A Connectionist Parser with Recursive Sentence Structure and Lexical Disambiguation*

George Berg
Computer Science Department
State University of New York at Albany, LI-67A
Albany, NY 12222 USA

Abstract

In order to be taken seriously, connectionist natural language processing systems must be able to parse syntactically complex sentences. Current connectionist parsers either ignore structure or impose prior restrictions on the structural complexity of the sentences they can process — either number of phrases or the "depth" of the sentence structure. XERIC networks, presented here, are distributed representation connectionist parsers which can analyze and represent syntactically varied sentences, including ones with recursive phrase structure constructs. No *a priori* limits are placed on the depth or length of sentences by the architecture. XERIC networks use recurrent networks to read words one at a time. RAAM-style reduced descriptions and X-Bar grammar are used to make an economical syntactic representation scheme. This is combined with a training technique which allows XERIC to use multiple, virtual copies of its RAAM decoder network to learn to parse and represent sentence structure using gradient-descent methods. XERIC networks also perform number-person disambiguation and lexical disambiguation. Results show that the networks train to a few percent error for sentences up to a phrase-nesting depth of ten or more and that this performance generalizes well.

Introduction

One of the biggest challenges facing proponents of connectionist models of natural language processing (NLP) is the rich structure of language. In most linguistic theories, a sentence consists (in part) of various types of phrases which are themselves composed of other phrases and/or words. 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 use structure and structurally-sensitive processing, or demonstrate an alternative explanation of the same phenomena. Recently, researchers have addressed some of these issues. However, their systems have *a priori* limits on the sentence structures they support. In this paper, we introduce the XERIC Parser, a connectionist syntactic analyzer which places no prior limit on the extent of the structure of a sentence. In practice its limitations seemingly reflect the actual structures of the training data and the limitations of the network's training regimen and finite representation.

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 methods proposed in these models to account for ordering and structured relationships either were *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 for connectionist models to account for structure (cf van Gelder, 1990). And several connectionist NLP systems have used these techniques. Miikkulainen (1990) uses recurrent network architectures (discussed below) as the basis for his parsing system. Jain [Jain and Waibel, 1990] uses gating networks to similar effect. In both cases, encod-

*This is a reprint of a paper from pp. 32–37 of the Proceedings of AAAI-92.

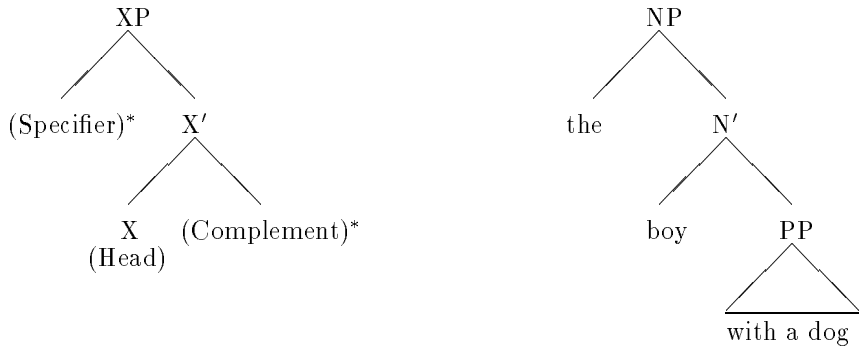

Figure 1: The X-bar template and an instantiated NP.

ings of words are presented to the network one after another in sequence, and the recurrence or gating 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.

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 addition, the parser does lexical disambiguation and propagates the number/person constraints in the sentence. And, 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.

Underlying Concepts

In contrast to the phrase-structure rules of traditional theories of sentence structure, X-Bar grammar uses a single structure for all phrases — the *X-Bar template* [Sells, 1985]. This template is instantiated to provide all of the structures needed, from simple Noun Phrases (NPs) to entire sentences. The template's instantiation 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. One technique (used by 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 $t+1$. 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 [Cleeremans *et al.*, 1989].

Jordan Pollack has used a variation on the recurrent

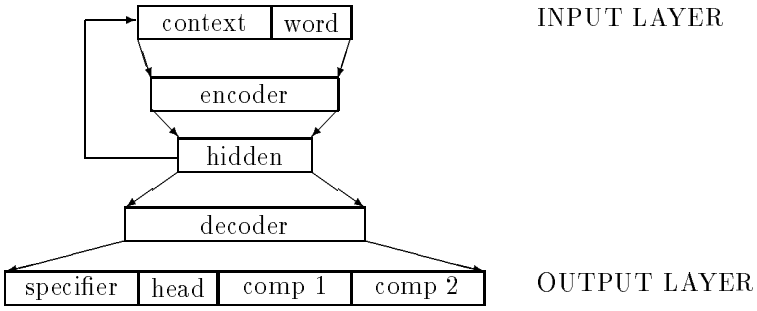

Figure 2: The basic structure of the XERIC parser network.

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 m hidden units encode the information to recreate the activations of the $2m$ output units. The hidden unit values can then be used as new, nonterminal inputs 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. From an initial vector of m units, this process can retrieve the tree structure that it encodes.

XERIC's Basic Network Structure

XERIC combines X-bar syntax, recurrent networks and a RAAM-style reduced encoding. The basic architecture is 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 word of a phrase (or null). 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 are 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, sentence 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 component phrases can be presented to the decoder to give 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), the potential economy of 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 en-

[1] This form of X-bar grammar is admittedly oversimplified (cf Fukui and Speas, 1986). However, it is adequate for XERIC, and can in principle be generalized.

