TRAINING A NEURAL NETWORK TO BE A CONTEXT SENSITIVE GRAMMAR

ROBERT F. SIMMONS
YEONG-HO YU

JULY 1990 AI90-135

Training a Neural Network to be a Context Sensitive Grammar [1]

Robert F. Simmons and Yeong-Ho Yu @cs.texas.edu

Department of Computer Sciences, AI Lab
University of Texas, Austin Tx 78712

Abstract

A system is described for training a neural network to represent a context-sensitive grammar. The trained network is accessed by a shift/reduce parser to analyze indefinitely long, deeply embedded sentences. The parser is a brief sequential program that queries the trained network, manages pushdown stacks, and records states of the parse so that a final analysis may be shown. It is argued that this hybrid approach generalizes immediately to case analysis of indefinitely long sentences.

1 Introduction

Moderately successful experiments in training feed-forward neural networks, FFNN, to translate short English sentences to Spanish [Allen, R. 1987], between German and English [Yu and Simmons 1990], and to case structures [McClelland and Kawamoto 1987, Miikkulainen and Dyer 1989] suggest that neural network technology may offer considerable benefit to computational linguistics. A significant difficulty in the NN technology is the restriction to fixed length input/output strings [2]. Recurrent neural nets introduced by Jordan [1986] have been used with some success to deal with continuing strings as in [St. John and MacClelland 1989, Miikkulainen and Dyer 1989] but this is less than a completely satisfying solution in that the recurrent technique has not been shown to manage embeddings, it can remember only a limited past context, and in applications to case-analysis parsing, it requires a fixed length output.

Other techniques have been used to deal with indefinitely long strings of input symbols. The epochal research of Sejnowski and Rosenberg [1988] that translated indefinitely long strings of words to speech used a sequential program to feed a string of characters through a fixed length network to map from successive contexts of characters into phonemic features. We believe that this is the most profitable approach for dealing with indefinitely long input.

In this paper we describe a hybrid system in which a sequential program provides successive states of a context-sensitive syntactic analysis of a sentence as training input to a fixed length network. These successive states represent a context-sensitive grammar. After the network has been trained, another sequential program presents successive segments of the input string to the trained network and accumulates the resulting parse states into a tree that is a phrase structure analysis of the sentence.

[1] This work is sponsored by the Army Research Office under contract DAAG29-84-K-0060.
[2] Cottrell's [1989] monograph surveys the original connectionist approaches to the problem.

```
Input <- syntactic string, Stack <- nil, Newsymbol <- nil
Count <- 0, Limit <- 300
Loop Count <- (1+ Count)
If (newsymbol <- (reduce (top stack)(next stack)))
begin (pop stack)(pop stack)(push newsymbol stack) end
          Else (push (pop input) stack) ;; shift
        If (eq (top stack) 'snt) and (null Input) (return 'snt)
          Else If (> Count Limit)(return 'fail)
end-loop
```

Figure 1: Shift-Reduce Recognizer

2 The Problem

We would like to obtain the advantages of pattern generalization, graceful degradation, and learning-by- example provided by neural net technology in application to syntactic analysis of English sentences. But English sentences may be indefinitely long and contain indefinitely many embedded structures, while simple neural nets are fixed-size structures whose computational complexity increases at least by the square of the number of nodes. Thus it is not reasonable to create a net that would accept sentences up to one hundred words — a number greater than the length of 99% of the sentences occurring in newspaper text. Instead, we must trade time for space by mapping successive segments of the input into states of the network.

A shift-reduce parser appears to be a natural approach for this task. In this class of parser there is a stack and an input string of syntactic classes, each of which may be indefinitely long. In the initial state the stack is empty. The first step is to *shift* the first element of the input string onto the stack. In the next and subsequent steps, a grammar is consulted to determine whether the top two elements of the stack *reduce* to a single symbol. If so, they are replaced by the new symbol; if not, a new element is shifted onto the stack. The procedure continues until the string is empty and the stack contains only the symbol Sentence. For convenience we restrict the grammar to binary rules that reduce no more than the top two stack elements.

