Department of Computer Science
University at Albany
State University of New York

Learning Recursive Phrase Structure: Combining the Strengths of PDP and X-Bar Syntax

TR 91-5

George Berg

UNIVERSITY AT ALBANY
STATE UNIVERSITY OF NEW YORK

Learning Recursive Phrase Structure: Combining the Strengths of PDP and X-Bar Syntax*

George Berg
Computer Science Department, and
Department of Linguistics and Cognitive Science
State University of New York at Albany
Albany, NY USA
berg@cs.albany.edu

1 Introduction

One of the biggest challenges facing proponents of connectionist models of natural language processing (NLP) is the rich structure of language. As construed in most linguistic theories, a sentence consists (in part) of various types of phrases which are composed of other phrases and/or the words of the sentence. This structure also helps constrain the semantics or meaning of the sentence (e.g. what elements fill the Agent, Patient, Instrument, etc. semantic case roles; to what other element(s) in the sentence may a pronoun legitimately refer).

To be taken seriously as models of NLP, connectionist systems must either show how they can accommodate structure and structurally-sensitive processing, or propose and justify some alternative which explains the same phenomena. In the past few years, systems have emerged which address some of these issues. However, these systems have *a priori* limits on the sentence structures they support. In this paper, we introduce the XERIC Parser, a connectionist syntactic analyzer with a fixed network architecture which places no prior limit on the extent of the structure of a sentence. In practice the limits reflect the actual structures of the training data and the limitations of the network's training regimen and representation.

2 Structure in Connectionist NLP

One of the underlying tenets of the modern cognitive sciences is that human languages have a form and constraints on their meanings which are well-accounted for by assuming that syntactically and semantically they have a *compositional structure* and that this structure affects the meaning of the sentences in the language. And, as put forth most forcefully by Fodor and Pylyshyn (1988), it is difficult to capture this notion of structure and structure-sensitivity in connectionist models of NLP. Despite their use of strawmen to make their case, Fodor and Pylyshyn did indeed have a point — many early connectionist models used representations which were *flat* — there was no principled way to model the relationships between elements in their representations. And the

*This paper is to be presented at the IJCAI 91 Workshop on Natural Language Learning in Sydney Australia.

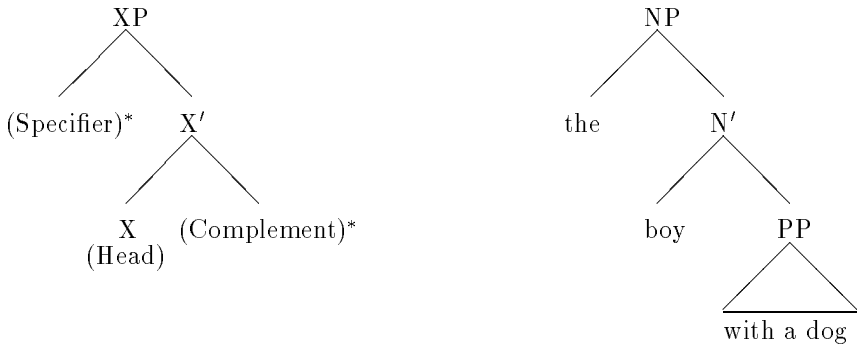

Figure 1: The X-bar template and an instantiated NP.

methods proposed in these models to account for ordering and structured relationships were either *ad hoc*, lacked systematicity, or were subject to combinatorial explosion if scaled up.

Since the time of their paper, however, several general approaches have emerged which might form the basis for connectionist models which account for the phenomena generally explained by structure (cf van Gelder, 1990). And several connectionist NLP systems have used these techniques. As representatives of the current generation of connectionist parsing research, Jain (Jain and Waibel, 1990) and Miikkulainen (1990) both use recurrent network architectures (discussed below) as the basis for their parsing systems. In both cases, the (encodings of) words are presented to the network one after another in sequence, and the recurrence in the network architecture provides a context which allows the networks to build representations for the sentences. However, both Jain and Miikkulainen constrain the types of structure which their networks accommodate. Jain's parser has a fixed limit on the number of phrase-like structures which it can represent. To represent a larger number, his network would have to be expanded by adding more of his modular representation units. In Miikkulainen's parser, the number of structures which can be represented is hard-wired into the network's architecture; to change it, the entire network would have to be altered and retrained. In fairness, both of these systems were designed to address other aspects of connectionist NLP, not the problem of the representing structure in general.

