Workshop ROMAND2000, Lausanne, 19-20 octobre 2000

Estimating a Probabilistic Grammar Using a Neural Network

James Henderson

University of Exeter
School of Engineering and Computer Science
Exeter EX4 4PT, UK

Abstract

Previous work has demonstrated the viability of a particular neural network architecture, Simple Synchrony Networks (SSNs), for syntactic parsing (Henderson & Lane, 1998), (Henderson, 2000). However the output was interpreted as a score, only interpretable in a heuristic way relative to other scores of the same type. In this paper we discuss a SSN parser trained using a method that allows us to interpret the output as probability estimates. These estimates are then used in a chart parser to find the most probable complete parse. This system compares favorably to one where the probability estimates are calculated from head bigram counts, and to a simple Probabilistic Context Free Grammar. We argue that this is due to the weaker independence assumptions necessary for training the SSN relative to statistical models based on frequency counts. If so, then such neural network architectures are a potential alternative to statistical smoothing techniques for developing robust statistical parsers.

1. Introduction

In many domains neural networks are an effective alternative to frequency based statistical methods. This has not been the case for syntactic parsing, but recent work has identified a viable neural network architecture for this problem (SSN) (Henderson & Lane, 1998), (Lane & Henderson, 1998), and this parsing method appears to be robust in the face of sparse training data (Henderson, 2000). However the lack of a probabilistic interpretation for the parsers' outputs makes it difficult to integrate such a parser with other robust NLP systems. This paper proposes a new version of the SSN parser whose outputs can be interpreted as probability estimates of structural relationships (i.e. of attachment decisions). We compare this technique to estimating these probabilities using frequency counts. When used in a chart parser to find the most probable parse, the SSN achieves much better performance than the frequency based method. This SSN parsing system also performs better than a simple Probabilistic Context Free Grammar (PCFG), used to establish a baseline.

We argue that the reason that the neural network parser performs better than its frequency based counterpart is that it can make weaker independence assumptions. In order to compensate for the necessarily limited amount of data available, methods which estimate probabilities by counting frequencies must make strong independence assumptions so as to have sufficient

counts. These assumptions lead to undesirable biases in the model generated, and may still not guarantee coverage of less frequent cases. Neural networks also require independence assumptions in order to define their input-output format, but these assumptions can be much weaker. Because neural networks learn their own internal representations, neural networks can decide automatically what features to count and how reliable they are for predicting the output. Thus they provide a potential alternative to using smoothing techniques to find estimates for infrequent cases, as is required for robust parsers.

2. The Probabilistic Grammar Model

This section presents the probabilistic model which is shared between the SSN and its frequency based counterpart. The following sections discuss a few additional assumptions made to suit each individual technique. The shared model is based on the probabilities of individual structural relationships between syntactic constituents, plus labels for the nonterminal constituents. The labels specify things like whether the constituent is a noun phrase or a verb phrase. Structural relationships specify things like the attachment of a prepositional phrase to a noun phrase, or the inclusion of a noun into a noun phrase. By specifying all such structural relationships in a constituent structure tree we can fully specify the constituent structure tree.

Two structural relationships are sufficient to fully specify the structures in the corpus discussed in section 5, *parent*, and *grandparent*. *Parent* is the relationship between a word[1] and the constituent that immediately dominates it in the tree. *Grandparent* is the relationship between a word and the constituent which immediately dominates the word's parent constituent. For the corpus used in this paper these *grandparent* relationships are sufficient to specify all immediate dominance relationships between two nonterminal constituents, because all nonterminal constituents have at least one word of which they are the *parent*. To prevent redundancy, we only specify the *grandparent* relationship for the first such child word for each constituent. For example the structure of a simple subject-verb sentence would be represented as

$$\textit{parent}(c_1, w_1), \textit{parent}(c_2, w_2), N(c_1), S(c_2), \textit{grandparent}(c_2, w_1)$$

where c_1, c_2 are nonterminals, w_1, w_2 are the words (such as "John loves"), and N, S are labels. This is the structure depicted in figure 1 after the second word under "Accumulated Output".