The recognition algorithm shown in Figure 1 shows the essential flow of control for a shift-reduce parser, but in practical use it would require additional code for recording a parse and for backing up or for recording all paths of a parse. The procedure **reduce** is required to determine if the top two elements on the stack form the right half of a grammar rule and if so, to return the new, left half, symbol, as for example, **(Reduce np vp) $->$ snt, (Reduce art adj) $->$ nil.**

Usually this class of parser applies a context-free grammar. In the adaptation to neural nets, it is natural and desirable that the grammar be context-sensitive. Complete context-sensitivity could be achieved if every rule in the grammar included the complete stack and remaining input for each state of the parse for each possible sentence. We abbreviate this unreasonable requirement by providing the top five elements of the stack and the next five elements of the input string. Each such ten-symbol description represents a state of the parse, and a succession of such states terminating with the symbol *snt* represents a complete parse of a sentence. The syntactic tree can be extracted from such a list by a brief sequential program.

This approach gives a basis for a neural network to map each state of a parse into an immediately succeeding state.

$$stack_i input_i \Rightarrow stack_{i+1} input_{i+1}$$

Thus we are defining rules for bottom-up, context sensitive parsing as having a left-half initial state mapping to a right-half subsequent state. In the following example we separate the stack from the input with the symbol, "*". The letter "b" stands for blank, art for article, adj for adjective, n for noun, p for preposition, and v for verb. In the sentence string, *art adj n p n v n* representing sentences such as "the old man from Spain ate fish," we show two states following the initial two states that shifted an art and then an adj to the stack.

(b b b art adj * n p n v n) – > (b b art adj n * p n v n b)
(b b art adj n * p n v n b) – > (b b b b art np * p n v n b)

The left half of the rule shows five elements of stack followed by the next five elements of the input string; the right half shows the next state. In the first rule the first element of the input was shifted onto the stack, resulting in the new state shown in the right half of the rule. In the second rule, the right half of the first rule became the input. This enabled a reduction of the adj-n pair to an np. The resulting new state is shown as the right half of the rule. In this manner the successive states of a parse may be shown for an indefinitely long string. The stack operations can also account for indefinitely deep embeddings.

In our initial explorations we trained nets with exactly the ten symbol input/output patterns illustrated above. But it soon became apparent that the training could accomplished more rapidly with ten input symbols and three outputs. Instead of completely specifying the output symbols, we indicate *shift* or *reduce* and the resulting top two elements of the stack. The example above then appears as follows:

(b b b art adj * n p n v n) – > (shift adj n)
(b b art adj n * p n v n b) – > (reduce art np)

The sequential parser creates the new state by applying the *shift* or *reduce* to the input, changing the stack as indicated, and then feeding the new input to the training network.

Let us develop a fragment of grammar for the sentence, *The old man from Spain eats fish.*

```
the old man from Spain eats fish
art adj  n   p     n    v    n

(b b b b b * art adj n p n)(shift b art)
(b b b b art * adj n p n v)(shift art adj)
(b b b art adj * n p n v n)(shift adj n)
(b b art adj n * p n v n b)(reduce art np)
(b b b art np * p n v n b)(reduce b np)
```

```
(b b b b np * p n v n b)(shift np p)
(b b b np p * n v n b b)(shift p n)
(b b np p n * v n b b b)(reduce np pp)
(b b b np pp * v n b b b)(reduce b np)
(b b b b np * v n b b b)(shift np v)
(b b b np v * n b b b b)(shift v n)
(b b np v n * b b b b b)(reduce np vp)
(b b b np vp * b b b b b)(reduce b snt)
(b b b b snt * b b b b b) nil
```

As a result we have a set of inputs, the left lists, and outputs, the right lists, that can be used to train a feedforward neural net[3]. After training the net with a back-propagation technique, the trained net represents a context-sensitive grammar that can be used by a shift-reduce parser such as that shown in Figure 1.

