*In: Proceedings of the Cognitive Science of Natural Language Processing Conference, Dublin City University, 1996.*

# Ambiguous Data in Connectionist Language Processing Models

**John A. Bullinaria**

Centre for Speech and Language, Department of Psychology,
Birkbeck College, Malet Street, London WC1E 7HX, UK

johnbull@ed.ac.uk

**Abstract:** I consider the problems that neural networks face when learning from (or responding to) ambiguous data. The general discussion is illustrated throughout with a range of language processing examples.

## 1. Introduction

From the point of view of connectionist modelling, we may define a particular input to be ambiguous if the network in question produces, or is expected to produce, two or more different outputs or some blend of those outputs. In this paper I will review the essential features and problems of neural network learning from ambiguous training data (as discussed in Bullinaria, 1995) and show how a number of different approaches may be applied to the problems of ambiguity in a range of simple connectionist language processing models.

There are essentially two reasons why a single network input pattern might correspond to more than one different output pattern :

1. The outputs really are different, but there is additional (context) information missing from the input data.

2. The outputs are actually equivalent, but this fact is obscured by the representation used.

For the first of these we have to distinguish between cases where the context information is never available (e.g. because it is noise) and cases where it is available during training but not during later operation. If the contexts are available during training, we can then distinguish two further sub-classes. First, situations where we need to make a suitable unambiguous response without the context information, e.g. when we are asked to pronounce an isolated homograph. Second, where we know that the context information will eventually become available, but want to be prepared for each of the possibilities, e.g. when we have heard the sound that corresponds to the beginning of more than one word.

For the second reason for ambiguity, the different outputs correspond to different ways of representing the same object. In this case we should aim to learn which representation (i.e. which non-ambiguous subset of a larger ambiguous set of training data) best captures any underlying regularities in that data and hence optimise generalization whilst minimising the problems of over-fitting. We shall see later how the deliberate generation of ambiguous training data can begin to solve some of the long standing representational problems of mapping time sequences, such as the alignment problem for reading and spelling.

If it is acceptable or required for the system to produce the different outputs in a stochastic manner according to some particular probability distribution, then there exist networks, such as the symmetric diffusion networks of Movellan & McClelland (1993), that can do this. It is also possible to model arbitrary conditional probability distributions using mixture models within the standard feedforward network framework (e.g. Jacobs et al., 1991; Bishop, 1995, sect 6.4). I will comment briefly on how these networks operate in the next section, but in this paper I shall concentrate on investigating the extent to which conventional deterministic feed-forward networks can deal appropriately with ambiguous inputs. I will end with a discussion of the extent to which current approaches are sufficient to deal with ambiguity in connectionist language processing networks.

## 2. Learning Probability Distributions

In the sense of data smoothing and interpolation we generally assume that the training data outputs correspond to a fixed "correct" value plus random noise and it is acceptable (in fact usually required) for the network to output an appropriate expectation value derived from the training data. Indeed, in the limit of infinite training data, the network can approximate the conditional average to arbitrary accuracy (e.g. Bishop, 1995, sect 6.1.3). If we only have a small set of training data with output units having only a few (or even only one) different activations for each particular input pattern, then the network output might simply be the average of just these few data points, but more often we will also want it to be adjusted by the training data for the other inputs so that it better approximates the underlying "correct" value. This is standard data modelling.