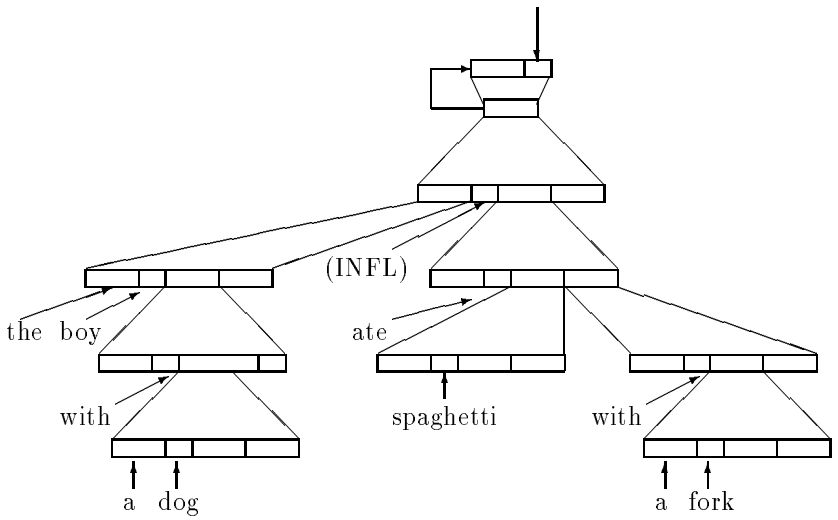

Figure 3: The "unrolled" network for the sentence "the boy with a dog ate spaghetti with a fork.". Unlabeled phrases at the output level are empty.

code 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 that the XERIC networks could parse sentences with deep and varied structure. As we indicate below, experiments show that this hope is vindicated.

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 decoder portion 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]

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).

The size of phrases is 45 units. Because of the recurrent mappings this is also the size of the hidden layer and its recurrent mapping at 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] Most of the units in word encodings represent

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

[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 deter-

linguistic features. Typical features are the syntactic category of the word (e.g. noun, verb, preposition), its number/person (e.g. first-person plural, third-person singular), and various category-specific features. 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, whether it takes a direct and/or indirect object, etc. The unit has a value of one if a feature is present, 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.

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 semantic information to do PP-attachment disambiguation, the generator restricts 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.

Training XERIC Networks

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).

miner), the encoding only uses 35 units, the others being set to a null value. XERIC has a "mode bit" unit to indicate whether the specifier contains a word, a phrase or is null.

The training method used for the XERIC parser is error backpropagation [Rumelhart *et al.*, 1986]. Weight 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. $\alpha = 0$).

The simulations for training and running XERIC are written in C and run on SUN SPARCstation 1 workstations. XERIC networks train at about 1.5 hours per epoch. This relatively slow rate of speed is due to 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.

Results and Future Work

XERIC networks typically converge to 1–8% overall error for both training and testing corpora (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 networks train 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.

For the best XERIC networks, the overall error at the end of a sentence varies between 1% to 9% as the nesting depth of the sentence increases from 2 to 10 in the training corpus. For the same network a typical testing corpus will have error rates of from 1% to 11% in this range. The error rates increase roughly in proportion to the depth in both cases. Beyond a depth of 10, error rates vary greatly from 5% to 18%, possibly reflecting memorization of the small number of exemplars or limits to the ability of the networks to represent these very deep sentences.

The 18 units at each phrase output layer which represent the word IDs account for 52.7% of the errors in the training corpus and approximately 50% for testing corpora. Since these units account for only 10.5% of the total number of output units, a disproportionate amount of the error of XERIC networks is due to the network's inability to correctly reproduce the IDs of the words it reads.

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.

Despite their ability to represent complicated and varied sentence structure, the current generation of XERIC networks are not general NLP models. They are very specifically designed to explore 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 and analyzers.

We are currently working on several approaches to improve both the performance and breadth of coverage of XERIC parsers. Using a more general X-Bar template will allow us to incorporate embedded and passive sentences as well as adjective phrases. Other elements of linguistic theory (cf Government-Binding Theory [Chomsky, 1981]) can be used by adding more networks to constrain the sentence's representation [Rager and Berg, 1990].

We are also adding semantic analysis to the XERIC parser. This is necessary even if one is only interested in syntactic structure of a sentence, because PP-attachment is, at least in part, semantically motivated. To be part of a useful NLP system, XERIC parsers must have a semantics or at least be compatible with a separate semantic analyzer.

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.

Acknowledgements

An early version of this work was presented at the IJCAI 1991 Workshop on Natural Language Learning. I would like to thank John Munson, Judith Swota, John Rager, Andy Haas, Carl Edlund, Doug Mokry and Jordan Pollack for insights related to this work. The Computer Science Department at Rensselaer Polytechnic Institute (and in particular, Chris Welty) provided additional workstations to help make this research possible.

References

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

Cleeremans, A.; Servan-Schreiber, D.; and McClelland, J. 1989. Finite state automata and simple recurrent networks. *Neural Computation* 1:372–381.

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

Fodor, J. A. and Pylyshyn, Z. 1988. Connectionism and cognitive architecture: A critical analysis. In Pinker, S. and Mehler, J., editors 1988, *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.; and Sagey, E., editors 1986, *MIT Working Papers in Linguistics, Vol. 8.* MIT Department of Linguistics and Philosophy. 128–172.

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

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

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

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

Sells, P. 1985. *Lectures on Contemporary Syntactic Theories.* Center for the Study of Language and Information, Stanford, CA.

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