3 The XERIC Parser

In contrast, in the XERIC parser the X-Bar theory of grammar and recurrent connectionist networks are combined to produce a network with no *a priori* limit on the length of the sentence or the depth of the resulting structure. What limits there are are due to limits of the "resolution" of the fixed-length vector of units which encodes the reduced representation of the sentence structure.

In common with most other connectionist parsers, the XERIC parser, because of its recurrent, feed-forward architecture, parses in time proportional to the length of the sentence. In addition, the parser does lexical disambiguation and propagates number/person constraints in the sentence.

3.1 Underlying Concepts

In contrast to traditional theories of sentence structure which use (among other things) a set of phrase-structure rules, X-Bar grammar uses a single structure for all phrases, given in the *X-Bar template*. This template is instantiated to provide all of the structures needed, from simple Noun Phrases (NPs) to entire sentences. The

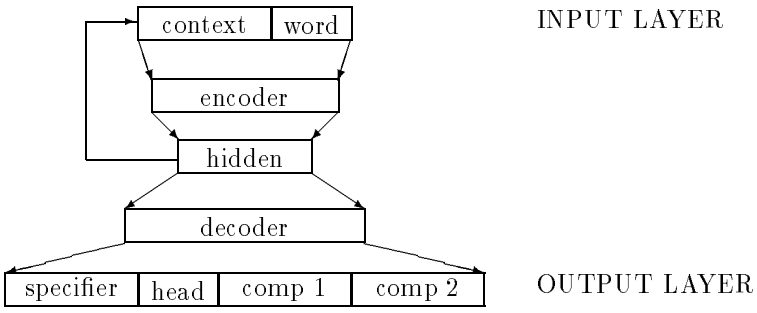

Figure 2: The Structure of the XERIC Parser Network.

exact instantiation of the template for a particular phrase is determined by the interaction of what is allowed in the phrase, as given in the lexical entry for the *head* word of the phrase, and what elements are actually provided in the sentence. For example, in Figure 1 the X-bar template indicates that a phrase may have zero or more specifiers, one head word and zero or more complements. An example instantiation for the NP "the boy with a dog" is also shown in the figure. In this phrase, the head noun, "boy" allows the determiner "the" as a specifier and the embedded Prepositional Phrase (PP) "with a dog" as a complement. This is in contrast to a NP whose head is a pronoun (e.g. "I", "them") which allows neither determiners nor PPs.

The ample information in the lexicon and the simplicity of the X-bar template provide a "grammar" which constrains the allowed structures for a particular sentence. This is more economical than phrase structure grammars, where information which must be in the lexicon is redundantly introduced in the specific phrase structure rules. And the large number of rules it takes to account for the variety of forms a particular phrase may take (e.g. Verb Phrases) is simply a manifestation of the varying requirements of the head words of the phrases.

In order to capture the sequential nature of parsing, some mechanism must be provided which allows a connectionist network to capture both the sequential "reading" of the words in the sentence, and to maintain some representation of partially read sentences. A popular technique used by, among others, Jain and Miikkulainen, is the *sequential* or *simple recurrent* network (Jordan, 1986; Elman, 1990). In a typical recurrent network, the activation levels of the hidden units at time t are presented as part of the input to the network at time $t+1$. As several researchers have shown, in many cases this gives the network a "history" or "context", so that the input at time $t+1$ can be processed in light of what inputs have been presented at earlier times.

Jordan Pollack has used a variation on the recurrent network to build connectionist network models of inherently structured representations such as stacks and queues. In his RAAM model (Pollack, 1990), he uses a $2m \times m \times 2m$ architecture to build representations of binary trees. Terminal symbols are encoded using m units. The network is trained to recreate its input on its output units. The trick is that the information necessary to properly activate the $2m$ output units is contained in the activations in the m hidden layer units. Pollack uses these values as a new, nonterminal input to subsequently encode more complex binary trees. The process can be reversed, by repeatedly feeding an encoding to the latter half of the network (the *decoder*) until terminals are found at both parts of the output.