The problem we need to address here is that such expectation values are not always what is required. If the "correct" data really is multi-valued (with noise added on top of that), the conditional average will clearly not give a good approximation to the data (e.g. Bishop, 1995, sect 6.1.5). Consider homographs and suppose we want to map from orthography to semantics. If we train on "bank" as in river-bank and also on "bank" as in money-bank, it is not particularly useful for the network to produce an intermediate semantic output for "bank", whatever that may be. Movellan & McClelland (1993, p468) argue that rather than producing such output blends, we require networks that can produce entire probability distributions of outputs. This can be done with stochastic networks in a way that is not possible with the corresponding conventional deterministic feedforward network. A network with the letters "bank" as its input would produce the different meanings on its outputs with probabilities determined by the frequencies in the training data, adjusted by context information if any were available. Movellan & McClelland showed how contrastive Hebbian learning applied to symmetric diffusion networks performs gradient descent on the total information gain error function. Then, since this error measure only vanishes when the obtained probability distribution matches the desired probability distribution, we have a way to approximate entire probability distributions. They confirmed this approach with explicit simulations of various discrete and continuous distributions.

One particularly relevant simulation they presented concerned ambiguous translations between Spanish and English. They used a fully connected network with 8 units to form a distributed code for 6 English words, 8 units to form a distributed code for 6 Spanish words, and 4 hidden units. It operates by clamping one language unit set and waiting for the translation to appear as activations of the other language units. For ambiguous words, where more than one translation is possible (e.g. "casa" may correspond to either "house" or "home"), the network produces one or the other possibility with the appropriate probability rather than an unacceptable blend.

A somewhat different approach to the modelling of complete conditional probability distributions $p(\mathbf{t}|\mathbf{x})$ of target $\mathbf{t}$ given input $\mathbf{x}$ involves using a mixture model

$$p(\mathbf{t}|\mathbf{x}) = \sum_{j=1}^{M} \alpha_j(\mathbf{x})\phi_j(\mathbf{t}|\mathbf{x})$$

where $M$ is the number of components in the mixture. If we take a simple Gaussian kernel

$$\phi_j(\mathbf{t}|\mathbf{x}) = \frac{1}{(2\pi)^{N_O/2}\sigma_j^{N_O}(\mathbf{x})} \exp\left\{-\frac{\left\|\mathbf{t}-\mu_j(\mathbf{x})\right\|^2}{2\sigma_j^2(\mathbf{x})}\right\}$$

where $N_O$ is the dimensionality of $\mathbf{t}$ (i.e. the number of output units in the conventional network), an extended network with $(2+N_O).M$ output units (corresponding to $\alpha_j$, $\sigma_j$ and the $N_O$ components of $\boldsymbol{\mu}_j$) can then be used to model the data and hence approximate conditional densities which are not necessarily unimodal for all values of $\mathbf{x}$. The details of such procedures are covered in more detail by Bishop (1995, sect 6.4).

## 3. Feedforward Networks

In this section we consider the extent to which standard deterministic feedforward networks can deal appropriately with the various types of ambiguity and determine when they might be a feasible or preferred alternative to the stochastic networks discussed above.

### *3.1 Missing Context Information*

As noted above, there are many cases to consider. The easiest to deal with is when the context information is available, since we can then simply include it in the inputs of the training data to remove the ambiguity and hence have no problems during training. The potential problems arise if the network is then expected to operate at a later stage without the context information. If the network has learnt to use the context information only when the inputs would otherwise be ambiguous, then the network's performance is likely to be the best that can be expected. If, on the other hand, the network's output depends more generally on essentially irrelevant context information, then it will not only perform unnecessarily poorly when that context information is not available, but it will also have a detrimental affect on the network's generalization performance. In these cases, some form of implicit or explicit restriction on the network's use of irrelevant context is required (e.g. MacKay, 1995). We shall look at this in more detail in Section 4 in the first of our explicit case studies.

Often the dis-ambiguating context information will not be available at all. If this corresponds to the common situation discussed above where it is the result of random noise in the system generating the data, the network should have no problem in producing acceptable outputs when the underlying "correct" data is single-valued. For example, if we trained a network to read aloud using a collection of data from people with a number of slightly different accents, the system would smooth out the training data and end up with some intermediate accent dependent on the frequencies in the training data. If, however, the "correct" data is multi-valued, as for homographs without contexts, we can expect problems. In this case, it is hard to see how we could do better than just picking one target from each ambiguous set and ignoring the rest.