To calculate an estimate for the probability of a given parse, we decompose the estimate into one estimate for each of the labels or structural relationships in the parse. To do this we first apply the chain rule in a bottom-up, left-to-right order. For example to compute the probability of the structure mentioned above:

$$\begin{aligned}
&P(\textit{parent}(c_1, w_1), \textit{parent}(c_2, w_2), N(c_1), S(c_2), \textit{grandparent}(c_2, w_1) \mid w_1, w_2) \\
&= P(\textit{parent}(c_1, w_1) \mid w_1, w_2) \\
&\quad * P(\textit{parent}(c_2, w_2) \mid \textit{parent}(c_1, w_1), w_1, w_2) \\
&\quad * P(N(c_1) \mid \textit{parent}(c_2, w_2), \textit{parent}(c_1, w_1), w_1, w_2) \\
&\quad * P(S(c_2) \mid N(c_1), \textit{parent}(c_2, w_2), \textit{parent}(c_1, w_1), w_1, w_2) \\
&\quad * P(\textit{grandparent}(c_2, w_1) \mid S(c_2), N(c_1), \textit{parent}(c_2, w_2), \textit{parent}(c_1, w_1), w_1, w_2)
\end{aligned}$$

Then we apply the independence assumptions of the grammar model to get the specific probabilities which need to be estimated, as discussed next.

[1] In the actual experiments below we use part-of-speech tags as inputs, not the actual words. We are using "words" here for expository convenience.

The first assumption is that labeling nonterminals is sufficiently simple (for our dataset) that knowing a label doesn't tell us anything the we don't already know from the part-of-speech tags for the words. This allows us to determine labels and structural relationships independently of each other. The second assumption is that nonterminals can be labeled incrementally, as soon as we see the first word which is an immediate child of the constituent. Similarly, the model assumes that the probability of a structural relationship can be estimated as soon as both the word involved in the relationship and the first word-child of the nonterminal involved in the relationship have been seen. Later words are assumed to be independent of these labels and relationships. So far we have reduced the calculation for the above example to the following:

$$
\begin{aligned}
& P(\textit{parent}(c_1, w_1) \mid w_1) \\
& * P(\textit{parent}(c_2, w_2) \mid \textit{parent}(c_1, w_1), w_1, w_2) \\
& * P(N(c_1) \mid \textit{parent}(c_1, w_1), w_1) \\
& * P(S(c_2) \mid \textit{parent}(c_2, w_2), \textit{parent}(c_1, w_1), w_1, w_2) \\
& * P(\textit{grandparent}(c_2, w_1) \mid \textit{parent}(c_2, w_2), \textit{parent}(c_1, w_1), w_1, w_2)
\end{aligned}
$$

The idea behind the remaining independence assumptions is that the first word which is an immediate child of a constituent is a good approximation to the syntactic head of the constituent, and as such determines most of the characteristics of the constituent. Except for these head relationships, we assume that structural relationships are independent of each other. We also assume that all relationships for constituents not involved in the relationship in question are not important. Specifically, the probability of a parent relationship is independent of all other structural relationships, except the head relationship for the parent constituent.[2] The probability of a grandparent relationship is independent of all other structural relationships, except the head relationship for the grandparent constituent and the fact that the grandchild word must be the head of its parent constituent. These are somewhat strict independence assumptions, but note that we are not ignoring the prior words, so to the extent that the prior parse is unambiguous these independence assumptions are inconsequential.

Given this full set of independence assumptions, the calculation for the above example is reduced to the following:

$$
\begin{aligned}
& P(\textit{parent}(c_1, w_1) \mid w_1) \\
& * P(\textit{parent}(c_2, w_2) \mid w_1, w_2) \\
& * P(N(c_1) \mid \textit{parent}(c_1, w_1), w_1) \\
& * P(S(c_2) \mid \textit{parent}(c_2, w_2), w_1, w_2) \\
& * P(\textit{grandparent}(c_2, w_1) \mid \textit{parent}(c_2, w_2), \textit{parent}(c_1, w_1), w_1, w_2)
\end{aligned}
$$

These are the probabilities which need to be estimated by the neural network and its frequency based counterpart. Given all such probabilities, a best first chart parser is used to find the most probable constituent structure for the sentence, in a similar way as for PCFGs.

3. Estimating the Probabilities with a Simple Synchrony Network

We use the same neural network architecture (Lane & Henderson, 1998) and the same parser design (Henderson, 2000) as has been used in previous work. The neural network architecture

[2] This assumption includes the incorrect assumption that choosing one nonterminal as the parent of a word is independent of choosing another as the parent, when in fact they are incompatible because a word can only have one parent. We are currently investigating a slight modification to the neural network design which should correct this inaccuracy.