3.2 XERIC's Network Structure

The structure of XERIC combines X-bar syntax, recurrent networks and a RAAM-style reduced encoding. As shown in Figure 2, the input to the network is an encoding of the features of the current word. The hidden layer

units' activation values from the last time step are presented as context input this time step. The activation feeds forward through the three hidden layers to the output layer. The structure of the output layer is based on a simple form of the X-Bar template used in XERIC: a phrase may have one specifier, one head word, and up to two complements.[1] The head position is the same size as the word encodings, and is used to represent the head words of the phrases (or nulls). The complements are either encodings of component phrases (or nulls). The specifier may encode either a word (e.g. a determiner such as "the"), a phrase, or null.

When parsing, the XERIC parser is started with all of the context units set to a predetermined *null* value (e.g. 0.5). The lexical encoding of a word is presented and activation feeds forward through the network. For each subsequent word in the sentence, the lexical encoding for the word is presented as input, along with the hidden unit activation values from the previous step. After the last word of the sentence has been presented as input, and activation fed forward through the network, the hidden layer units' activations will be an encoding of the representation of the entire sentence.

The structure of the sentence can be shown explicitly by using the "back half" of the network (the hidden, decoder and output layers) as a RAAM-style decoder. Feeding the activations of the encoding of a simple declarative sentence to the decoder will yield an encoding of the sentence's subject NP in the specifier position, inflection information in the head position, an encoding of the sentence's VP in complement position 1, and null values in complement position 2 (indicating that there is no element in this position). The encodings of these component phrases can then be presented to the decoder to yield their phrasal components. To get the structure of the entire sentence, simply continue this process until all of the phrases give null values for their specifier and complements. This is similar to the approach that Hinton (1988) takes in descending through a similarly-represented part/whole hierarchy.

It is the relationship between the X-Bar template and the lexical entry for the words in the sentences that makes an X-Bar approach to parsing attractive. The template is simple, and its variations are constrained by the information in the lexicon and what words are actually in the sentence. Since we are using a fixed network, and a fixed length vector of units to encode representations (the hidden layer in Figure 2), this potential economy in this method of representing structure is a critical feature of the model. Our use of a RAAM-style recursive encoding allows us to train our network to attempt to encode potentially arbitrarily-deep structures. Of course, limitations of the training methods used and on the ability of network units to encode information will put some sort of bound on what XERIC-style networks can successfully parse and represent. We hoped that these bounds would not be too tight and the the XERIC networks could parse sentences with deep and varied structure. As we indicate below, experiments show that this hope is at least partially vindicated.

3.3 Unrolled training Architecture

The basic architecture of a XERIC parsing network must be augmented in order to train it. Since one of the goals of this work is that the training of the network result in it developing an encoding scheme in order to represent the structure of the sentences and component phrases, we do not know *a priori* the representation scheme for specifiers and complements, and hence what the target activation values for the non-null phrases should be. In order to get around this problem, the network used in training is actually a number of *virtual copies* of the actual XERIC parsing network. As shown in Figure 3, using multiple copies of the network to emulate a network which completely matches the structure of each sentence allows the training to have only word encodings and nulls at the output layers. The training regimen is then free to determine phrase encodings at the (replicated) hidden layers. By slaving all of the virtual copies of the network to the actual one so that all weight changes due to learning are applied to the one actual XERIC network, we retain the function of the back half of the XERIC network as the general-purpose phrase decoder described above.[2]

[1] This form of X-bar grammar is admittedly oversimplified (cf Fukui and Speas, 1986). However, it is adequate for the purposes of XERIC, and there is no principled reason why a more general template which accommodates the varying number of specifiers or complements couldn't be used.

[2] This technique is similar in form, if not in purpose, to "back-propagation through time" (Rumelhart et al., 1986).