Suppose we do know (or assume) that there should be a unique correct output for each input, but nevertheless the training data still contains examples of the same input producing vastly different outputs. We must then assume that the ambiguities are caused by errors (or outliers) and the way to proceed is to require the network to pick out the correct set of targets and train only with those so that the internal representations and outputs are not contaminated by the erroneous targets. If all the training patterns were independent and we insisted on using a non-ambiguous subset of the training data, then the best we could do would be for each different input to take the target with the greatest prior probability and train on that (or if there were more than one with equal highest probability pick one of them at random). Generally however, for the type of problem that neural networks are typically used for, the training patterns are not all independent and being given the correct target for one input will influence the probabilities of the target for other inputs. For example, given that the word 'five' is pronounced /fIv/ and 'hive' is pronounced /hIv/, we are more likely to believe 'give' is pronounced /gIv/ than /giv/, but we will still have to be able to cope appropriately if we are told that /giv/ is definitely correct or that it has some high probability of being correct. This, of course, is tied in with the whole problem of generalization by neural networks. We know that given enough hidden units and connections they can learn any non-ambiguous input-output mapping, but they will not necessarily generalize, i.e. give useful outputs for new inputs. The aim here is to have the network pick the set of consistent non-ambiguous targets that best captures any underlying regularities in the data without preventing the handling of any special case training patterns that do not follow the regularities. The network will then be in the best possible position to generalize to new input data.

For networks that have already been partially trained it clearly makes good sense when presented with a new ambiguous training instance to choose to train on the output target that already best fits in with the existing knowledge of that problem domain. Given that the network will already be operating by performing gradient descent on some error measure $E$, it seems natural to use $E$ to determine that best fit. The simplest way to proceed would thus be, for each input pattern, to assume that the most appropriate target is the one that already has the smallest $E$ and to train the network on that. To take account of the prior probabilities $P$ we can define some function $F(E,P)$ to give an effective fitness for each training pattern. As before, we only wish to train on one of the set of targets corresponding to each input, but instead of choosing the output pattern with the lowest error $E$ we now use the output with the lowest $F$. Frequently we can do no better than assume equal priors $P$, in which case we can just take $F = E$ again. In Bullinaria (1995) it was shown how the simple relation $F = (1-P).E$ gave reasonable priority to the most likely targets with appropriate responses for $P = 0$ and $P = 1$. A more principled derivation of $F(E,P)$ would require a more detailed knowledge of the problem type and error function $E$, (e.g. see Bishop, 1995, ch 6).

To see how a multi-target approach such as this may also be useful when we are learning from scratch, suppose that we have *npat* input patterns each with *ntarg* equally probable targets, one of which follows some general rule and the rest are the result of some experimental error. Prior to training, the network is given a set of small random weights and so the initial chosen targets for each input pattern will also be random. Then during the first epoch there will (on average) be *npat/ntarg* coherent weight changes attempting to encode the rules and *npat.(ntarg-1)/ntarg* weight changes which will generally at worst be an incoherent set of random changes that will tend to cancel each other out. Often the errors on any given incorrect target will only occur on a small fraction of the output bits and consequently even these targets will tend to contribute to the learning of the rules. The overall result of these weight changes will be that the network's outputs will tend towards those corresponding to the rules, even for the inputs that were trained on the wrong targets. After a number of epochs (depending on the learning rates, number of targets, etc.) the lowest error target for all the inputs will be the one corresponding to the rule, and training will then proceed as though we had a non-ambiguous training set. The network will similarly settle into an appropriate set of training patterns if the prior target probabilities are different or if the training data consists of a whole series of rules, sub-rules and exceptions.

