# A corpus-based connectionist architecture for large-scale natural language parsing

JONATHAN A. TEPPER[a], HEATHER M. POWELL[a, *] and DOMINIC PALMER-BROWN[b]

[a]*The Nottingham Trent University, Dept of Computing & Maths, Burton Street, Nottingham, NG1 4BU,UK*
[b]*Leeds Metropolitan University, School of Computing, Beckett Park, Leeds, LS6 3QS, UK*

*Abstract.* We describe a deterministic shift-reduce parsing model that combines the advantages of connectionism with those of traditional symbolic models for parsing realistic sub-domains of natural language. It is a modular system that learns to annotate natural language texts with syntactic structure. The parser acquires its linguistic knowledge directly from pre-parsed sentence examples extracted from an annotated corpus. The connectionist modules enable the automatic learning of linguistic constraints and provide a distributed representation of linguistic information that exhibits tolerance to grammatical variation. The inputs and outputs of the connectionist modules represent symbolic information which can be easily manipulated and interpreted and provide the basis for organising the parse. Performance is evaluated using labelled precision and recall. (For a test set of 4,128 words, precision and recall of 75% and 69% respectively were achieved). The work presented represents a significant step towards demonstrating that broad coverage parsing of natural language can be achieved with simple hybrid connectionist architectures which approximate shift-reduce parsing behaviours. Crucially, the model is adaptable to the grammatical framework of the training corpus used and so is not predisposed to a particular grammatical formalism.

*Keywords:* connectionist networks, hybrid architectures, natural language processing, deterministic shift-reduce parsing, corpus linguistics, treebank grammar

## 1. Introduction

Syntactic analysis or *parsing* (see Sells 1985, Dale *et al.* 2000) plays a central role in Natural Language Understanding (NLU) systems (Winograd 1983, Allen 1995) and is used to recognise the structural relationships between words within a sentence. The ability to syntactically parse realistic subsets of

---

∗Corresponding author.
*E-mail address:* Heather.Powell@ntu.ac.uk

unconstrained natural language remains one of the greatest obstacles to achieving practical NLU systems.

Successful statistical models such as Magerman 1995, Collins 1997, Charniak 1997 and 2000, Bod 1996 and 2000 and Brill *et al.* 1998 typically employ such techniques as N-Gram statistics, Monte Carlo methods, Probabilistic Context-free Grammars (PCFG) using the Viterbi algorithm or the Inside-Outside re-estimation algorithm for estimating derivational probabilities. In the last ten years, symbolic parsers have incorporated numerical processing, heuristics and transformation techniques (see Gibson 1991, Hindle and Rooth 1991, Brill 1993, Hindle 1993, Brill and Resnik 1994) to either induce treebank grammars directly from annotated text corpora, or to process realistic syntactic structures. In fact, the vast amount of NLU research being reported in this area suggests a significant shift from strictly symbolic rule-based approaches towards the more successful hybrid symbolic/numerical approaches.

What is also clear, however, is that similar levels of success have yet to be achieved by parsers that use connectionist networks (Feldman and Ballard 1982, Rumelhart *et al*. 1986, Smolensky 1990a, Dyer 1995, Wermter and Sun 1999, Palmer-Brown *et al.* 2002) with either localist or distributed representations. Successful localist connectionist parsers have typically utilised either phase synchronisation (Shastri and Ajjanagadde 1993, Henderson 1994) or Competition-based Spreading Activation (Reggia 1987, Stevenson 1994, Stevenson and Merlo 1997) to illustrate that by combining localist representations with symbolic computation and linguistic theory it is possible to model human syntactic processing and to some degree process realistic grammatical structures without learning. Distributed connectionist parsers typically consist of combinations of feed-forward Multi-layered Perceptron (FF-MLP) networks and locally recurrent Multi-layered Perceptron networks trained with the Back-propagation learning algorithm.

Pure distributed connectionist parsers, such as Cleeremans *et al.* 1989, Miikkulainen 1990 and 1995, Elman 1991, Reilly 1992, Berg 1992, Sharkey and Sharkey 1992, Giles *et al.* 1990, Ho and Chan 1997, and Mayberry and Miikkulainen 2000, achieved limited success but have not yet been shown to scale-up to realistic natural language domains. Rather than use pre-parsed corpora as a basis for training the system, these parsing models have been limited to using hand-crafted linguistic grammars as the basis for constructing the parser's training and test data. Such systems typically risk high precision and low coverage because the inflexibility of the rule-based constraints limits the number of allowable grammatical structures. It could also be questioned whether presenting a connectionist architecture directly with artificial grammar rules or associating an architecture with a specific grammar is beneficial or holds any advantage over classical symbolic parsers.

An alternative and more successful approach is to tightly couple connectionist and symbolic modules so that they work together (Wermter and Weber 1994, Buo *et al*. 1994, Buo and Waibel 1996, Henderson and Lane 1998). With the exception of Buo *et al*. 1994 the connectionist modules in these parsers were also trained using naturally occurring corpus data. This seems a more natural use of neural networks. They are given the task of implicitly deducing linguistic constraints directly from naturally occurring language samples.

As the task of producing an annotated text corpus is very resource intensive, there are few to choose from that have been successfully annotated with syntactic structure to form a reasonably sized treebank. However, a number of suitable treebanks are available (Johansson *et al*. 1978, Francis and Ku era 1979, Garside *et al*. 1987, Marcus *et al*. 1998 and BNC Consortium 1995) and have been used successfully by a number of researchers. A natural complication of using different corpora is that there is no definitive method for comparing the performance of such 'treebank' parsers. There is also no singularly adhered to standard of corpus annotation, thus complicating comparison further[1].

To date, perhaps the most promising results achieved on a corpus of naturally occurring language with a connectionist architecture are those reported by Henderson and Lane 1998. Henderson and Lane integrate Elman's Simple Recurrent Network (SRN) (Elman 1990) with phase synchronisation to form Simple Synchrony Networks (SSN). This model incorporates learning into Henderson's original localist parsing model (Henderson 1994) which used phase synchronisation and a hand-crafted grammar to perform syntactic parsing. Henderson and Lane applied SSNs to the Susanne corpus to identify 'grand parent', 'parent' and 'sibling' relationships. This is not full parsing but performance levels reported for this task were starting to approach those attained by Charniak 1997. However, there is still no generally accepted method of combining connectionist, symbolic and statistical architectures to effectively perform broad coverage natural language parsing.

## *Motivation*

This study contributes to the understanding of hybrid modular connectionist parsers trained on a relatively small sample of language drawn from a corpus and investigates whether they can acquire the ability to process a wide range of naturally occurring language as exemplified by the contents of the

1 Although Leech 1993 defined a number of principles for corpus annotation there are still clear distinctions between different schemes.

corpus. It aims to assess performance generalisation and degradation for syntactic structures that are not present in the training phase. We are equally concerned with evaluating a particular hypothesis concerning the nature of a modular architecture.

Corpus-based, statistical parsing methods have achieved notable successes in recent years (McMahon and Smith 1998). We adopt a corpus based approach that is statistical in terms of training and evaluation, but modular-connectionist in architecture, with low-level syntactic information represented as patterns of activation. Architectures based on FF-MLPs and SRNs have been shown to be capable of approximating human performance on acquiring and processing syntactic sequences, especially in terms of their humanlike limited ability to accurately predict sequences which contain embedding (Christiansen and Chater 2001). However, there has been little work extending these approaches to large scale parsing of real-world samples of natural language. This work starts to address the question of whether a modular-connectionist approach scales-up in a cognitively plausible fashion.

Recent evidence on human parsing (Friederici 2002) supports the notion of a syntactic processing phase that precedes the semantic stages of processing. The work presented here evaluates, from a connectionist standpoint, the computational plausibility of a particular modular decomposition of the syntactic parsing process. Shift-reduce parsing is executed by the coordinated action of three connectionist modules operating alternately to scan symbols (in the form of patterns) to a) find left-hand phrase boundaries, b) find right-hand phrase boundaries, and c) recognise the segmented phrases. This particular modular decomposition is one that emerges as a natural candidate from a connectionist viewpoint, where finding the location of objects (phrases) and edges (phrase boundaries) in a pattern space is a well established approach to pattern recognition.