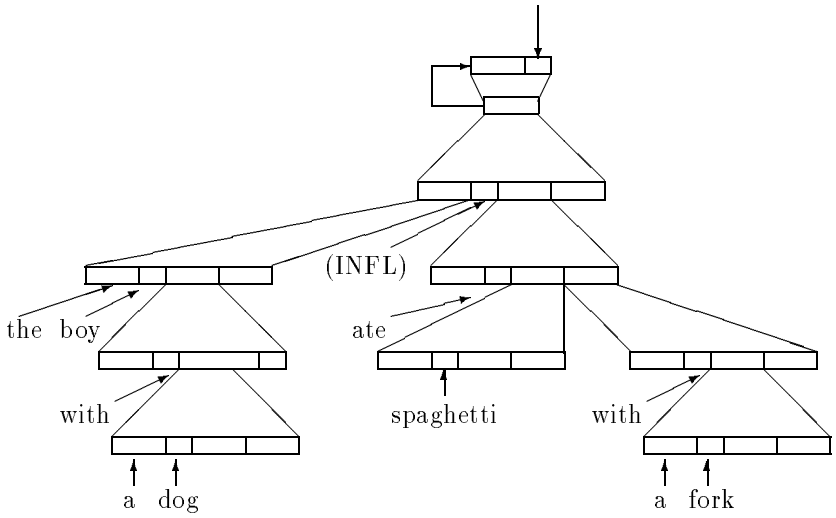

Figure 3: The "Unrolled" network for the sentence "the boy with a dog ate spaghetti with a fork.". Unlabeled phrases at the output level are empty.

3.4 Other Features

In many cases, a word in isolation is lexically ambiguous. For instance, the word "drive" may either be a noun or a verb. In XERIC networks, such words are presented as input in an ambiguous form. One of the tasks the network performs is to disambiguate such words. Depending on where in the sentence an ambiguous word is read, it is disambiguated to its proper form.

In addition, the XERIC network also propagates number and person agreement between the subject noun phrase and the sentence's main verb. As the head noun and the main verb are read, they constrain each other as to what number and person values they may legitimately have (e.g. "the sheep are" constrains sheep to be third-person plural, whereas "the sheep is" must be third-person singular).

There is also one additional feature that the XERIC parser shares with many other of the connectionist parsers. Since this is a fixed network feed-forward architecture, the time to parse a sentence is linear in the number of words in the sentence.

4 Training XERIC Networks

In its current form, phrases are allotted 45 units. This is, of course, also the size of the hidden layer and the context part of the input layer.[3] A lexical entry (used at the word part of the input and for phrase heads at the output layer) is encoded using 35 units.[4] In general, individual units in word encodings represent the presence

[3] The encoder layer is currently 62 units and the decoder layer 108 units.

[4] When the element in a specifier is a word (e.g. a determiner), the encoding only uses 35 units, the others being set to a null value. Somewhat redundantly, the current XERIC architecture has a unit which serves as a "mode bit" for the specifier — its value indicates whether the specifier contains a word or a phrase (or null).

or absence of certain linguistic features in that word. Among the features are a set which gives the syntactic category of the word (e.g. noun, verb, preposition), its number/person (e.g. first-person plural, third-person singular), as well as various features specific to syntactic categories of words. For nouns, these include whether or not a noun is a pronoun, an anaphor, a proper-noun, etc. For verbs the features include the tense of the verb and whether it takes a direct and/or indirect object. If a feature is present, the corresponding unit has a value of one, a value of zero if it is absent, and a value of 0.5 if it is either optional or unclear. An example of an optional feature is a verb that may optionally take a direct object. An example of an unclear feature would be the features "noun" and "verb" in the encoding of a word where its category is initially ambiguous. The lexical encoding for each word also contains a 9-unit "ID code" which uniquely distinguishes each word, apart from its syntactic features.

One simplification used in the current lexicon is that all verb-subcategorization features are fixed at either one or zero. As explained below, since XERIC currently does not have the semantic information necessary to resolve ambiguous PP-attachments (e.g. whether a PP is an indirect object of the verb or whether it is a complement of the directly-preceding NP).

Testing and training are done on separate corpora each containing the patterns for 1000 sentences. The average sentence is between 6.5 to 7 words long, resulting in corpora containing between 6500 and 7000 patterns. The corpora contain sequential presentations of randomly generated, syntactically legal sentences. There is no restriction on the length or "depth" of the generated sentences, although the random selection is biased slightly against complex phrases. This results in most of the sentences being between 2 and 8 phrases deep, with fewer sentences of greater depths, typically with a maximum between 14 and 20.