### *3.2 Inappropriate Representations*

This second main reason for ambiguity, in which the different targets are actually equivalent in some way, is more difficult to deal with by any system. The problem can often be solved by carefully choosing an alternative representation for the training data, but this approach is open to much criticism since such pre-processing of the training data can leave very little of the original problem for the system to solve. In other cases the ambiguities can be introduced deliberately precisely because we wish to avoid the need to pre-process the training data.

We generally do not want our system to produce an output intermediate between two equivalent targets. For example, if one target represented an angle of 30° and another represented the equivalent 390°, a system that output the average 210° would not be particularly useful, whereas we might expect the same system to produce 210° given targets of 209° and 211°. The obvious way to proceed would be to pre-process the training data by hand so that all the angles were in the range 0° to 360°. Few people would object to this, but real world problems are not necessarily going to be this straightforward and transparent. A more satisfactory approach in general would be to have our network pick its own non-ambiguous subset of the targets based on some fitness function $F$ as above. This will not always be as easy as picking only targets corresponding to angles in the range 0° to 360° and even in this case it relies on the training data being sufficiently representative to have one of the targets in this or some other 360° range for (almost) all the inputs. In fact, one could imagine situations in which it would be worth introducing extra training patterns 'near' or 'in between' those already present in order to increase the chances of arriving at a complete set of correct targets (e.g. given targets corresponding to angles 209° and 211° it might be reasonable to also give the network the opportunity of settling on the target corresponding to 210°). This possibility has been examined in more detail in Bullinaria (1994b). Generally, as soon as one coherent set of targets begins to dominate the weight changes it will recruit more and more of that set into the training data subset being used and exclude the others in the same way as the erroneous targets discussed previously. In cases where there is more than one equally good non-ambiguous subset of the training data (e.g. corresponding to different 360° ranges) we can rely on the initial random weights and/or the random order of example presentation to break the symmetry.

Of particular interest to us is the case where the ambiguous training data has been generated deliberately, for here we know that it is fully representative. This kind of ambiguity can then actually be used to our advantage to allow a network to choose on its own (from a set of possibilities) the most appropriate representation to solve a problem rather than requiring the modeller to choose it by hand. This approach will be illustrated explicitly in Section 4 with my recent models of reading and spelling.

### *3.3 The Multi-Target Approach*

The preceding discussion suggests that a simple multi-target approach may be sufficient to deal effectively with many types of ambiguous training data in feedforward neural networks. This approach was investigated empirically by Bullinaria (1995) and found to be rather successful. If, for each input pattern with a set of possible target patterns, the network only trains on the target that already has the lowest effective fitness *F*, then (for a sufficiently representative set of training data and a sufficiently small learning rate) the network does indeed settle down into using only the best non-ambiguous subset of the training data.

The multi-target approach is essentially the same for both of the types of ambiguity discussed above. In the first case the network picks the correct target from each set that also includes errors. In the second case it picks the best target to represent each given output from a set of possible ways of representing it. It follows that we can apply the same multi-target procedure to any set of ambiguous training data without necessarily having any information about what might be causing the ambiguities.

## 4. Explicit Language Processing Examples

The above general ideas will now be illustrated with three explicit examples from simple connectionist models of language processing tasks.

### *4.1 Homographs in Reading*

A typical language processing ambiguity that illustrates the use of context information is provided by homographs, i.e. words that are spelled the same but have different meanings and/or pronunciations. If our network can learn complete probability distributions, there should be no problems dealing with homographs. Here I will show how we can simulate such ambiguities in a conventional feedforward network model of text to phoneme conversion (i.e. reading aloud) and study the use of context in its training and testing. We shall base this investigation on the simplified framework of Bullinaria (1996) shown in Figure 1. The input layer consists of units for each of a small subset of possible onset