(SSNs) is appropriate for syntactic parsing because of its ability to learn generalizations over syntactic constituents (Henderson & Lane, 1998). We will first review the parser design and then discuss the probabilistic interpretation of the outputs.

3.1. Outputting the Relationships

The input-output format of the SSN parser is designed to allow us to directly interpret it as estimates of the probabilities discussed in the previous section. The main challenge in defining such an input-output format is that there are $O(n^2)$ probabilities for all the possible structural relationships,[3] but if we wait until the end of the sentence then the SSN architecture can only produce $O(n)$ outputs. The SSN architecture only produces a bounded number of outputs per constituent at any given time, and only a bounded number of constituents can be introduced in each time step, so at the end of the sentence there are a linear number of constituents and therefore a linear number of outputs. The solution is not to wait until the end of the sentence, but to produce outputs during each of the $O(n)$ time steps during the course of the parse. $O(n)$ time steps times the $O(n)$ outputs per time step gives us the necessary $O(n^2)$ total outputs. In other words, we need to output the structural relationships incrementally.

The particular SSN parser used here outputs (the probabilities for) the structural relationships maximally incrementally. Words are input to the parser one at a time, and with each word a new constituent is introduced to act as the constituent which the word heads, if needed. Below we will use c_i to denote the constituent which is introduced during the input of word w_i. A constituent c_i forms a part of the final parse if and only if *parent*(c_i, w_i), in which case w_i is presumed to be the syntactic head of c_i. A structural relationship is output as soon as both the word in the relationship has been input and the constituent in the relationship has been introduced. This incremental output is illustrated in figure 1. When the parse is complete the accumulation of all the outputs fully specifies the parse, as illustrated at the bottom of figure 1.

The output of a neural network is the activation values calculated by the network's output units. For this parser the output units are divided into three parts, one for producing estimates of *parent* probabilities, one for producing estimates of *grandparent* probabilities, and one for producing estimates of label probabilities. The *parent* probabilities can be output with a single unit, which calculates a different output value for each constituent during each time step. The *parent* output for constituent c_i during the time step in which word w_j is input is the estimate of $P($*parent*$(c_i, w_j) \mid$ *parent*$(c_i, w_i), w_1, \ldots, w_j)$, or if $i = j$ it is the estimate of $P($*parent*$(c_i, w_i) \mid w_1, \ldots, w_i)$. In figure 1 the later case applies to the desired *parent* output for NP, VVZ, and AT, and the former case applies to the desired *parent* output for NN.

The part of the output which estimates the probabilities for the *grandparent* relationships consists of two output units, one for right branching cases and one for left branching cases. Both these output units also calculate a different output value for each constituent during each time step. The *left-grandparent* output for constituent c_i during time step w_j is the estimate of $P($*grandparent*$(c_i, w_j) \mid$ *parent*$(c_i, w_i),$ *parent*$(c_j, w_j), w_1, \ldots, w_j)$ (as for c_2 and AT in figure 1). The *sibling* output for constituent c_i during time step w_j is the estimate of $P($*grandparent*$(c_j, w_i) \mid$ *parent*$(c_i, w_i),$ *parent*$(c_j, w_j), w_1, \ldots, w_j)$ (as for c_1 and VVZ in figure 1).

In addition to these structural outputs, there is also one output unit for each possible non-

[3] Here we are assuming that the number of constituents is linear in the number of words. This is uncontroversial, being implied by any lexicalized or dependency-based grammar.

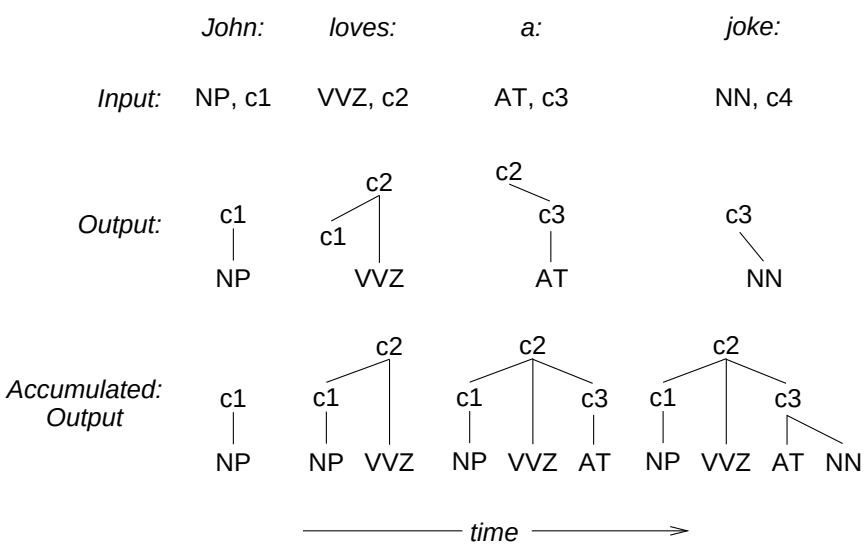

Figure 1: An example of the SSN parser's output, depicted graphically.