The main task of the parser presented in this paper is to annotate natural language texts with syntactic structure according to the annotation scheme used in the training corpus. In other words, the parser has to learn the implicit linguistic constraints used to originally annotate the training corpus. The aim of the research is to produce an efficient parsing system that is trainable, robust and able to learn realistic grammatical constraints without the explicit use of grammar rules. Another aim of this work is to produce a parsing architecture that is not ultimately predisposed to a particular grammatical framework or theory of syntax, unlike previous connectionist-symbolic hybrid parsing systems. This will widen the scope and reusability of the parser, particularly for different corpora using different annotation schemes. The parser aims for broad coverage of language, therefore its performance is assessed using statistical techniques. Module level performance is evaluated in terms of network generalisation and sentence level performance is evaluated in terms of precision and recall (Harrison *et al.* 1991). Generalisation is the percentage of correct outputs when the network is given previously unseen data. We further distinguish between *pure* and *natural* generalisation. Unseen text is likely to include grammatical structures in common with the training text samples. If these are removed then all of the test data is new and performance on this gives rise to pure generalisation. This indicates how well the network has learnt the general problem. If the common structures are left in the test sample then the performance gives rise to natural generalisation. This indicates how useful the learned representations are for the given natural language task.

*Organisation of the Paper*

Section 2 begins with a detailed description of the hybrid parsing architecture and algorithm. In section 3 the pre-parsed corpus samples and tag representations used for training and testing the system are described. In sections 4 and 5 the phrase delimitation and recognition aspects of the parser are presented. They include the training and test performance for each module. In sections 6 and 7 the sentence level performance of the parser is considered. Aspects of its behaviour are noted and its performance is evaluated and compared with that of previously reported connectionist parsers. The paper concludes with a summary of the main findings in section 8.

## 2. Parsing Architecture and Algorithm

### 2.1 *Overview of the Architecture*

Work reported in Tepper *et al.* 1995a and 1995b and Tepper 2000 illustrated the limitations of using a single FF-MLP to learn all functions of a shift-reduce parser, i.e. to group and classify tags into phrase types from sequences presented. The conclusion drawn is that the overall task is too complex for a single network. An alternative is to decompose the problem into two or more stages or parts (Tepper *et al.* 2001). The process of parsing involves both identifying phrase boundaries and recognising the different types of phrase structure. Two stages to the processing therefore suggest themselves: delimitation of phrases and recognition of phrase structure. By decomposing the problem the dimensionality of the search space is reduced.

#### 2.1.1 *Phrase Delimitation*

Delimitation involves identifying both beginnings and ends of phrases or clauses. It is hypothesised that when scanning a tag sequence, the transitions at starts and ends of phrases would form two distinct clusters in terms of the distribution of their characteristic features. The delimitation process is therefore decomposed into two sub-processes which should each be easier to learn than full delimitation. This further decomposition is empirically supported by the difficulty in training these modules such that all of the training data is acquired (see section 4). We have a right-to-left delimiter (RLD) process to discover the beginning of a phrase and a left-to-right delimiter (LRD) process to discover the corresponding end of the phrase. The number of input symbols processed by either delimiter before the beginning (or end) of a phrase is encountered is variable and not known *a priori*.

An FF-MLP would require a temporal input window of sufficient width to process the largest symbol distance between the end of a sentence (where parsing begins) and the nearest start of a phrase boundary. This approach is inefficient as the window width would be defined for the worst (longest) case and in most cases there would be considerable redundancy involved in processing the entire window. A better approach would seem to be to apply a network able to process the input sequentially. Recurrent MLPs that have connections feeding from the hidden units back to the input units, such as Elman’s SRN, have proved very successful at temporal processing of linguistic input and at enabling temporal relationships to be established between sequential input symbols (see Lawrence *et al.* 2000 for a comparative review of architectures for grammar induction). However, SRNs possess severe memory limitations due to the use of the untrained ‘don’t care’ cycles. This means that errors in processing cannot be corrected until the target output is encountered. This is problematic when the target output vector is not encountered for several time steps of unconstrained processing. The Temporal Auto-associative SRN (TASRN) developed by Ghahramani and Allen 1991 is therefore used. This attempts to overcome the problem by ensuring that targets, which relate to previous inputs, are available at every processing stage. The TASRN compares favourably with the SRN for both the LRD and RLD delimiter tasks (see Tepper 2000 for a full comparative analysis). The LRD and RLD are presented in detail in Section 4.

#### 2.1.2 *Phrase Structure Recognition*

Despite the failure of a single FF-MLP to learn full phrase reduction processing this architecture *is* suitable for a phrase structure recogniser module (termed the PSR module). This can be justified for a number of reasons: firstly, the network is only required to recognise the overall phrase type as opposed to processing embedded structure; secondly, FF-MLPs have a general record of success in classification problems where the maximum width of the window is known; and finally, unlike the delimitation problem there is no requirement to determine position since phrase boundaries have already been found. This also means that the network does not have the burden of being required to determine the validity of a phrase or clause. A ‘nearest match’ computation is therefore applied to the output activations to find the closest valid phrase for the given input. This may produce useful output if the system is presented with slightly ungrammatical sentences or to recover from errors produced by the other modules. It enables a “best effort” parse in all circumstances. The PSR is discussed in detail in Section 5.

### 2.1.3 *Symbolic Parse Organisation*

Three core symbolic structures are used to store input symbols (representing word and constituent tags), properties of these input symbols and the phrase structure tree. These are: a linked list to store tag information; a stack to store parse state information and the resulting parse tree (the *Parse-stack*); and finally a Last-In-First-Out stack to store the current input state (the *Input-stack*). Two temporary stacks are also used to store transient symbols during delimitation. Standard symbolic structures have been employed because existing connectionist approaches to structure representation (Pollack 1990, Kwasny and Kalman 1995, Plate 1995, Callan and Palmer-Brown 1997) and variable binding (Smolensky 1990b, Sun 1992, Shastri and Ajjanagadde 1993) have not been shown to generalise effectively in order to process realistic language domains and at present do not offer significant advantages over symbolic approaches. The symbolic components are tightly coupled with the connectionist components of the parser. The latter represent linguistic constraint and signal the actions to be performed, whilst the former allow simple communication between the networks and allow interpretation of the parser’s actions as a whole. The *Scheduler* controls the symbolic and connectionist components and the flow of information between them.

A schematic of the architecture developed is shown in figure 1.

[Insert figure 1 about here]

## 2.2 *Description of the Algorithm*

The Scheduler supervises a deterministic shift-reduce parsing strategy that parses from right-to-left. The strategy implemented is similar to that defined by Shieber (1983) and was selected due to its simplicity (only two actions) and efficiency. Parsing from right-to-left has also proved to be efficient in that local ambiguity in English can be resolved sooner than with left-to-right parsing (making training easier) and the number of Parse-stack actions can be significantly reduced (see Tepper 2000).

Rather than processing at word level, the parser only accepts as input the word tags and constituent tags used to annotate the corpus. Word tags represent the grammatical class of the word. Although this means that the corpus must be pre-tagged, it has the advantage of significantly reducing the possible number of inputs to consider as there are obviously far fewer different word tags than there are words. The annotated corpus used is the Lancaster Parsed Corpus (LPC) (Garside *et al*. 1987). Each word in the LPC is followed by an underscore character and a sequence of symbols (normally capital letters) which represents a word tag. For example, 'become_VB' represents the word 'become' acting as a VB, which refers to a 'base form of a lexical verb'. The constituent tags are the non-terminal symbols that label the phrases represented by groups of word tags and other constituent tags. Constituent tags in the LPC are denoted by capital letters that indicate the major class of constituent that the tag labels with additional lower-case letters to indicate a sub-classification.

The output of the parser after each shift-reduce processing stage is a phrase or clause represented in labelled bracketed format (see Sells 1985). In the LPC, *[* represents the beginning of a constituent, and *]* represents the end or completion of a constituent. The symbols alongside these brackets are labels for constituents. For example, *[V* denotes the beginning of a verb phrase and *V]* denotes the end of a verb phrase. The final parser output is a labelled bracketed structure that encodes the parse tree for the entire sentence. A typical output example is shown in figure 2b together with the associated parse tree, figure 2a.

[Insert figure 2 about here]

When parsing of a sentence begins, its tokens are passed sequentially to the RLD starting at the last token of the sentence and shifting to the left. The output of the RLD triggers to indicate the start of the first phrase to be reduced, i.e. the left-hand phrase boundary has been identified. Tokens are then passed sequentially to the LRD, starting with this left-hand boundary token but now shifting to the right. The output of the LRD triggers to indicate that the right-hand phrase boundary has been identified, i.e. the end of the first phrase to be reduced has been found. The Scheduler now has enough information to define the position and width of the reduction window, which is the input to the PSR.

The reduction indicated by the PSR is then substituted for the phrase in the sentence sequence on the Input-stack and delimitation begins again. The delimiters are reset, and the RLD input is taken once again from the end of the sentence stored on the Input-stack. This time it will receive one non-terminal symbol (constituent tag) amongst the remaining terminal symbols (word tags) in its input sequence. At each reduction stage the scheduler updates the parse stack so that it holds the current state of the parse in labelled bracketed form (see Figure 2b). This involves popping (and then restoring) sufficient elements to be able to insert the end of the phrase indicator (e.g. *V]*) as well as pushing the start of phrase indicator and any word tags that are now part of the reduced phrase.