At first it was quite tedious to transform sentences by hand into the desired sequence of stack-input states, but we soon developed programs to minimize the amount of typing required, and now a grammar-acquisition editor presents an input and asks for the correct output. If the output is *shift*, the system shifts input to the stack, presents the new input, and continues; if *reduce* it expects a new top element for the stack, creates the new input and continues. The result is a very helpful system for creating the desired grammar.

3 Experiments

For our experiments we needed to prepare a grammar of the above form and a parser that would query the resulting trained networks. We also needed various programs to aid in preparing the grammar and translating it to the exact forms required by our NN simulator.

3.1 Preparing the Grammar

We selected three texts, each comprised of 10-12 sentences. Two were brief articles on hepatitis and measles taken from "The Young People's Encyclopedia of Science" and one was a New York Times article, circa 1976 that described a robbery and assault. The sentences in the first two articles ranged from as few as ten to about twenty-five words in length. The newspaper article included very complex sentences ranging from twenty to fifty words in length.

We assigned syntactic wordclasses by hand to each sentence and prepared the input data following the procedure shown above. As a result we obtained 1012 rules, 552 from the two disease texts and 460 from the news article.

3.2 Training the Nets

In initial training experiments with the disease grammar, we discovered that back-propagation training resulted in no more than 98 percent correct after 2000 epochs with no prospect of further improvement. We therefore shifted to the use of the *descending epsilon* approach. This approach [Yu and Simmons 1990] uses correctness as a major criterion of success in

[3]Note that the succession of left halves are sufficient to define the parse.

contrast to the criterion of minimizing mean squared deviations. *Epsilon* is a measure of the margin of error tolerable in training; when used during learning, if the the target pattern minus the output pattern is less than *epsilon* we cease back-propagation on that pattern.

$|(t_{pi} - o_{pi})| < epsilon \rightarrow error = 0$

By changing *epsilon* in a descending pattern we usually achieve 100 percent correctness on the training set, providing it contains no contradictions. The descending epsilon method is analogous to an annealing technique in which first it is required that all training instances be correctly learned within a loose tolerance of .5, then .4, ... finally .05.

Using *descending epsilon* we were able to train each of the grammars to 100 percent correctness. For the combined measles and hepatitis grammar we required 380 epochs; for the news article, 300, and for all texts combined, 725. We used learning rates, *eta*, of .1, momentum .9, and included 60 input, 7 output, and from 30 - 50 hidden units, increasing the number as the number of rules increased. The training was accomplished on a Symbolics Lisp machine, in a Lisp version of Back Propagation that includes a dictionary shell for translating from symbols to the binary patterns required by the neural network, and from binary patterns back to readable symbols.

3.3 Generalization

With about 250 rules from the hepatitis text, the resulting trained net recognized 28.8 percent of the rules for the measles text. The net trained with 552 rules from the combined hepatitis and measles texts correctly recognized 41 percent of the newspaper text. The net trained with 460 rules from the news article recognized 36 percent of the measles/hepatitis text.

In previous studies with large samples of text, we have found that if we have samples representing 25 percent of the relevant combinatorial space, we can expect up to fifty percent generalization; with a 50 percent sample we can obtain about 90 percent generalization. We can guess that our subset grammars of about 500 rules may represent a sample of about 15-20 percent of the grammar required for the class of texts we have studied. The total 1000 rule grammar may exceed 25 percent. Additional texts must be analyzed and studied before these guesses about generalization can be verified or rejected.

4 The Hybrid Parser

The parser is a sequential program that takes as inputs a sentence, the corresponding syntactic string, and (when available) the list of correct stack output states. It maintains two stacks; one is simply the stack whose top five items becomes part of the input to the net, the other maintains constituents so that at the end of the process, the parse can be printed as the list structure of which *Snt* is the head. Since the parse may fail, a count is maintained in the loop and when greater than a reasonable number of constituents, say 300, the parse terminates with failure.