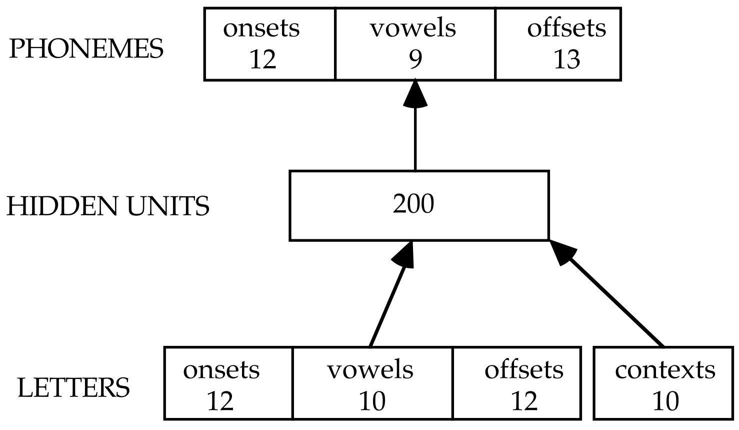

*Figure 1. Simplified model of reading aloud incorporating context information.*

| Training type | Homograph errors | Non-homograph errors | Generalization (%) |
|---|---|---|---|
| No homographs | 12.0 (0.0) | 0.0 (0.0) | 88.3 (0.9) |
| Multi-target | 12.0 (0.0) | 0.0 (0.0) | 88.3 (0.4) |
| No context | 23.8 (0.4) | 0.0 (0.0) | 86.3 (1.1) |
| Context (1 bit) | 18.4 (0.5) | 24.7 (2.0) | 83.5 (1.4) |
| Context (1 of 5) | 16.5 (0.8) | 0.0 (0.0) | 84.6 (1.6) |
| Reduced (1 bit) | 21.2 (0.9) | 2.7 (1.3) | 86.3 (1.2) |
| Reduced (1 of 5) | 21.3 (1.4) | 0.0 (0.0) | 86.5 (1.6) |

*Table 1. Performance of the simplified reading model on isolated words.*

consonant clusters, vowel clusters and offset consonant clusters plus a block of units in which we may provide context information. The output layer consists of similar units for the phonemic onsets, vowels and offsets. The network is feedforward fully connected with one hidden layer, sigmoidal activation functions and trained by back propagation with cross-entropy error measure, learning rate 0.05 and no momentum or weight decay. The training data consisted of 526 monosyllabic words including 12 homograph pairs. Generalization was measured with a random set of 200 non-words not appearing in the training data. On testing, an output was deemed to be wrong if any bit had a squared activation error greater than 0.1 so that only distinct output patterns (rather than blends) were counted as correct.

The investigation consisted of training the network for 2000 epochs in seven different ways, by which time the non-ambiguous training data had been learnt correctly in each case. To minimise the effects of statistical fluctuations, each variation was trained 12 times from different small random initial weights. The results are shown in Table 1 with the standard deviations in brackets. Since the network can produce only one output for each input there will be a minimum of 12 homograph errors in each case. The first case is the control condition with no context information at all and the least regular of each homograph pair removed. In this case the 12 homograph errors are the 12 removed from the training data. The next case used our multi-target training procedure on the full training data with no context information so that the network picked a non-ambiguous set of training data for itself. The 12 homograph errors each run were again those excluded from the training data. There was no loss in generalization performance over the no homograph condition. The third case also used the full training data with no context information but now with standard (rather than multi-target) training. Here the outputs learnt for the homographs are blends so that virtually all homograph outputs are deemed wrong. There are still no errors on the non-homographs but there is a slight degradation of the generalization performance.