By repeating this process, of delimitation followed by reduction and substitution, the parser continues until the PSR produces the constituent tag, *S*, which denotes a completed parse. Table 1 shows the stages involved in the shift-reduce delimitation and recognition method for the sentence in figure 2.

[Insert table 1 about here]

Although not shown in the above example parse, full-stop symbols are placed at the beginning and end of the sentence to act as begin and end markers. These are the only punctuation symbols.

To enable the parser to settle on one parse, a strictly deterministic parsing mechanism is employed whereby sufficient contextual information is required by the delimiters and recogniser at each stage. Consequently, the RLD does not immediately trigger on a left-hand phrase boundary token, but has a delayed response to allow it to take into account sufficient look-back context (symbols to the left of the left-hand phrase boundary) to establish the validity of the boundary. The look-back, i.e. delay in the response, is fixed at 4. The LRD requires context before and after the phrase: it has 2 look-back symbols and 2 look-ahead symbols. The PSR also requires both look-ahead and look-back symbols in order to put the phrase into sufficient context to determine its type. The input window for the PSR therefore consists of the symbols to be reduced with some symbols either side (maximum of 10 phrase symbols, 4 look-back and 1 look-ahead).

With phrases of less than 10 symbols or where the full look-ahead/back is not available because the phrase is too near a sentence boundary, the fields are padded out with null symbols, denoted by ^. In the case of the delimiters, this allows phrase boundaries to be signalled after the fixed look-back/ahead context. The number of look-ahead/back symbols used for each module is the minimum required to provide unambiguous training data.

## 3. Training and Test Samples

### 3.1 *Lancaster Parsed Corpus*

The LPC is a corpus of English sentences excerpted from printed publications of the year 1961, and is a subset of the LOB corpus. Each word is tagged with its syntactic category and each sentence in the LPC has undergone syntactic analysis in the form of labelled bracketing. The LPC contains 11,827 sentences (13.29% of the LOB corpus).

## 3.2 *Tag Representations*

Design of the input representation is part of adapting the model to a given pre-tagged corpus. The approach is to help the training process by reflecting known similarities between symbols into their coding. This is achieved by separating the input space into regions such that each corresponds to a different symbol type, i.e. represents a group of symbols of the same type. There are 5 terminal symbol groups and 7 non-terminal symbol groups. These 12 groups are represented by separate fields of the input vector. The symbols within a group are represented within the group field using linear binary coding, with an additional bit in the field to indicate a symbol of this group type. Therefore the number of bits in a field is the minimum required to represent all symbols in the corresponding group plus 1, e.g. as there are 83 symbol types in the noun phrase group, the field representing this group is 8 bits wide (7 bits are needed for 83 types). This coding ensures that patterns for symbols in different groups are always orthogonal to one another whereas patterns for symbols within a group are not. Figure 3 shows the input field for each tag type.

[Insert figure 3 about here]

A total of 61 bits are used to encode all possible input symbols. The terminal symbol groups are: punctuation (Pu), conjunctions (Co), nouns (NP), verbs (VP) and prepositions (PP). The non-terminal symbol groups are sentences (S), finite clauses (F), non-finite clauses (T), major phrase types (V), minor phrase types (M) and slash tag phrases (A/B). Coordinated phrases (those consisting of more than one phrase of a particular type connected by for example ‘and’) are indicated using the final field (&+-=) in conjunction with the symbol for the type of phrase coordinated. (This coding means that the pattern for a coordinated phrase is non-orthogonal to all other coordinated phrases and to the patterns in the phrase group containing the uncoordinated version of that phrasal type).

## 3.3 *The Training Sample*

A subset of the LPC was selected for training. A complexity constraint defined in relation to the RLD was applied in order to bring training times down to manageable levels whilst still allowing the approach to parsing to be assessed on a large sample of naturally occurring language.

If the RLD network has to process more than 9 symbols before the beginning of a phrase is found then the sentence containing that phrase is not used in training or testing. Every 8th sentence was extracted for training provided it was within this complexity constraint.

The training sample consists of 654 sentences: 5.5% of the entire LPC corpus with average sentence length of 7 words. (Without the complexity constraint, 12.5% of the LPC with average sentence length of 13 would be selected). However, 5.5% is considered to be a reasonable coverage of the LPC allowing computationally feasible training sets containing simple and complex sentence structures.

### 3.4 *The Test Sample*

As with the training sample, the test sentences must be sampled across all text categories in the LPC. It is essential that none of the test sentences is present in the training sample. For this reason, the LPC was sampled at every 8th + 1 sentence then filtered according to the complexity constraint to generate the test data. It is assumed that there is no correlation resulting from sentence adjacency so the test and training data are independent samples. Therefore this systematic selection results in random samples within the two sets and enables tests to be performed with sentences that are comparable in terms of complexity to the training set sentences. As with the training sample, the test sample is also nearly 6% of the entire LPC at 5.8% and the average sentence length is also 7 words. The resulting composition of the test sample for each of the networks is discussed in Section 4.

## 4. Phrase Delimitation

### 4.1 *TASRN Delimiter Architecture and Training Sets*

Figure 4 shows the TASRN architecture used for the RLD and LRD networks. The symbols are sequentially presented to the input layer. The input layer has 61 input units to accommodate an input symbol. Additional input units, called context units, are used to store the previous hidden state. The current input symbol is therefore processed in the context of the previous symbols of the input sequence.

The TASRN is required to output a value between 0 and 1 which relates to how close a given input symbol is to a phrase boundary. The phrase boundary indicator is a single output unit that is trained with a ramp-step function indicating the phrase boundary and the proximity to the boundary, i.e. the first required output is a ‘don’t care’, followed by 0 ramping up to 0.4 for the penultimate symbol and 1 for the last symbol. It was found that without this ramp followed by a step function, the input patterns requiring a high output were swamped by the preceding sequence stages requiring zero outputs. This ramp mechanism may help the network to learn the lengths of sequences by being forced to learn how far it is through a phrase at each stage whilst retaining a significant margin between the last symbol and all previous ones. The network is also trained to reproduce the input symbol and previous hidden state onto the remaining output units, in accordance with the TASRN method.

[Insert figure 4 about here]

The high processing time did not allow for the exact number of hidden units for optimum generalisation to be found during the training phase. However, the minimum number of hidden units required to gain acceptable training results (above 90% of sequences learnt) was found for both delimiters. This was used as the basis for imposing a maximum number for the hidden units and the value below this giving the lowest root mean squared error (RMSE) for the training data was adopted.

Training is by Back-propagation with a momentum term of 0.9 and a bias unit for the input and hidden layers. All training is performed in on-line mode rather than batch mode. Hidden unit and output unit activation values are calculated using the standard sigmoid function. A pattern error-sensitive learning rate (Tepper *et al*. 1995a) was used as it was found to aid learning.

All context units are initialised to 0.5 before each new training (or test) sequence is presented. All training experiments are terminated at convergence (RMSE value <= 0.05) or 1,000 epochs whichever occurs first.

The raw LRD/RLD training sets generated from the training sample contained significant imbalances between sequences of different lengths. During preliminary training experiments, this resulted in a tendency to find a local minimum, with the networks learning the most frequent sequence lengths only. To overcome this problem, the training sets are balanced to produce a uniform distribution across different sequence lengths. This is achieved by first identifying and removing naturally occurring duplication in the sequences. Then sequences of the less common lengths are replicated. The length distributions before and after balancing for the RLD and LRD are shown in table 2.

[Insert table 2 about here]

### 4.2 *Delimiter Performance*

#### 4.2.1 *Delimiter Training Performance*

Table 3 shows the training results for the delimiters. The difference between the number of hidden units adopted and the better training results indicates that the LRD has a task of lower complexity than the RLD.

Both networks learnt a higher percentage of the replicated sequences than the unreplicated ones as might be expected, e.g. the RLD learnt 95% of the length 6 and 9 sequences.

[Insert table 3 about here]

4.2.2. *Delimiter Test Performance*

The composition of the RLD/LRD test samples is shown in table 4. *Natural* indicates the total in the sample, *pure* indicates the number that are left when sequences that coincidentally also occurred in the training set are removed. It can be seen that both test samples contain a high level (89% and 84%) of pure test sequences.

[Insert table 4 about here]

The resulting generalisation performance achieved during the test phase is a clear indication of the amount of language coverage gained by the networks. Table 5 summarises the natural and pure generalisation results obtained. The RLD configuration was able to achieve natural generalisation levels above 80% for all sequence lengths and correctly process nearly 92% of length 9 sequences, whilst the LRD achieved above 85% natural generalisation for all sequence lengths (90% or higher for all but one) and correctly processed 97% of length 8 sequences. Differences in the performance between different test sequence lengths is hypothesised to be determined by two factors: sequence length, longer being harder to learn; and level of replication in the test set, high replication giving better performance as there is less variation to learn.