Figure 2 shows the essential algorithm in pseudo-lisp. It can be augmented with logic to compare each computed state with the next state from the list of correct states and produce a score, showing the correspondence between the actual and correct parse.

```
Define Parse (text, syn-string, states)
Pairs <- (Pairlis syn-string text), Context <- syn-string, Stack <- nil
Pairs-stack <- nil, Count <- 0

Loop ;;begin loop
Count <- (1+ Count)
Right-half <- Ask-Net(Append (reverse (make-vec 5 stack))
                             (make-vec 5 Context))
;; makes input vector, Ask-net returns value of Right-half
;; by querying the trained neural net
If (Right-half = ('Reduce sym1 sym2))
         begin (pop stack)(pop stack)(push sym2 stack) ;reduce
          (push (list sym2(reverse (pop Pairs-stack)(pop Pairs-stack)))
                Pairs-stack) end
else begin (push (pop context) stack) ;; shift
                  (push (pop Pairs) Pairs-stack) end
;; maintaining two stacks in parallel
If (and (eq (top stack) snt)(null Context)(null(nxt stack))
(return (prettyprint Pairs-stack)) ;print parse
If (> count 300) (return fail)
;;end loop
;; (make-vec n lst) extracts from lst n items, padding with 'blank
```

Figure 2: Shift-Reduce Hybrid NN Parser

Jordan, Michael, "Serial Order: A Parallel Distributed Processing Approach," Tech Report ICS-8604, Inst. for Cog. Sci., Univ. of Calif., La Jolla, Calif., 1986.

Leow, Wee-Keng, and Simmons, R.F.,"A Constraint Satisfaction Network for Case Analysis," AI Tech Report, AI90-129, Dept. Comp. Sci,, Univ. of Texas, Austin, Texas, 1990.

McClelland, James L. and Rumelhart, D. E., *Parallel Distributed Processing*, Vols. 1 and 2, MIT Press, Cambridge, Mass., 1986.

McClelland, J.L., and Kawamoto, A.H., "Mechanisms of Sentence Processing: Assigning Roles to Constituents," In McClelland and Rumelhart, 1986.

Miikkulainen, Risto, and Dyer, M., "A Modular Neural Network Architecture for Sequential Paraphrasing of Script-Based Stories", Artif. Intell. Lab., Dept. Comp. Sci., UCLA, 1989.

Sejnowski, Terrence J., and Rosenberg, C., "NETtalk: A Parallel Network that Learns to Read Aloud", in Anderson and Rosenfeld (Eds.) *Neurocomputing*, MIT Press., Cambridge Mass., 1988.

St. John, Mark, and McClelland J.L., "Applying Contextual Constraints in Sentence Comprehension," *Proc. Tenth Annual Conf. of Cog. Sci. Soc.*, Montreal, Quebec, 1988.

Yu, Yeong-Ho, and Simmons, R.F. "Descending Epsilon in Back-Propagation: A Technique for Better Generalization," In Press, *Proc. Int. Jt. Conf. Neural Networks*, San Diego, Calif., 1990.