In the remaining cases context information was made available during training but not while testing. For the fourth variation each training word was assigned a single fixed context unit which was activated during training to remove the ambiguities. As long as the context was available the performance was good but when the context was not available it not only resulted in numerous blend errors for the homographs but also in many errors in the non-homographs. This is because the context information is being used at the expense of the non-ambiguous orthographic information, particularly for words that include irregular orthography to phonology mappings. This also has the effect of reducing the generalization performance. One easy way to reduce this over dependency on the context information is to reduce its reliability. In the fifth variation we therefore assigned five different context units to each word and used just one of them at random on each training occurrence. This completely cured all the non-homograph errors and also some of the homograph errors but some degradation of the generalization performance persisted. The sixth variation attempted to remove the context over dependency by reducing the learning rate for the connections from the context units by a factor of ten. This had the desired effect of reducing the non-homograph errors by a large factor and also increased the generalization performance. Finally, the two context over dependency measures were applied together resulting in isolated word performance similar to the case with no context information during training but with the advantage of perfect performance when the context was available. The problem of over use of context information may also be solved in a more principled manner using weight decay in the Bayesian Automatic Relevance Determination approach of MacKay (1995). In more realistic models that include word frequencies, the higher frequency words will tend to dominate the training and result in any output blends being skewed. This is likely to result in the more dominant homograph pronunciations being deemed correct as tends to be found experimentally with humans.

### *4.2 Cohort Effects in Speech Recognition*

A related example of ambiguity is provided by the cohort effect in speech recognition (e.g. Marslen-Wilson, 1987). There is experimental evidence from priming studies of humans that ambiguous word beginnings (e.g. "capt") do indeed result in the (partial) activation of more than one possible completion (e.g. "captain" and "captive"). From the point of view of computational efficiency it certainly makes good sense to use the speech information to map to semantics as it becomes available rather than waiting for all the context information (i.e. the word endings) to become available. It is not difficult to see how networks that can learn complete probability distributions would be able to make use of partial speech information in this way.

This effect can also be modelled easily in a simple feedforward network that maps between phonology and semantics with an architecture similar to that of Figure 1 (Bullinaria, 1996). The input phonology is again represented by units for each of the possible onset, vowel and offset phonemes for English monosyllables. Rather than attempting to generate realistic semantic representations, each word was simply assigned a random sparse 30 bit binary vector. The output semantic unit activations were then determined by these vectors with each of the 'on bits' taken to represent an activated semantic micro-feature. The model has no explicit 'lexicon' as such, but it is implicit in the distributed semantic activations. The activation is taken to cascade through the network from the inputs through the hidden layer to the outputs in the same manner as in a recurrent network. This has been shown elsewhere to provide realistic simulated reaction times, priming effects and so on (e.g. Bullinaria, 1996). The input sequentiality can then be simulated simply by having the phonological input activations build up in order over a number of times slices. Figure 2 shows how the corresponding total semantic output activations then build up over time for one representative set of four words. Initially all the words consistent with the first phoneme /n/ are activated. Then, as the second phoneme /E/ appears in the input, the word /nOz/ loses activation whilst /nE/ and /nEs/ continue rising. Finally, as the whole phonology becomes available, the unique consistent word /nEs/ remains active whilst the others fall to the levels appropriate to their semantic relatedness with the actual word.