Since the lexical entries do not contain sufficient semantic information to do PP-attachment disambiguation, the generator places a restriction on the sentence forms for verbs which subcategorize for both a direct and an indirect object. In these cases, the direct object NP (in the VP's comp1 position) will be *simple* — it will contain no embedded phrases. This way there is no phrase attachment ambiguity. The first NP after the verb will be the verb's direct object, and the following PP (and any subsequent phrases) will be part of the verb's indirect object. This is in contrast to the general case where the direct object may be a NP with embedded component phrases. Without semantic information it is impossible to tell in general whether following PPs belong to the direct object NP or are the beginning of the verb's indirect object.

In the corpora, the target outputs for each pattern are the "unrolled" network outputs (cf Figure 3) of as much of the information in the sentence as can be determined from the words read so far (along with whatever disambiguation can be inferred from the partial sentence).

The training method used for the XERIC parser is error backpropagation (Rumelhart et al., 1986). Weight and bias updates are done after every pattern presentation. Investigation has shown that the networks converge best with low learning rate values (typically using $0.01 \leq \eta \leq 0.05$) and using no momentum term (i.e. $\alpha = 0$). We are also investigating other methods of training the networks, such as Fahlman's quickprop (Fahlman, 1988); however, our early results tend to confirm Fahlman's report of the instabilty of quickprop and related methods on recurrent networks (Fahlman, 1991).

The actual simulation programs for training and running XERIC networks are coded in the C language, and run on SUN SPARCstation 1 workstations. On a SPARCstation, XERIC networks train on the 1000-sentence corpora at about 1.5 hours per epoch. This relatively slow rate of speed is because of the large size of the corpora, the depth of the virtual networks and the relative complexity of the software necessary to implement the virtual network training.

5 Results

XERIC networks typically converge to 1–4% error (where an error is a unit which deviates from its target value by a certain amount, e.g. 0.2). It typically takes a network 500–1500 epochs to reach a rough asymptote for error values. XERIC trains to correctly encode *the structure* of all but the most atypical or extremely deep sentences it parses. Most of the errors are lexical in nature — incorrect word ID codes or syntactic features. These errors increase as more of the sentence is read. They also increase with the depth of the nesting in a sentence's structure. Testing on corpora other than the one on which a network was trained shows only minor increases in error percentages for sentences of depth up through ten.

These results suggest that the reduced description of the sentence, which is the heart of this architecture, is an "information bottleneck". Intuitively, either the limitations of the learning algorithm, or the inability of a finite floating-point simulation of a fixed-size network to represent more than a certain amount of information causes this to be harder for the network to store and reproduce. We are currently analyzing this.

6 Future Work

Despite their ability to represent complicated and varied sentence structure, the current generation of XERIC networks are not general parsing models. They are very specifically targeted at exploring the need to represent the syntactic structure of sentences. In one sense, these networks are only a prerequisite for connectionist NLP systems. We do however plan to use them as the basis for more complex parsers.

Currently the sentences XERIC parses are all active-form, declarative sentences. Using the existing network architecture the corpora can be extended to include passive form sentences, questions, imperative sentences and sentences which themselves contain embedded sentences. There is one addition, which although minor, will require restructuring the networks — adjectives. Using the present network architecture to handle NPs with adjectives would require the addition of some *ad hoc* phrase structures. There are better ways to handle adjectives, but they require a different underlying model of X-Bar grammar (cf Fukui and Speas, 1986). Also, there is the possibility that an appropriate extension to X-Bar grammar may in some cases allow the network to encode alternative structures where there is syntactic ambiguity.

Even with a more general theory of X-Bar grammar, the XERIC system would not be able to completely parse complicated sentences. A complete parsing system of this sort would use X-Bar grammar as only one of a set of systems. These systems would act as constraints or filters to determine the correct syntactic representation of the sentence (Berwick and Fong, 1990; Rager and Berg, 1990).

In particular, the set of constraints usually associated with *Government-Binding* (GB) Theory (Chomsky, 1981) can be used. One avenue which we are exploring is using Rager and Berg's (1990) connectionist implementation of motion and government in GB to provide constraints on the sentence forms generated by the XERIC parser.

Even if augmented with the full armamentorium of GB constraints, a XERIC parser could not handle phenomena such as PP-attachment, which depend on semantic knowledge. To do so requires that the lexical items for words include not only syntactic information, but semantic information as well. While in principle possible, it plays to a weakness of the XERIC architecture — one of the things it is worst at is preserving the features of individual words. Greatly expanding the size of the lexical entries may only further reduce the networks' ability to store and recover the lexical entries for the words in the sentences it reads. This is important because a parser which cannot faithfully represent the sentences it reads is of doubtful value.