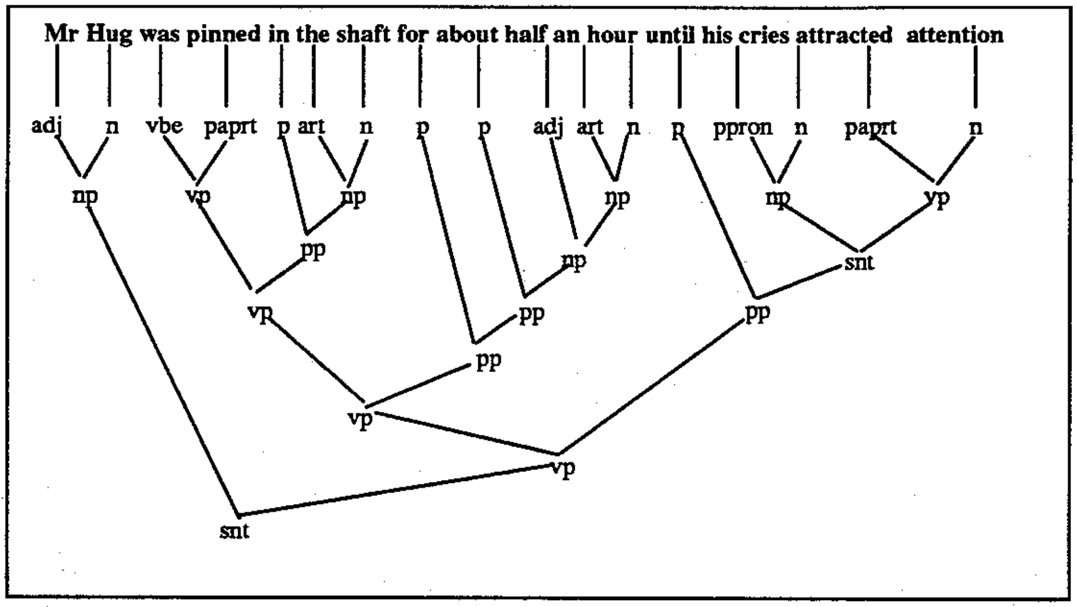
Mr Hug was pinned in the shaft for about half an hour until his cries attracted attention
adj
n
vbe
paprt
p
art
n
p
p
adj
art
n
p
ppron
n
paprt
n
np
vp
np
pp
vp
np
pp
pp
vp
vp
snt
np
vp
snt
pp
snt

Figure 3: Computed Syntactic Analysis

$$Score = correct/total * 100$$

After the network was trained, the entire set of sentences parsed with only a half dozen errors involving prepositional phrase attachments and scope of conjunctions. If the grammar had not been trained to 100%, there would have been many errors in the analyses. An analysis of one of the news-text sentences using the trained net is shown in Figure 3.

5 Discussion

5.1 Syntactic Word Class and Sense Selection

Prior to the experiments reported above, we explored possibilities for training networks to recognize syntactic word-classes in the context of sentence usages, and the capabilities for networks to recognize sense meanings of words in context.

To train syntactic classes we presented the data as a stream of text words and a stream of corresponding syntactic classes. Each stream was moved one term at a time through a window which was trained to assign the syntactic class to each word in a seven-word context. The net was trained to 99.4% accuracy in five hundred cycles for a 150 word sample, using simple back propagation. We have no doubt that the descending epsilon technique can train for recognizing syntactic word-classes to 100% accuracy.

In our first experiments with the parsing technique described above, we trained the net by giving as input the English string for each sentence; the canonical output was the syntactic word-class that was shifted to the stack or the reduction of two stack elements to one. Notice that the context was English words and the stack elements syntactic terms. For this training we generally required on the order of 2000 epochs to learn *both* the syntactic classes and the grammar for a 250 rule system[4]

In our concern with selecting correct word senses in context, we used a sample of 300 ten-word strings from AP Newswire text each of which represented a usage of the word *light*. We hand-coded each usage of the word as *light-1, light-2, ... light-6* and trained the net with a sample of 200 contexts. The accuracy on the training sample was 99%, achieved in approximately 500 epochs. Generalization of the trained net to recognizing usages in the 100 new sample contexts was 70%. These results are averages from repeating the experiment ten times. This preliminary study of recognizing word sense in context showed that although neural nets are an effective computation technique for the purpose, massive amounts of training data must be prepared to train networks to select correct sense meanings for even a small vocabulary.

These studies of training word-class and sense selection encourage us to believe that the hybrid shift/reduce parser can be developed into a worthwhile system that can be trained to accomplish accurate syntactic analysis and word-sense selection for large amounts of text data — but the cost of preparing training examples and achieving the training will be high.