[Insert table 5 about here]

The pure generalisation figures are similar to those for natural generalisation because of the high percentage of pure test sequences in the test data. Although neither delimiter fully acquired the training data, overall performances of ~85% and ~90% respectively were considered adequate to test the approach.

## 5. Phrase Structure Recognition

5.1 *FF-MLP Architecture*

The parser has no access to semantic or discourse information and relies purely on deterministic processing to resolve any phrasal ambiguity it encounters. It uses look-back and look-ahead symbols to provide enough context for this at each stage.

The number of look-back symbols and look-ahead symbols required for the recognition task is established empirically. 4 look-back symbols and 1 look-ahead symbol was the optimum combination to provide sufficient context to resolve ambiguity between the training phrases of the PSR. The maximum phrase length found within the entire LPC (10) was adopted for the PSR input. The PSR's input layer is therefore comprised of 15 symbols, which at 61 bits per symbol results in a total of *915* input units. The output layer represents the reduction symbol for the input phrase and as such is 61 bits for one symbol.

The architecture for the PSR is illustrated in figure 5. The recognition task is simplified by the fact that the phrase is kept in the same position within the input layer and the boundaries between the look-back and look-ahead symbols are distinct. If the look-back or phrase is shorter than the fixed limit imposed then it is appropriately padded out with null symbols to provide this position invariance.

The PSR is not required to validate phrases so the output vector is subject to a nearest match classification to give the best attempt in all circumstances. As the output unit activation values are continuous they are first converted to discrete values. The resulting output vector is then compared with the set of possible constituent tag representations and the one that it is closest to according to its Euclidean distance is taken as the reduction.

[Insert figure 5 about here]

## 5.2 *Recogniser Training Performance*

The training sample for the PSR consists of 2,588 training patterns, each corresponding to one of 72 constituent tag types (see section 3.2). It was trained in on-line mode using back-propagation with a momentum term of 0.9. A bias unit of 1.0 is included in the input and hidden layers.

The configuration adopted had 50 hidden units as all training patterns were learnt within 800 epochs (configurations with 40 and 45 hidden units were unable to acquire the training data after 1000 epochs). This shows that if the complexity of the recognition task is reduced by reducing the variations in phrase position and removing the requirement to also validate phrases, an FF-MLP architecture is adequate to learn the task.

## 5.3 *Recogniser Test Performance*

The test sample consists of a total of 2,765 test patterns after natural occurring pattern replication has been removed. 88% of these test patterns were not found in the training data, i.e. were pure. A detailed analysis of the basic phrasal composition of the test sample showed that above 90% of the noun (*N*), preposition (*P*) and adjective (*J*) phrases in the test sample consist of pure test patterns. Above 80% of the verb (*V*) phrases and adverbial (*R*) phrases are also pure test patterns. These high levels are due to repeated basic phrase structures being presented in a variety of contexts.

For both natural and pure generalisation tests, the RMSE values were based upon the raw output activations of the PSR. A summary of the natural and pure generalisation results for the test sample is shown in table 6. It can be seen from this that using the nearest match computation significantly improves the levels of generalisation (10% increase). The number of incorrect classifications produced by the PSR, after the nearest match computation has been applied, is an indication of the level of recognition failure to be expected during parsing.

[Insert table 6 about here]

## 6. Sentence Level Parse Performance

We use the PARSEVAL measures that have become standard for assessing statistical broad coverage parsers (Harrison *et al*. 1991) to assess sentence level performance :

$$\text{Labelled Precision} = \frac{\textit{number of correct constituents output by parser}}{\textit{number of constituents output by parser}}$$

$$\text{Labelled Recall} = \frac{\textit{number of correct constituents output by parser}}{\textit{number of constituents in treebank parse}}$$

Correct means the constituents have the same grouping *and* labelling as those found in the treebank parse. These standard measures evaluate the level of correctness where the produced parse does not match the treebank parse exactly.

[Insert table 7 about here]

Table 7 shows the sentence level results for the training and test sets. The figures show that the parser produced a completed parse for 93.4% of the test sentences. Whilst the parser gave exact matches for around half of the test sentences, the labelled precision and recall figures show that overall, the mismatched parses often contain significantly similar structure to their corresponding treebank parses.

An analysis of the exact matches showed that the parser was typically able to match simple and compound sentence structures (those structures containing coordinating conjunctions where clauses are joined with a conjunction like *and*, *but*, *or*, *nor*, or *neither*). This is illustrated by the parse for the sentence, *it greatly improves the appearance and the strength*, shown in figure 6. It can be seen that the parser correctly joins the coordinated noun phrase, *and the strength*, to the noun phrase, *the appearance*, to form a compound noun phrase denoted by *N&*. The parser was also able to correctly process complex sentences containing subordinating conjunctions although the number of errors increased with respect to sentence length. The maximum sentence length correctly matched in the test set was 11 words.

[Insert figure 6 about here]

The parser demonstrates a preference for attaching the right-most preposition phrase to the nearest noun phrase. It can be seen in figure 7 that rather than attaching the preposition phrase, *in the water*, directly to *S+* so that it functions as an adverbial phrase, the parser attaches the prepositional phrase as a post-modifier to the noun phrase, *his hand*. This behaviour is similar to Minimal Attachment preferences discussed by Frazier and Fodor (1978) and a linguistic feature consistently exhibited by the parser correctly and incorrectly. However, the parser's output is a reasonable alternative and very close to the corresponding LPC parse.

Although the parser failed to parse 45 (6.6%) of the test sentences, only two unlearnt training sequences were responsible for parse failures. The parser fails if one of the delimiters fails to trigger, i.e. fails to detect a pattern indicating a phrase boundary. This may be a fault at the current delimitation stage or the result of an error made at a preceding delimitation or recognition stage. The errors can be mainly attributed to the incorrect pure generalisations made by the RLD and PSR networks. The minimum length of the 45 sentences is 6 words, the average is 12 words and the maximum is 24 words. Even though the PSR network learnt all of its training data, it significantly contributed to all 45 parse failures. Clearly, the PSR network is sensitive to position and context, therefore these generalisation failures represent a weakness of the phrase invariant recognition technique. However, a 6.6% failure rate is low and demonstrates the ability of the networks to generalise and compensate for failures made by each other.

[Insert figure 7 about here]

## 7. Comparisons with Previous Work

As outlined in the introduction, most work on connectionist parsing has been based on language generated from artificial context-free grammars and so has not attempted the much more difficult problem of processing naturally occurring language. The most relevant is that of Buo *et al.* 1994 who report a similar approach to parsing as the one taken here. They use one network to find phrase boundaries and one to label the phrase. Information to assess the work fully is not available in the paper. However it is clear that as well as being limited by using a simple context-free grammar, the architecture is less powerful as only two feed-forward networks were used. The finite network input vector imposes an upper limit on the length of phrase that can be processed.

In contrast, Wermter and Weber (1994) use a corpus, but this contains only 37 utterances of 394 words so is not very comparable. Buo and Waibel (1996) also use a corpus of spoken dialogues, with 600 sentences for training several neural networks. Their system selects the network with the best response at each stage. This is based on the relative activation levels, together with the result of comparing the possible alternatives with information from a rule set of syntactic structure. They report performances of 71.8% correct parses on the test data but only 33.8% when the rule-based information is not used. It therefore seems that most of the useful linguistic processing is being done symbolically.

The most comparable work is that reported in Henderson and Lane (1998). They use a different corpus (the Susanne corpus), which precludes direct comparison. Their training set (13,523 words) is much larger than their test set (4,602 words), the latter being comparable in size to our test set. Their architecture finds syntactic relationships only in terms of ‘grandparent’, ‘parent’ and ‘sibling’, i.e. there is no attempt at constituent labelling. They report slightly worse precision and recall figures of 62.6% and 69.4% respectively on the test data, but it is important to note that this is not *labelled* precision and recall as treebank constituent labels are not used. Therefore this is a less exacting measure. Although possessing potentially useful features such as the ability to revise decisions in the light of subsequent information, a weakness of Henderson and Lane’s architecture is again that there is an implicit limit on the length of sentence that can be processed (due to the encoding of time on the inputs and outputs).

The current state-of-the-art performance for a statistical parser (Charniak 2000) gives 90.1% average labelled precision/recall for sentences of length 40 and less, and 89.5% for sentences of length 100 and less when trained and tested on a standard section of the Wall Street Journal tree-bank.

## 8. Conclusions

This work represents a significant step towards demonstrating that broad coverage parsing of natural language can be achieved with a simple hybrid connectionist architecture for shift-reduce parsing.