terminal label. For simplicity these outputs are also incremental, but only one output value is calculated in each time step. The output for label L_k during time step w_i is the estimate of $P(L_k(c_i) \mid \textit{parent}(c_i, w_i), w_1, \ldots, w_i)$.

3.2. The Soft Biases

Given enough data and enough training time, the network just described can accurately estimate the probabilities in the grammar model. However, given that we always have limited amounts of data it is important to bias the network towards solutions which we have some prior reason to believe will be good. We can do this with the SSN parser by adding inputs which will emphasize the importance of certain information, plus making use of the timing of the computation.

Because all the words are input to the representation of each constituent, in theory any earlier input can affect a later output, and thus they are not being assumed to be independent. However, in practice a recurrent network such as an SSN will learn more about the dependencies between an input and an output if they are close together in time. The immediate effect of this is a form of recency bias; the most recent input words will effect an output more than earlier input words. We can also make use of this property by adding input units which express information that the network should already have, but that may need emphasizing at different times during the parse. For example we bias the network to pay particular attention to the head word for each constituent by providing the head word as an input at every time during the life of a constituent. So the input for constituent c_i during time step w_j includes w_i as the constituent's head, as well as w_j as the current input word. This results in the network paying particular attention to the words that are directly related to the outputs being produced for constituent c_i during time step w_j, with previous input words providing an influence proportional to their recency.

We also bias the network training by providing a *new constituent* input unit which is only active for c_i at time w_i. This helps the *parent* output unit distinguish between making w_i the head of a new constituent (c_i) and attaching w_i to an old constituent. In addition this input helps the network produce other outputs because the short period after this input is when most of the outputs involving c_i will be required. For similar reasons we provide a *last parent* input unit, which is the disambiguated *parent* output from the previous input word. This input information

is also correlated with nonzero outputs being required for a short time afterwards. These additional inputs have been devised in part on the basis of the author's linguistic knowledge and in part on the basis of experimental results.

3.3. The Training Method

The training method we use is an extension of Backpropagation Through Time (Lane & Henderson, 1998), but we use a different output error function, called Cross Entropy. This allows us to train the outputs using 0/1 targets, and have the network training converge to an output which is an estimate of the probability that the output should be a 1 given the input, assuming that all the outputs are independent. By using a 0 target when a label or relationship is not true and a 1 target when it is true, the network can be trained to estimate the probabilities of labels, *parent* relationships, and *grandparent* relationships, given the words which have been input so far. This covers all the probabilities in the grammar model, except that most of them are also conditioned on some *parent* relationships. To achieve this we use a technique based on mixture models (Bishop, 1995), where the *parent* outputs are used as the mixing coefficients. The effect of this technique is simply that we only train outputs in cases where the true *parent* relationships are compatible with the condition of the probability. For example in the case of NN in figure 1, the *left-grandparent* output unit and the *sibling* output unit would not be trained.

4. Estimating the Probabilities from Frequencies

The standard approach to estimating the probabilities in section 2 would be to make strong enough independence assumptions that the necessary parameters can be estimated from frequency counts. We will call this approach to estimating these probabilities the Probabilistic Structural Relationships (PSR) model. In addition to all the independence assumptions discussed in section 2, the PSR model assumes every structural relationship is dependent on the word involved in the relationship and the head word of the constituent involved in the relationship (and whether they are the same), but they are independent of all other words. This independence assumption is strong enough to provide us with sufficient statistics given our training data, but still captures to the extent possible the relevant information for estimating the structural relationship probabilities. Given this assumption, all the parameters of the above model can be estimated by counting the pairs of the words and head words involved in each relationships.

This choice for the independence assumptions of the PSR model is designed to provide a minimal pair with the SSN model. First of all, the PSR model's independence assumptions subsume all the independence assumptions made to define the SSN's input-output format, which were discussed in section 2. The only difference between the independence assumptions for the two models is that the PSR's assumptions impose hard constraints in the cases where the SSN model imposes soft constraints, namely the recency preferences discussed in section 3.2.

