowel in the second position of a word based on whether or not an 'E' occurred as the fourth and last letter. The patterns being compared here are associated with words like *hip* and *hide*, *lack* and *lame*. It was found that there are a group of four hidden units activated in a unique way depending on the particular letter pattern in the input word containing the consonant-vowel-consonant or consonant-vowel-consonant-'E' structure. Depending on the exact combination of input, some of these nodes tend to activate the short-vowel phonemes in the output and inhibit the long-vowel phonemes, while the other nodes activate the long-vowel sounds and inhibit the short-vowel sounds. In this sense, the network has learned to "segment" the word by identifying vowel_E as a grapheme (where "_" is an intervening consonant) and to associate that vowel with a long phoneme.

To better illustrate this pattern, two of the hidden units described above will be discussed in more detail. Figure 2 depicts these two hidden units labeled A and B. Hidden unit A is activated by *hide*, but not by *hip*. Thus, this node activates the long-I output phoneme /ai/ and inhibits the short-I output phoneme /ih/. Hidden unit B becomes active when the word *hip* is presented to the network, and in turn activates /ih/ nine times more strongly than /ai/.

A similar phenomenon is seen when comparing *lame* and *lack*. One hidden unit is active for the word *lame*. This unit has a weight of +1.316 to the output unit corresponding to the correct phoneme /ay/, or long-A, and a weight of -34.2 to the output node for the phoneme /ae/, or short-A.

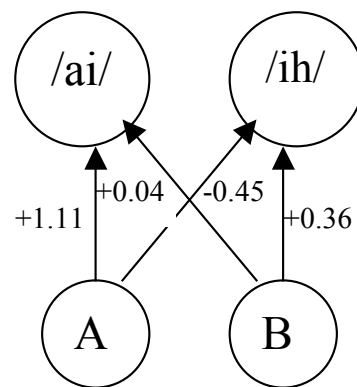


Figure 2: Hidden units A and B with weights to the given output phonemes.

Likewise, the complementary hidden unit that is active for *lack* inhibits the activation of /ay/ more than /ae/.

A second example involves the end of word marker, EOW. This was of particular interest because it was not at all obvious a priori how the network might learn to distinguish words like *me* and *mean* in terms of the number of output phonemes to produce in the output sequence. One would hope that the input generated by the *a* and *n* in *mean* would be enough activation to prevent the network from concluding that it has pronounced the word *me* after saying /m ee/. As it turns out, there are only three hidden units with fairly strong positive weights, greater than 2.5, to the EOW node. All other weights from hidden units to the EOW output unit are less than +0.835, with most of these other weights being negative in direction. At least two of these three hidden units were fully activated (> 0.90) for the EOW node in 88% of the words. Since the input is held fixed throughout the pronunciation of the word, the information used to generate each subsequent output phoneme must come from the context units, which represent the two most recent states of the network, with a greater emphasis on the most recent state. Therefore, the key to activating these three hidden units comes from the context layers. In fact, it is seen that the states that correspond to the most frequently appearing output phonemes have net positive weights from the context units to these three hidden units, and thus serve to activate the EOW output node. In addition, the most common last phonemes, before the EOW signal, have comparably high positive weights from context layer 1 to these hidden units. For example, Figure 3 depicts one of these hidden unit which receives positive input from the context units corresponding to /t/, /n/, and /r/, the three

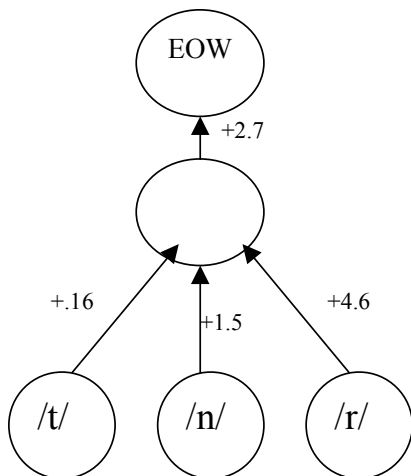


Figure 3: Context units corresponding to the most common final phonemes, with weights to the hidden unit with strong excitatory connection to the EOW node.

most common last phonemes from words in the training set. By inspection, it was seen that these three hidden units were together most active for the EOW node, but not highly active for earlier phonemes in the words.