A natural language parsing architecture has been described that is trainable and able to automatically learn linguistic constraints directly from annotated text corpora. The model is a hybrid architecture that relies on the integration of standard symbolic representation and manipulation techniques with a novel approach to deterministic shift-reduce parsing using connectionist modules. The method of phrase delimitation and recognition developed allows robust parsing of a wide-range of naturally occurring English text without the use of explicit phrase structure rules.

The parser is also not restricted to any particular type of grammatical formalism. Since the grammatical formalism used to annotate the training corpus is not hard-wired into the parsing architecture, the parser is easily adapted to a new treebank using a different annotation scheme by modifying the tag representations and then generating the appropriate training data. It relies upon the connectionist modules of the system to learn the actual linguistic constraints embedded within the corpus. Right-to-left parsing was adopted to match the higher level of embedding found towards the end of English sentences. However, the parsing mechanism will work in either direction provided the individual modules are able to acquire the training data.

## References

Allen, J., 1995, *Natural Language Understanding.* 2nd Edition, (The Benjamin/Cummings Publishing Company, Inc.).

Berg, G., 1992, A connectionist parser with recursive sentence structure and lexical disambiguation. In *Proceedings of the Tenth National Conference on Artificial Intelligence (AAAI-92),* Cambridge, MA, pp 32-37, (AAAI Press/MIT Press).

BNC Consortium, 1995, *British National Corpus.* (Oxford University Press).

Bod, R., 1996, Monte Carlo Parsing. *Recent Advances in Parsing Technology,* (Kluwer Academic Publishers, Boston) , pp 255-280.

Bod, R., 2000, Parsing with the shortest derivation. In *Proceedings of COLING 2000*, Saarbrucken, Germany.

Brill, E., 1993, *A Corpus-Based Approach to Language Learning*. Unpublished Ph.D Thesis, University of Pennsylvania.

Brill, E., Florian, R., Henderson, J. C., and Mangu, L., 1998, Beyond N-Grams : can linguistic Sophistication improve language modeling? In *Proceedings of COLING '98*, Qu'ebec, Canada.

Brill, E., and Resnik, P., 1994, A rule-based approach to prepositional phrase attachment disambiguation. In *Proceedings of COLING '94*, Kyoto, Japan.

Buo, F. D., Polzin, T. S., and Waibel, A., 1994, Learning complex output representations in connectionist parsing of spoken language. In *Proceedings of the International Conference on Acoustic, Speech & Signal Processing (ICASSP '94)*, Adelaide, Australia, pp 365-368.

Buo, F. D., and Waibel, A., 1996, FeasPar - a feature structure parser learning to parse spoken language. In *Proceedings of the COLING '96*, Kopenhagen.

Callan, R., and Palmer-Brown, D., 1997, (S)RAAM : An analytical technique for fast and reliable derivation of connectionist symbol structure representations. *Connection Science*, **9**: 139-159.

Charniak, E., 1997, Statistical parsing with a context-free grammar and word statistics. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence*, Menlo Park, CA, pp 598-603, (AAAI Press/MIT Press).

Charniak, E., 2000, A maximum-entropy-inspired parser. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-2000)*, pp 132-139.

Christiansen, M. H., and Chater, N., 2001, *Connectionist Psycholinguistics*. (Ablex Publishing).

Cleeremans, A., Servan-Schreiber, D., and McClelland, J., 1989 Finite state automata and simple recurrent networks. *Neural Computation.* **1 (3)**: 372-381.

Collins, M. J., 1997, Three generative lexicalised models for statistical parsing. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics*, pp 16-23.

Dale, R., Moisl, H., and Somers, H., 2000, *Handbook of Natural Language Processing.* (Marcel Dekker Inc, New York).

Dyer, M., 1995, Connectionist Natural Language Processing: A Status Report. In R. Sun (ed.) *Computational Architectures Integrating Neural and Symbolic Processes*, pp 389-429, (Kluwer Academic Publishers, Boston).

Elman, J. L., 1990, Finding structure in time. *Cognitive Science*, **14**: 179-211.

Elman, J. L. 1991 Distributed representations, simple recurrent networks, and grammatical structure. *Machine Learning. 7*: 195-224.

Feldman, J. A., and Ballard, D. H., 1982, Connectionist models and their properties. *Cognitive Science*, **6**: 205-254.

Francis, W. N., and Ku era, H., 1979, *Manual of Information to Accompany a Standard Corpus of Present- day Edited American English.* Linguistics Department, Brown University.

Frazier, L., and Fodor, J., 1978, The sausage machine: a new two-stage parsing model. *Cognition*, **6**: 291-325.

Friederici, A. D., 2002, Towards a neural basis of auditory sentence processing. *Trends in Cognitive Sciences*, **6(2)**: 78-84. (Elsevier Science).

Garside, R., Leech, G., and Varadi, T., 1987, *Manual of Information to Accompany the Lancaster Parsed Corpus.* Department of English, University of Oslo.

Ghahramani, Z., and Allen, R., 1991, Temporal processing with connectionist networks. *IEEE Transactions on Neural Networks*, pp 541-546.

Gibson, E., 1991, *A Computational Theory of Human Linguistic Processing: Memory Limitations and Processing Breakdown.* Unpublished Ph.D Thesis, Carnegie-Mellon University.

Giles, C. L., Sun, G. Z., Chen, H. H., Lee, Y. C. and Chen, D., 1990. Higher order recurrent networks & grammatical inference. In Advance in Neural Information Processing Systems 2, (Morgan Kaufmann), pp 380-387.

Harrison, P., Abney, S., Black, E., Flickinger, D., Grishman, R., Gdaniec, C., Hindle, D., Ingria, R., Marcus, M., Santorini, B., and Strzalkowski, T., 1991, Evaluating Syntax Performance of Parser/Grammars of English. In *Proceedings of the Workshop on Evaluating Natural Language Processing Systems,* Association For Computational Linguistics, pp 71-78.

Henderson, J., 1994, Connectionist syntactic parsing using temporal synchrony variable binding. *Journal of Psycholinguistic Research*, **23(5)**: 353-379.

Henderson, J., and Lane, P., 1998, A connectionist architecture for learning to parse. In *Proceedings of COLING '98*, Montreal, Quebec, Canada, pp 531-537.

Hindle, D., 1993, A parser for text corpora. In B.T.S. Atkins and A. Zampolli (eds) *Computational Approaches to the Lexicon*, (Clarendon Press), pp 103-151.

Hindle, D., and Rooth, M., 1991, Structural ambiguity and lexical relations. In *Proceedings of the 29th Annual Meeting of the Association for Computational Linguistics*, Berkeley, California, pp 229-236.

Ho, E. K. S., and Chan, L. W., 1997, Confluent preorder parsing of deterministic grammars. *Connection. Science.* **9 (3)**: 269-293.

Johansson, S., Leech, G., and Goodluck, H., 1978, *Manual of Information to Accompany the Lancaster-Oslo/Bergen Corpus of British English.* Department of English, University of Oslo.

Kwasny, S. C., and Kalman, B. L., 1995, Tail-recursive distributed representations and simple recurrent networks. *Connection Science*, **7**: 61-80.

Lawrence, S., Giles, C. L., and Fong, S., 2000, Natural language grammatical inference with recurrent neural networks. *IEEE Transactions on Knowledge and Data Engineering*, **12(1)**: 126-140.

Leech, G., 1993, Corpus annotation schemes. *Literary and Linguistic Computing*, **8(4)**: 275-81.

Magerman, D. M., 1995, Statistical decision-tree models for parsing. In *Proceedings of the 33rd Annual Meeting of The Association for Computational Linguistics*, pp 276-283.

Marcus, M., Santorini, B., and Marcinkiewicz, M., 1998, Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics*, **19 (2)**: 313-330.

Mayberry, M. R., and Miikkulainen, R., 2000, Combining maps and distributed representations for shift-reduce parsing. In S. Wermter, and R. Sun (eds) *Hybrid Neural Systems*, (Springer Verlag, Heidelberg), pp 144-157.

McMahon, J., and Smith, F., J., 1998, A review of statistical language processing techniques. *Artificial Intelligence Review*, **12**: 347-391.

Miikkulainen, R., 1990, A PDP architecture for processing sentences with relative clauses. In *Proceedings of 13th International Conference on Computational Linguistics*, pp 201-206.

Miikkulainen, R., 1995, Subsymbolic parsing of embedded structures. In R. Sun (ed.) *Computational Architectures Integrating Neural and Symbolic Processes*, (Kluwer Academic Publishers, Boston), pp 153-186.

Palmer-Brown, D., Tepper, J. A., and Powell, H. M., 2002, Connectionist Natural Language Parsing. Review paper accepted for publication in *Trends in Cognitive Sciences*, (Elsevier Science).