Anticipating this difficulty, one extension to XERIC we are investigating is circumventing the need to store

all of the lexical information from the words in the sentence in the XERIC network. We are experimenting with architectures for storing the input forms of the words in the sentence in a separate network, which can be used by the XERIC network to recover the full (disambiguated) lexical entries in decoding.

7 Conclusion

By using a sparse method of representing structure and a network architecture which, in principle, can encode arbitrarily complex structures the XERIC parsing networks provide a model of connectionist NLP which can support the rich and varied structure of sentences in human language. Preliminary experiments show that such networks can encode in a finite network enough information to represent complex sentences with fairly deep recursive phrase structures. In addition a XERIC parser can also do lexical disambiguation and number/person constraints as it parses. If the XERIC model can be augmented to handle additional syntactic principles and semantic representation and processing, then it will represent an important step forward in connectionist natural language processing.

Acknowledgments

I would like to thank Judith Swota, John Rager, Andy Haas, Carl Edlund, Doug Mokry and Jordan Pollack for their insights on this work and related matters. The Computer Science Department at Rensselaer Polytechnic Institute (and in particular, Chris Welty) provided additional workstations which helped make this work possible.

References

Berwick, R. C. and Fong, S. (1990). Principle-based parsing: Natural language processing for the 1990s. In Winston, P. H. and Shellard, S. A., editors, *Artificial Intelligence at MIT: Expanding Frontiers, Vol. 1*. MIT Press, Cambridge, MA.

Chomsky, N. (1981). *Lectures on Government and Binding*. Foris, Dordrecht.

Elman, J. L. (1990). Finding structure in time. *Cognitive Science*, 14:179–212.

Fahlman, S. E. (1988). Faster-learning variations on back-propagation: An empirical study. In *Proceedings of 1988 Connectionist Models Summer School*. Morgan Kaufmann.

Fahlman, S. E. (1991). The recurrent cascade-correlation architecture. Technical Report CMU-CS-91-100, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA.

Fodor, J. A. and Pylyshyn, Z. W. (1988). Connectionism and cognitive architecture: A critical analysis. In Pinker, S. and Mehler, J., editors, *Connections and Symbols*. MIT Press, Cambridge, MA.

Fukui, N. and Speas, P. (1986). A theory of category projection and its applications. In Fukui, N., Rapoport, T. R., and Sagey, E., editors, *MIT Working Papers in Linguistics, Vol. 8*, pages 128–172. MIT Department of Linguistics and Philosophy.

Hinton, G. (1988). Representing part-whole hierarchies in connectionist networks. In *Proceedings of the Tenth Annual Conference of the Cognitive Science Society*, pages 48–54, Montreal, Canada.

Jain, A. N. and Waibel, A. H. (1990). Incremental parsing by modular recurrent connectionist networks. In Touretzky, D. S., editor, *Advances in Neural Information Processing Systems 2*. Morgan Kaufmann, San Mateo, CA.

Jordan, M. I. (1986). Attractor dynamics and parallelism in a connectionist sequential machine. In *Proceedings of the Eighth Annual Conference of the Cognitive Science Society*, pages 531–546, Amherst, MA.

Miikkulainen, R. (1990). A PDP architecture for processing sentences with relative clauses. In *Proceedings of the 13th International Conference on Computational Linguistics*, Helsinki.

Pollack, J. B. (1990). Recursive distributed representations. *Artificial Intelligence*, 46:77–105.

Rager, J. and Berg, G. (1990). A connectionist model of motion and government in Chomsky's government-binding theory. *Connection Science*, 2(1 & 2):35–52.

Rumelhart, D. E., Hinton, G., and Williams, R. J. (1986). Learning internal representations by error propagation. In McClelland, J. L., Rumelhart, D. E., and the PDP Research Group, editors, *Parallel Distributed Processing: Volume 1: Foundations*. MIT Press, Cambridge, MA.

van Gelder, T. (1990). Compositionality: A connectionist variation on a classical theme. *Cognitive Science*, 14:355–384.