A related pattern involves the letter X. This letter is the only individual grapheme that corresponds to two separate output phonemes, /k s/. The letter X in the input sequence has negative weights that inhibit the hidden unit described above that correlates to the EOW. This negative input allows the network to produce one more phoneme until there is enough activity in the context layer to trigger the EOW signal. Specifically, there are large positive weights that correspond to a /k/ as the second to last phoneme (+5.618 from context layer 2), and /s/ as the last phoneme (+6.667 from context layer 1), from the context units to this hidden node. Therefore, the presence of the /k s/ sound in the state of the network informs the hidden node in question that the network is finished sounding the phonemes for the letter X.

3.3 Evaluation of Backpropagation Variants

The various backpropagation methods implemented in this study yielded significantly different results. It was found that there was no difference in using basic backpropagation [3] and backpropagation with momentum for this task. Therefore, only that variant with a momentum term was tested significantly. An adaptive learning rate algorithm [2] was found to improve the performance over basic backpropagation. Furthermore, RPROP [5] improved on the performance of the adaptive learning rate algorithm. The use of a penalty term, as described in section 2.1, was found to improve performance in all methods, so it was used consistently. Each method was also tested using various numbers of hidden units and various initial settings, i.e. initial learning rate, initial momentum term. In Table 1, the best simulations for each variant of backpropagation are compared based on the number of training epochs it took each to reach its best RMSE and PCT.

Table 1: Summary of the best results based on the method of backpropagation used.

| TRAINING METHOD | TRAINING EPOCHS | BEST RMSE | BEST PCT |
|-------------------------------|-----------------|-----------|----------|
| Backpropagation with Momentum | 3000 | 0.0700 | 80.06 |
| Adaptive Learning Rate | 5000 | 0.0430 | 91.75 |
| RPROP | 160 | 0.0100 | 99.55 |

4 Conclusions

The resilient RBP network learned very effectively to generate the correct pronunciation for 150 words and was the best of the methods we tried. To our knowledge, this is the first demonstration of effectiveness for resilient backpropagation with sequence processing. This is not trivial at all, since the network is not told how to segment the words to produce the correct phonemes. Some interesting rules of pronunciation discovered by the network were extracted despite the network's distributed representation. Even though the network learned to pronounce the words in the training set, the network generalized poorly to words not in its training set.

It is not fully clear how the network achieved the level of performance necessary to learn to pronounce the words in the training set. It is possible to extract some rules and patterns that the network may have learned, but this process on a network of this magnitude, with 346 total nodes and many weights, is very tedious. It is also possible to see that the network learned the pronunciations of some words while creating no rules at all. Many words in the training set did not follow a grammatical pattern similar to another word. Therefore, the network learned to pronounce these types of words independent of any other word in the training set. The internal representation is very complex. Simply analyzing all the weights by inspection is an enormous task. Perhaps using a better technique for investigating the network's internal representation (clustering, rule extraction, etc.) is the next step in this research. However, even to develop a program to extract some information requires some insight beforehand.

Since the network performed well on the training set in a very reasonable run time, another possible path to take in this work is to train a similar network using a much larger and more diverse training set. The hope in this area would be that the network could better generalize to more words in the English language and even predict the perceived pronunciation of made-up words in the language. The larger training set would allow more patterns to be investigated, since certain segmentation patterns would each

appear more often in the training set. Also, it might be more obvious how the network learns to handle unique words that don't follow any specific segmentation and pronunciation pattern.

Acknowledgment

This work was supported in part by grant R01-DC00699 to the University of Maryland School of Medicine.

References

- [1] Berndt, R. S., C. L. D'Autrechy, and J. A. Reggia. Functional Pronunciation Units in English Words. *Journal of Experimental Psychology: Learning, Memory, and Cognition*. Vol. 20, No. 4, 977-991, 1994.
- [2] Hagan, M. T., H. B. Demuth, and M. H. Beale. *Neural Network Design*. PWS Publishing, Boston, 1996.
- [3] Jordan, M. I. Attractor Dynamics and Parallelism in a Connectionist Sequential Machine. *Proceedings 8th Annual Conf. Cog. Sci. Soc.*, 531-546, 1986.
- [4] Reggia, J. A., P. M. Marsland, and R. S. Berndt. Competitive Dynamics in a Dual-route Connectionist Model of Print-to-sound Transformation. *Complex Systems*, 2, 1998, 509-547, 1988.
- [5] Riedmiller, M., and H. Braun. A Direct Adaptive Method for Faster Backpropagation Learning: The RPROP Algorithm. *Proceedings of ICNN*, San Francisco, 1993.
- [6] Sejnowski, T. J., and C. R. Rosenberg. Parallel Networks that Learn to Pronounce English Text. *Complex Systems*, 1, 145-168, 1987.