Plate, T., 1995, Holographic reduced representations. *IEEE Transactions on Neural Networks*, **6 (3)**: 623-641.

Pollack J.B., 1990, Recursive Distributed Representations. *Artificial Intelligence*, **46**: 77-105.

Reggia, J., 1987, Properties of competition-based activation mechanism in neuromimetic network models. In *First International Conference on Neural Networks*, San Diego, pp 131-138.

Reilly, R., 1992, Connectionist technique for on-line parsing. *Neural Networks*, **3**: 37-46.

Rumelhart, D.E., Hinton, G.E., and Williams R. J., 1986, Learning internal representations by error propagation. In D. E. Rumelhart and J. L. McClelland (eds) *Parallel Distributed Processing, Explorations in the Microstructure of Cognition, Vol 1: Foundations*, (MIT Press), pp 318-362.

Sells, P., 1985, *Lectures on Contemporary Syntactic Theories,* Centre for the Study of Language and Information (CSLI), Stanford University.

Sharkey, N. E., and Sharkey, A. J. C., 1992, A modular design for connectionist parsing. In *Twente Workshop on Language Technology 3: Connectionism and Natural Language Processing*, University of Twente, Enschede, The Netherlands, pp 87-96.

Shastri, L., and Ajjanagadde, V., 1993, From simple associations to systematic reasoning: A connectionist representation of rules, variables and dynamic bindings using temporal synchrony. *Behavioral and Brain Sciences*, **16**: 417-194.

Shieber, S. M., 1983, Sentence disambiguation by a shift-reduce parsing technique. *Computer Speech and Language*, pp 297-323.

Smolensky, P., 1990a, Connectionism and the foundations of AI. *Foundations of Artificial Intelligence*, **11**: 1-74.

Smolensky P., 1990b, Tensor Product Variable Binding and the Representation of Symbolic Structures in Connectionist Systems. *Artificial Intelligence*, **46**: 159-216.

Stevenson, S., 1994, Competition and recency in a hybrid network model of syntactic disambiguation. *Journal of Psycholinguistic Research*, **23(4)**: 295-322.

Stevenson, S., and Merlo, P., 1997, Lexical structure and parsing complexity. *Language and Cognitive Processes*, **12(2/3):** 349-399.

Sun, R., 1992, On variable binding in connectionist networks. *Connection Science*, **4 (2)**: 93-124.

Tepper, J. A., Powell, H. M., and Palmer-Brown, D., 2001, Corpus-Based Connectionist Parsing. In *Proceedings of the Second Workshop on Natural Language Processing and Neural Networks*, National Center of Science, Tokyo, ISSN 1346-6682, pp 8-15.

Tepper, J. A., 2000, *Corpus-based Connectionist Parsing.* Unpublished Ph.D Thesis, The Nottingham Trent University, England.

Tepper, J. A., Powell, H., and Palmer-Brown, D., 1995a, Ambiguity resolution in a connectionist parser. In *Proceedings of the 4th International Conference on the Cognitive Science of Natural Language Processing*, Dublin City University.

Tepper, J. A., Powell, H., and Palmer-Brown, D., 1995b, Integrating symbolic and subsymbolic architectures for parsing arithmetic expressions and natural language sentences. In *Proceedings of the 3rd SNN Neural Network Symposium*, Nijmegen University, Netherlands.

Wermter, S., and Weber, V., 1994, Learning fault-tolerant speech parsing with SCREEN. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, Seattle, vol1, pp670-675, (AAAI Press/MIT Press).

Wermter, S., and Sun, R., 1999, *Hybrid Neural Systems,* (Springer Verlag, Heidelberg, New York).

Winograd, T., 1983, *Language as a Cognitive Process.* (Addison-Wesley, Reading, MA).

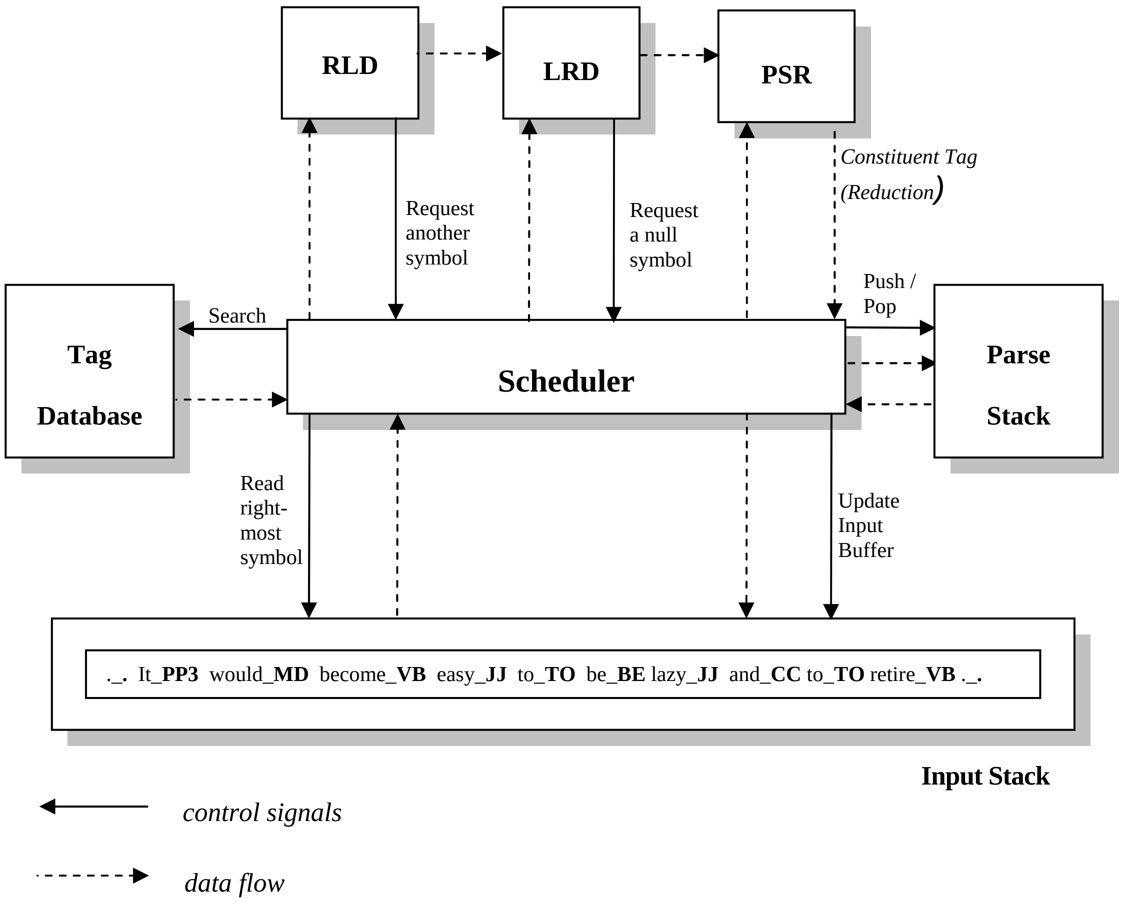
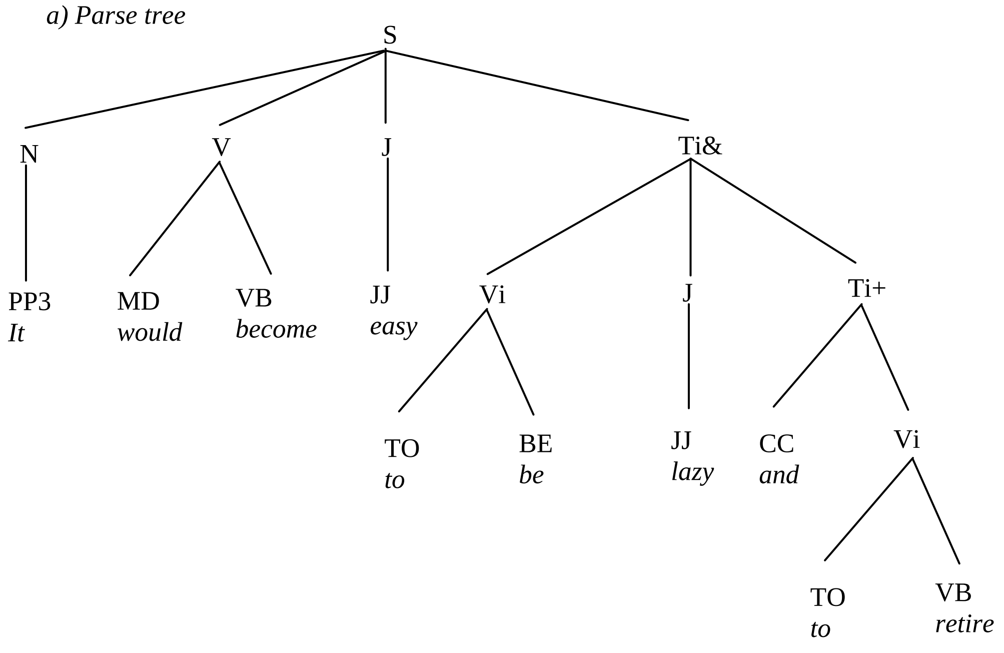


Figure 1. A schematic of the hybrid parsing architecture. Connectionist components consist of a Right-to-Left Delimiter (RLD), a Left-to-Right Delimiter (LRD) and a phrase structure recogniser (PSR) module. Symbolic components consist of the Tag database, Parse-stack and Input-stack[2].