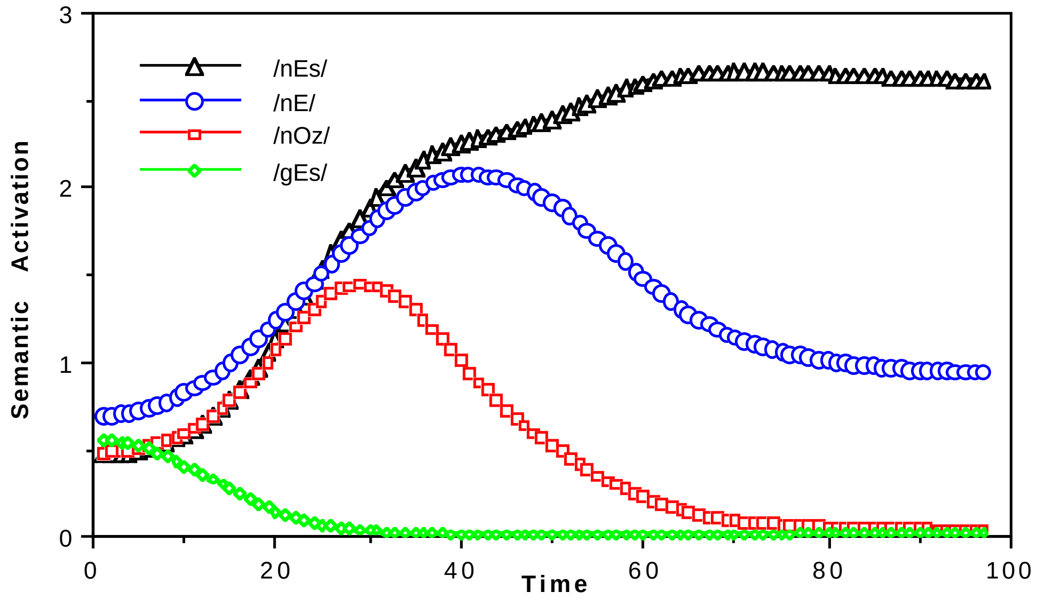

*Figure 2. The 'cohort effect' arising from sequential speech input.*

## *4.3 The Alignment Problem in Reading and Spelling*

Suppose we wish to model sequences of inputs and outputs, for example the {*letters*} and {*phonemes*} in a reading or spelling system. The aim is to predict the outcome of any particular sequence of inputs, given only a training set of previous sequences of inputs and outputs. Typically there will not be a simple one-to-one relationship: sometimes one input may produce many outputs (e.g. ‘j’ $\rightarrow$ /dZ/) and other times it may require many inputs to produce a single output (e.g. ‘th’ $\rightarrow$ /T/). Moreover, whether or not an input results in a particular output depends on the presence of previous or subsequent inputs and the order of events in time is important, i.e. we have to take into account time dependent context information. There are two obvious ways to introduce time effects into neural network systems. One is to allow the network to have a sufficiently general set of recurrent connections that it can effectively store up information about sequences of events. The other is the moving window approach in which each input sequence is split up into a number of sub-sequences by a ‘window’ which moves across the input sequence. In this case the network is trained with the sub-sequences as inputs and the outputs corresponding to the input in the centre of the input window (e.g. Sejnowski & Rosenberg, 1987). In both cases the network can cope as long as we know how the inputs and outputs are meant to line up in time. The problems arise when there can be time delay effects which mean that it is not possible to line up the causes and effects uniquely in time. This is where the ambiguities come in: for a single input pattern there may be more than one equivalent way of lining up the output and hence more than one possible target vector. Here we do not want to model target probabilities, but rather, we want to pick the most appropriate of the possible targets.

For example, for a reading system, the word ‘bathe’ is pronounced /bAD/ but, given only this information, there is no way of knowing which subsets of letters correspond to which subsets of phonemes, i.e. how the letters and phonemes line up. If alignment is denoted by phonetic nulls /–/, possibilities include /bAD––/, /bA–D–/, /b–A–D/, /–b–AD/ and so on, and each corresponds to a particular set of letter to phoneme rules. Each of these sets of rules (or targets) are equally valid for this single word but, for a system that has to learn many words and be able to generalize to new words and non-words, some are more appropriate than others. The two, somewhat unsatisfactory, solutions to this problem used in the past are:

1. Pick one of the possibilities for each word by hand and hence have a non-ambiguous set of training data. This is the approach adopted by us in the reading model above and by Sejnowski & Rosenberg (1987) in NETtalk, but this degree of pre-processing of the training data leaves very little of the original problem for the network to do and is usually considered unacceptable when modelling the brain.

2. Use a radically different representation for the training data in which the ambiguities do not occur. This is the approach adopted by Seidenberg & McClelland (1989) but it made the interpretation of the networks output difficult and presented difficulties in understanding the nature of the internal representations. Their model was also restricted to mono-syllabic words and had poor generalization performance.