5. The Experiments

In this section we empirically test the ability of Simple Synchrony Networks to estimate the probabilities required by the probabilistic grammar model. We compare its performance both to its frequency based counterpart (the PSR model) and to a simple PCFG. To demonstrate the advantages of using an estimation method that does not require strong independence assump-

tions, we do these tests using a relatively small training set.[4] To handle the resulting sparseness of training data the network must learn which of the inputs about a constituent are important, as well as how they correlate with the outputs. The advantage of SSNs is that they automatically learn the relative importance of different inputs as part of their training process.

5.1. A Corpus

We use a subset of the SUSANNE corpus (Sampson, 1995) as a source of preparsed sentences for our experiments. The language is English. We removed annotations which are not relevant to specifying syntactic constituency and we used the shorter version of the part-of-speech tags (the Lancaster-Leeds Treebank (Garside *et al.*, 1987) tagset). These tags were used as our input, instead of specific words. Random selection of sentences was used to produce a training set of only 26,480 words, a cross validation set of 4365 words (30,845 words total, compared the million word corpora typically used (Charniak, 1997; Collins, 1999; Ratnaparkhi, 1999)), and a testing set of 4304 words. By using a small training set we are placing greater emphasis on the ability of the parser to generalize to novel cases in a linguistically appropriate way, and to do so robustly.

As was done for previous work on SSN parsers (Henderson & Lane, 1998), we also simplified the structures in the corpus slightly so that they could be expressed in terms of only *parent* and *grandparent* relationships. In particular, some pairs of constituents are conflated so that every constituent has a word as an immediate child, since otherwise great-grandparent relationships would be necessary. This is a local modification which does not change the recursive nature of the structures or the ability to extract predicate-argument information from them. It is also motivated on linguistic grounds (Henderson, 2000). However, we should emphasize that it would be possible to define a different input-output format for an SSN parser which could use any corpus's definition of constituency.

5.2. Training the Models

Neural network training is an iterative process, which requires intermediate testing with a cross validation set to avoid over-fitting. This technique also allowed us to develop multiple versions of the network and evaluate them using the cross validation set, without ever using the testing set until a single network has been chosen. Our chosen network trained for 325 passes through the training set before reaching a maximum of the average between its recall and precision on constituents in the cross validation set.

Estimating the PSR model is straightforward. All the word pairs associated with each structural relationship (or label-head pairs) are extracted, counted, and normalized in accordance with the probability model. Because this process does not require a cross validation set, we use the combination of the network's training set and the network's cross validation set.

Estimating the parameters of a PCFG is also straightforward. All the sequences of child labels that occur in the corpus for each parent label need to be extracted, counted, and normalized in accordance with the conditional probabilities required by the model. As with the PSR model, we use the network's training set plus its cross validation set to estimate the probabilities.

[4] It should be noted that in situations where large amounts of training data are available, relative to the amount of variability in the domain, then weaker independence assumptions could be made and robustness could be achieved without resorting to neural networks or other sophisticated techniques for smoothing. Also, in such situations the SSN parser would not be appropriate, under its current implementation, because the training process is too slow for the use of very large datasets to be feasible.

5.3. Testing Results

Once all development and training had been completed, the SSN, the PSR, and the PCFG were tested on the data in the testing set. For each model the most probable parse according to the model was taken as the output of the parser.[5] The results of this testing are shown in table 1.

The main concern of this paper is whether the SSN does a better job at estimating the parameters of the probabilistic model than counting frequencies. The performance of the SSN model compared to the PSR model shows that it clearly does. Under every measure the SSN does much better than the PSR.

The first column shows the percentage of sentences for which some parse was found. The SSN never outputs estimates that are exactly zero, so there is always some basis for choosing a parse. For some cases the PSR model did not provide enough nonzero probabilities to construct a parse with nonzero probability, so the sentence could not be parsed. But this is true in only 6.6% of the sentences, showing that our independence assumptions were sufficiently strong to get reasonable estimates of the probabilities. For a robust parser we would need to totally remove these unparsed cases, so we also applied a simple form of smoothing to the PSR, called "PSR Sm" in table 1. In this model, zero probabilities are eliminated by adding 0.5 to all the counts. Consequently this model also finds parses for 100% of the sentences, but its performance is worse under all the other measures.