[2] Temporary stacks used to hold data passing between the RLD and LRD modules, and between the RLD and PSR modules have been omitted for clarity.

*a) Parse tree*



*b) Labelled Bracketing*

[S[N it_PP3 N][V would_MD become_VB V][J easy_JJ J][Ti&[Vi to_TO be_BE Vi][J lazy_JJ J]
[Ti+ and_CC [Vi to_TO retire_VB Vi]Ti+]Ti&]S]

Figure 2. Example tagged and parsed sentence based on one from the Lancaster Parsed Corpus.

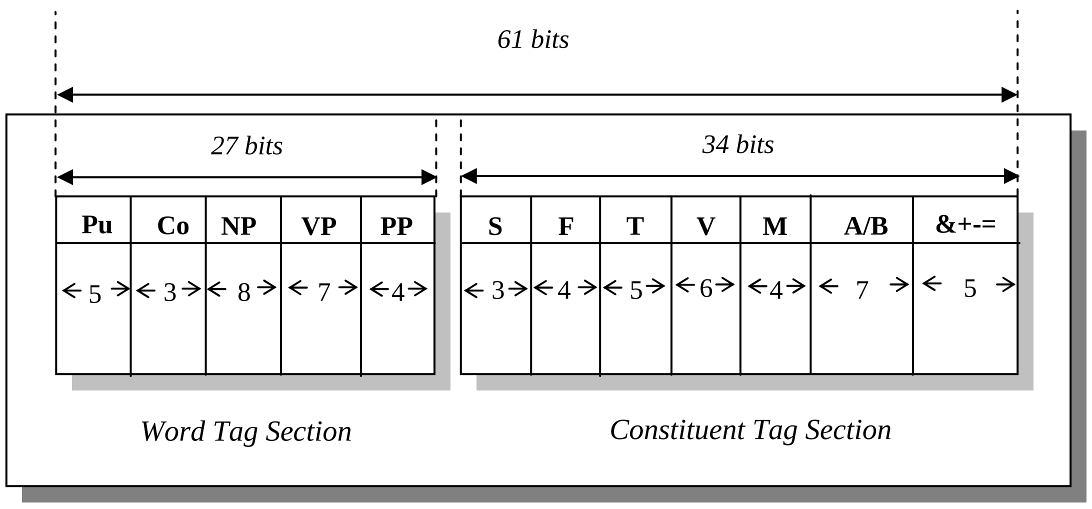


Figure 3. Word tag and constituent tag input representation.

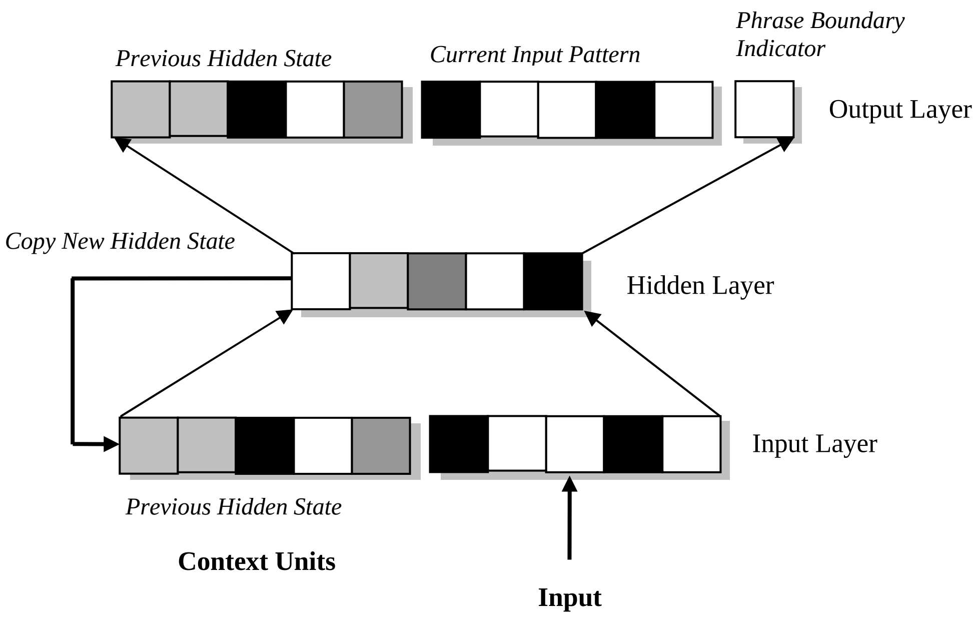


Figure 4. The TASRN architecture used for the LRD and RLD modules.

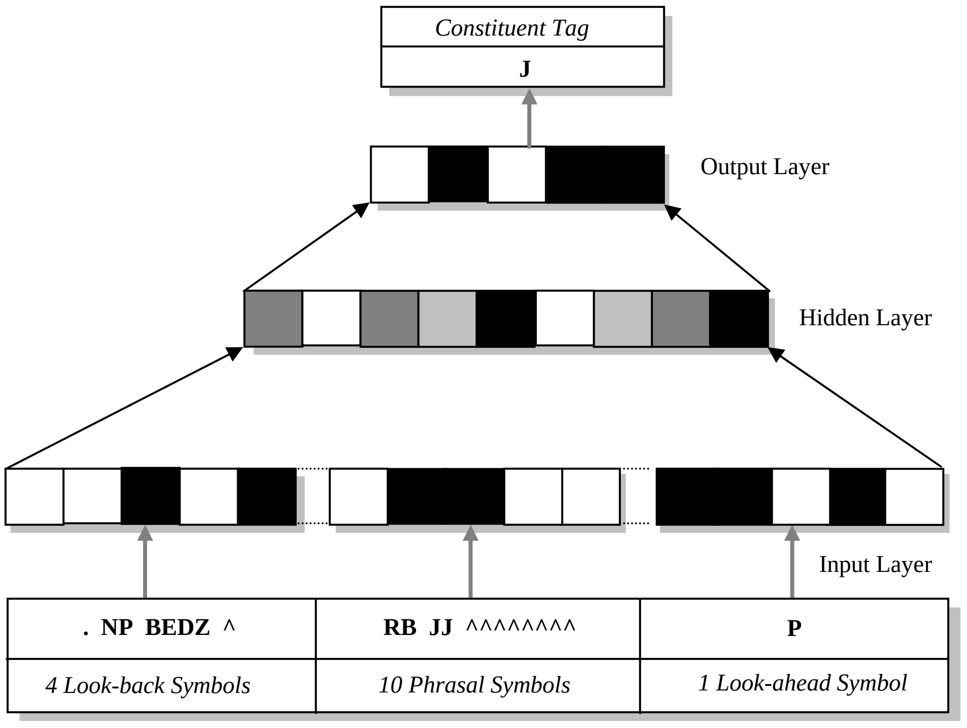


Figure 5. The FF-MLP architecture for the PSR module.

[S [N it_PP3 N] [R greatly_RB R] [V improves_VBZ V] [N& the_ATI appearance_NN [N+ and_CC the_ATI strength_NN N+] N&] S]

Figure 6. The matching parse for *it greatly improves the appearance and the strength*.