The alternative approach suggested here is to generate ambiguous training data including each possible target (i.e. alignment) for each training word and allow the network to use the multi-target training approach to choose the most appropriate alignments for itself based on the network output activation errors. Using a NETtalk style moving window architecture, Bullinaria (1994a, 1995) showed that this approach did indeed work rather well. Given a sufficiently representative set of 2998 training words, common potential rules such as ‘b’ $\rightarrow$ /b/ and ‘a’ $\rightarrow$ /A/ dominated the weight changes over others such as ‘t’ $\rightarrow$ /A/ and ‘a’ $\rightarrow$ /b/ and the network automatically settled into a useful set of targets with the kind of letter-phoneme correspondences that we would consider natural. Unlike other models, it can thus handle multi-syllabic words and long range dependencies with no need for pre-processing of the training data nor complicated data representations. This approach also led to a generalization performance far superior to earlier reading models (e.g. Seidenberg & McClelland, 1989) with realistic simulated reaction times and surface dyslexia and is equally applicable to models of spelling and past tense acquisition (Bullinaria, 1994a).

## 5. Discussion and Conclusions

We have seen how the learning of unwanted output averages or blends can cause serious problems for neural networks that are trained on ambiguous data. Networks such as stochastic diffusion networks or mixture models that can approximate complete distributions of conditional probabilities do not suffer from these blends. However, to date, such systems have rarely been used for language processing models.

The conventional deterministic feedforward networks that form the basis of many simple language processing models do suffer from these output blends, but we have seen in this paper that they are not necessarily as bad at dealing with ambiguous data as often imagined. They can be set up to have additional input units for context information, and it is possible for them to learn to use the available context information in a sensible manner that allows them to continue operating appropriately if that information later becomes unavailable. They can also be set up to pick their own non-ambiguous subset of the training data in such a way as to optimise their overall performance. This multi-target approach can be used either to automatically remove errors from the training data or to develop an efficient representation of the training data from a set of possible representations.

Several simple connectionist language processing models have been presented to illustrate the possibilities for dealing appropriately with various types of ambiguity. However, it remains to be seen how well these will fit together into a coherent overall model that is consistent with human behaviour.

## References

Bishop, C.M. (1995). *Neural Networks for Pattern Recognition*. Oxford, UK: Oxford University Press.

Bullinaria, J.A. (1994a). Modelling reading, spelling and past tense learning with artificial neural networks. *Brain and Language*, 59, 236-266.

Bullinaria, J.A. (1994b) Noise Reduction by Multi-Target Learning. *Proceedings of the European Symposium on Artificial Neural Networks*, 193-198. Brussels: D facto.

Bullinaria, J.A. (1995). Neural network learning from ambiguous training data. *Connection Science*, **7**, 99-122.

Bullinaria, J.A. (1996). Connectionist models of reading: Incorporating semantics. Submitted.

Jacobs, R.A., Jordan, M.I., Nowlan, S.J. & Hinton, G.E. (1991). Adaptive Mixtures of Local Experts. *Neural Computation*, **3**, 79-87.

MacKay, D.J.C. (1995). Probable networks and plausible predictions: A review of practical Bayesian methods for supervised neural networks. *Network: Computation in Neural Systems*, to appear.

Marslen-Wilson, W. (1987). Functional parallelism in spoken word recognition. *Cognition*, **25**, 71-102.

Movellan, J.R. & McClelland, J.L. (1993). Learning continuous probability distributions with symmetric diffusion networks. *Cognitive Science*, **17**, 463-496.

Seidenberg, M.S. & McClelland, J.L. (1989). A distributed, developmental model of word recognition and naming. *Psychological Review*, **96**, 523-568.

Sejnowski, T.J. & Rosenberg, C.R. (1987). Parallel networks that learn to pronounce English text. *Complex Systems*, **1**, 145-168.