6 Problems for Further Research

Syntactic analysis is usually a first stage computation on the way toward a deeper structure of understanding which is often presented as a case analysis [Allen, J. 1985]. One of the strengths of the back propagation technique is its capability to develop a function that maps directly from text strings to their case analyses [See St. John and McClelland, Miikkulainen and Dyer, Yu and Simmons, Leow and Simmons]. The weakness in the current approaches is their limitation to fixed length strings for input and/or output. Although there is not space to report them here, we have initial successful experiments that train a neural net to be a shift/reduce case grammar that accepts indefinitely long, complex sentences.

Generally there is a problem of linguistic classification: to determine fine enough syntactic classes to obtain case analyses without training a net to form a massive grammar based on word co-occurrences. Cook [1989] offers twelve subcategories for verbs to provide sufficient distinctions for case analysis, but a comparable system for classifying nouns is still lacking. Given a classification system, back propagation methods can train a network to map words to syntactic classes in context. But there are significant difficulties associated with the development of any large dictionary, and the training of networks to map 50-100 thousand words to their contextually determined syntactic classes will prove a formidable task.

In addition to syntactic class determination, word-sense selection is an even larger and more difficult task. On the average, a collegiate dictionary assigns 2 to 3 sense meanings for each word. We have found that nets can be trained to select an appropriate sense in context, but again, the project to prepare training data and to train a set of networks to do this for thousands of words would be an arduous one.

[4]We chose to concentrate our present work on the study of parsing syntactic strings so that we could obtain some estimate of the generalization capabilities of the grammar. Further developments toward a practical parsing system may favor simultaneous training of syntactic class and grammar.

With regard to the purely syntactic parser, it is important to observe that it is deterministic — it provides one parse only. Since sequences of syntactic classes are not sufficient to resolve attachment ambiguities, there is no assurance with any amount of syntactic context that the parse will be correct. Most attachment ambiguities occur when we attempt to decide for a prepositional phrase, which preceding noun or verb it modifies. Relative clauses offer similar difficulties in deciding which preceding noun they modify, and conjunctive phrases are frequently ambiguous in scope and attachment. These are serious difficulties for any purely syntactic parsing. The solution requires the incorporation of additional semantic information either by finer subclassification than syntactic classes or by reference to the actual words used in the sentence.

For the small grammars presented here, the context sensitive rules are sufficient to resolve most of the attachment problems that occur in the testing data. But when the grammar is expanded to obtain maximum generality, they will not be resolvable without more information than is presently provided in our forty-odd syntactic classes.

7 Conclusions

We have presented a hybrid, neural net, shift/reduce parser in which the parser is a simple sequential program that manages pushdown stacks and the grammar is embodied in a trained neural network. Our approach is adequate to obtain the advantages of pattern generalization and learning-by- example provided by neural net technology. The approach enables parsing — or case analysis — of indefinitely long sentences with deeply embedded structures.

We had previously devoted many man-months to the development of symbolic, syntactic grammars for the texts considered in this paper. But specifying and training the neural network grammars for these texts required only man-weeks of effort. Since the output of our hybrid parsing system is strictly comparable to that of our previous syntactic parsing systems, we are led to believe that our neural network approach can reduce the costs of acquiring syntactic grammar without loss in efficiency of parsing.

Our emphasis in this paper has been to show that the hybrid, neural net parser is adequate to analyze English sentences of indefinite length and complexity. Our future research will concentrate on semantic analysis and translation of text in the expectation that this hybrid neural network approach will lead to more effective systems for computer understanding of natural languages.

REFERENCES

Allen, Robert, "Several Studies on Natural Language and Back Propagation", Proc. Int. Conf. on Neural Networks, San Diego, Calif., 1987.

Allen, James, *Natural Language Understanding,* Benjamin Cummings, Menlo Park, Calif., 1987.

Cook, Walter, *Case Grammar Theory,* Georgetown Univ. Press, Wash. D. C., 1989.

Cottrell, Garrison, *A Connectionist Approach to Word Disambiguation,* Morgan Kaufmann Publishers Inc., San Mateo, Calif., 1989.