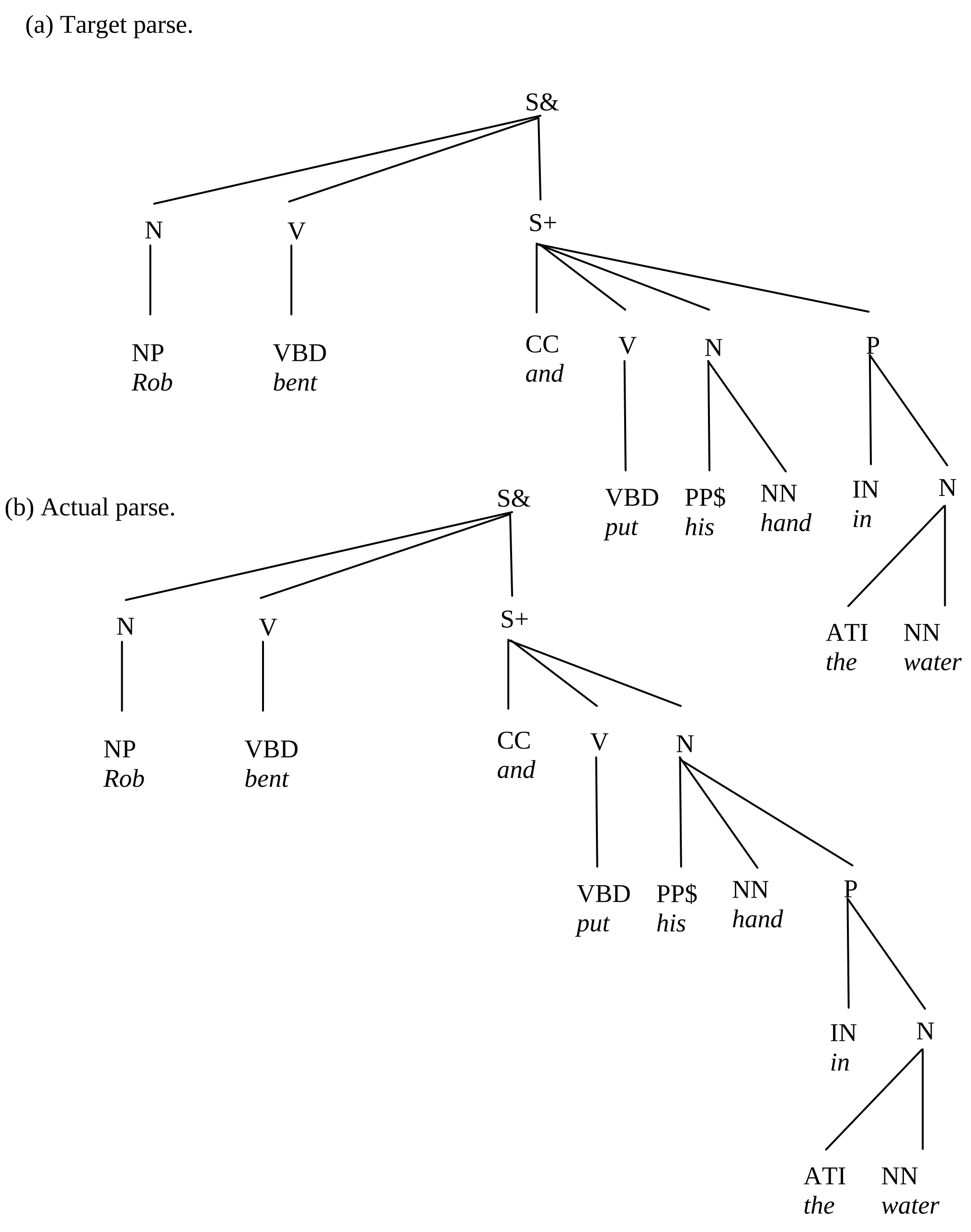


Figure 7. (a) The target parse for *Rob bent and put his hand in the water*. (b) The parser's actual parse for the sentence.

| **A Parse Example** | | | |
|---|---|---|---|
| **Stage** | **Module** | **Input Symbol(s) Indicated** | **Determined to be:** |
| 1 | RLD | to_TO (before retire_VB) | Start of Phrase (SOF) |
| 2 | LRD | retire_VB | End of Phrase (EOF) |
| 3 | PSR | to_TO retire_VB | reduced to Vi |
| 4 | RLD | and_CC | SOF |
| 5 | LRD | Vi | EOF |
| 6 | PSR | and_CC Vi | reduced to Ti& |
| 7 | RLD | lazy_JJ | SOF |
| 8 | LRD | lazy_JJ | EOF |
| 9 | PSR | lazy_JJ | reduced to J |
| 10 | RLD | to_TO (before be_BE) | SOF |
| 11 | LRD | be_BE | EOF |
| 12 | PSR | to_TO be_BE | reduced to Vi |
| 13 | RLD | Vi | SOF |
| 14 | LRD | Ti+ | EOF |
| 15 | PSR | Vi J Ti+ | reduced to Ti& |
| 16 | RLD | easy_JJ | SOF |
| 17 | LRD | easy_JJ | EOF |
| 18 | PSR | easy_JJ | reduced to J |
| 19 | RLD | would_MD | SOF |
| 20 | LRD | become_VB | EOF |
| 21 | PSR | would_MD become_VB | reduced to V |
| 22 | RLD | it_PP3 | SOF |
| 23 | LRD | it_PP3 | EOF |
| 24 | PSR | it_PP3 | reduced to N |
| 25 | RLD | N | SOF |
| 26 | LRD | Ti& | EOF |
| 27 | PSR | N V T Ti& | reduced to S |

Table 1. Individual module indications in the processing sequence for the sentence in Fig. 2

| Delimiter Training Sequences | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | **Length 5 Sequences** | **Length 6 Sequences** | **Length 7 Sequences** | **Length 8 Sequences** | **Length 9 Sequences** | **Total Sequences** | **Total Patterns** |
| **RLD** | **Raw** | - | 277 | 1,015 | 933 | 584 | 2,809 | 21,487 |
| | **Balanced** | - | 1,015 | 1,015 | 1,015 | 1,015 | 4,060 | 30,450 |
| **LRD** | **Raw** | 1,017 | 938 | 417 | 202 | - | 2,574 | 15,248 |
| | **Balanced** | 1,017 | 1,017 | 1,017 | 1,017 | - | 4,068 | 26,442 |

Table 2. The composition of the raw and balanced delimiter training sets.

| Delimiter Training Results | | |
|---|---|---|
| | **RLD** | **LRD** |
| Hidden units for >90% sequences learnt | 150 | 65 |
| Upper limit imposed on No. hidden units | 165 | 115 |
| Hidden units adopted (give lowest RMSE within imposed limit) | 165 | 110 |
| No. Weight connections | 75,137 | 38,012 |
| Lowest RMSE | 0.07 | 0.057 |
| % unique training sequences learnt | 91 | 94 |

Table 3. Overall Delimiter Training Results.

| Delimiter Test Sample | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | **Length 5 Sequences** | **Length 6 Sequences** | **Length 7 Sequences** | **Length 8 Sequences** | **Length 9 Sequences** | **Total Sequences** | **Total Patterns** |
| **RLD** | **Natural** | - | 322 | 1,049 | 971 | 662 | 3,004 | 23,001 |
| | **Pure** | - | 285 (89%) | 935 (89%) | 867 (89%) | 576 (87%) | 2,663 (89%) | 20,375 (89%) |
| **LRD** | **Natural** | 1,087 | 979 | 443 | 233 | - | 2,742 | 16,274 |
| | **Pure** | 881 (81%) | 832 (85%) | 387 (87%) | 192 (82%) | - | 2,292 (84%) | 13,642 (84%) |

Table 4. Composition of the Delimiter Test Sample.

| Delimiter Generalisation Results - % Correct | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | **Length 5 sequences** | **Length 6 sequences** | **Length 7 sequences** | **Length 8 sequences** | **Length 9 sequences** | **Total Patterns** | **Total Sequences** |
| **RLD** | **Natural** | - | 86.65 | 86.46 | 83.93 | 91.84 | 97.92 | 86.85 |
| | **Pure** | - | 84.91 | 84.92 | 82.35 | 90.80 | 97.68 | 85.35 |
| **LRD** | **Natural** | 96.96 | 85.19 | 90.52 | 97.42 | - | 98.33 | 91.76 |
| | **Pure** | 96.59 | 83.05 | 89.15 | 96.88 | - | 98.07 | 90.45 |

Table 5. Natural and pure generalisation results for the RLD and LRD networks.

| Generalisation Results for The Test Sample | | | | | |
|---|---|---|---|---|---|
| | **Number of Patterns** | **RMSE Value** | **Correct Classifications** | **Correct Classifications after 'Nearest Match' Computation** | **% Generalisation** |
| **Natural Test Patterns** | 2,765 | 0.0683 | 2,190 | 2,461 | 89.01 |
| **Pure Test Patterns** | 2,433 | 0.0726 | 1,861 | 2,132 | 87.63 |

Table 6. The natural and pure generalisation results for the PSR with 50 hidden units.

| Sentence Level Results | | | | | | |
|---|---|---|---|---|---|---|
| | **Sentences** | **Words** | **Parsed** | **Exact Matches** | **Labelled Precision** | **Labelled Recall** |
| **Training Set** | 654 | 3,846 | 96.9% | 62.2% | 81.7% | 76.1% |
| **Test Set** | 687 | 4,128 | 93.4% | 49.0% | 75.1% | 68.9% |

Table 7. Sentence Level Training and Test Results