The remaining four columns in table 1 give the performance of each parser in terms of recall (percentage of desired which are output) and precision (percentage of output which are desired) on both constituents and parent-child relationships. An output constituent is the same as a desired constituent if they contain the same words and have the same label. This measure is a common one for comparing parsers. Parent-child relationships are the result of interpreting the parse as a form of dependency structure. Each parent-child relationship in the parse is interpreted as a dependency from the child word (or the head word of the child constituent) to the head word of the parent constituent. Two such relationships are the same if their words are the same. This measure is more closely related to the output of the SSN parser and the PSR model, and may be more appropriate for some applications.

From the relative performance of the PSR and the SSN under all these measures it is clear that the SSN is able to extract important information from words which the SSN model assumed were less relevant, but the PSR model assumed were totally irrelevant. The PSR model had to impose these independence assumptions in order to get sufficient counts for estimating the model's probabilities. These independence assumptions turned out to be strong enough to get reasonable coverage, but they were too strong to get good performance.

The PCFG was tested in order to provide a baseline for evaluating the results for the SSN and PSR models.[6] The first thing to notice about the testing results is that the PCFG only found parses for about half of the sentences. This lack of robustness is a consequence of making weaker independence assumptions than the PSR model.

Rather than providing a default output in the unparsed cases, we computed performance statistics on the subset of sentences for which the PCFG did parse, shown in table 2. Even on

[5]We would like to thank Jean-Cedric Chappelier and the LIA-DI at EPFL, Lausanne, Switzerland for providing the tools used to train and test the PCFG.

[6]Note that this is a simple PCFG. We have applied no sophisticated smoothing and we have not modified the corpus labeling to make it more appropriate for the generalizations learned by a PCFG (as we did do for the SSN parser).

	Sentences		Constituents		Parent-child	
	Parsed	Correct	Recall	Precision	Recall	Precision
SSN	100%	14.4%	64.2%	64.8%	82.1%	82.3%
PSR	93.4%	2.8%	38.2%	40.1%	64.7%	65.9%
PSR Sm	100%	2.8%	35.9%	36.8%	58.8%	59.4%
PCFG	50.8%	3.3%	29.2%	53.7%	38.2%	73.3%

Table 1: Testing results.

	Sentences		Constituents		Parent-child	
	Parsed	Correct	Recall	Precision	Recall	Precision
SSN	100%	16.3%	65.1%	66.5%	83.0%	83.6%
PSR	97.8%	4.3%	40.1%	41.2%	68.5%	69.2%
PCFG		6.5%	57.5%	53.7%	75.2%	73.3%

Table 2: Testing results on the sentences parsed by the PCFG.

this subset the SSN parser did better than the PCFG under every measure. This baseline ensures that the SSN parser's results are sufficiently good that it is worth answering questions about its specific characteristics, as we are doing here in the comparison with the PSR model.

The subset of sentences which the PCFG does parse can also be used as an indication of the "easy" sentences, relative to the half which it does not parse. By comparing performance on the "easy" half of the test set to that on the total set, we can get an indication of the robustness of the methods. The PSR model has only a third as many unparsed sentences on the "easy" half as on the total, and the other performance measures reflect this. In contrast, the SSN parser only improves slightly on the easy sentences, relative to the total set, indicating its robustness on the hard sentences.

References

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

CHARNIAK E. (1997). Statistical techniques for natural language parsing. *AI Magazine*.

COLLINS M. (1999). *Head-Driven Statistical Models for Natural Language Parsing*. PhD thesis, University of Pennsylvania, Philadelphia, PA.

GARSIDE R., LEECH G. & (EDS) G. S. (1987). *The Computational Analysis of English: a corpus-based approach*. Longman Group UK Limited.

HENDERSON J. (2000). A neural network parser that handles sparse data. In *Proceedings of the 6th International Workshop on Parsing Technologies*, p. 123–134, Trento, Italy.

HENDERSON J. & LANE P. (1998). A connectionist architecture for learning to parse. In *Proceedings of COLING-ACL*, p. 531–537, Montreal, Quebec, Canada.

LANE P. & HENDERSON J. (1998). Simple synchrony networks: Learning to parse natural language with temporal synchrony variable binding. In *Proceedings of the International Conference on Artificial Neural Networks*, p. 615–620, Skovde, Sweden.

RATNAPARKHI A. (1999). Learning to parse natural language with maximum entropy models. *Machine Learning*, **34**, 151–175.

SAMPSON G. (1995). *English for the Computer*. Oxford, UK: Oxford University Press.