TRAINING A RECURRENT NEURAL NETWORK TO PARSE SYNTACTICALLY AMBIGUOUS AND ILL–FORMED SENTENCES

Ssu–Liang Lin and Von–Wun Soo
Department of Computer Science
National Tsing–Hua University, Hsin–Chu, Taiwan, 30043.

ABSTRACT

We are investigating to what extent can neural networks learn to parse a natural language. In particular, we present a recurrent neural network architecture and the learning experiments used to train the neural network. We train the recurrent neural network using the extended error backpropagation method by giving a sequence of lexicons as input whose categories may be ambiguous (more than one category is possible). Instead of encoding the parse tree within the neural network, the correct phrasal links as well as the lexical categories are clamped at the output layer of the network at the training phase while lexical categories are being fed into the neural network. With phrasal links, however, a complete parse tree can be easily reconstructed. Our results indicate that with a few training examples, the neural network can parse not only syntactically ambiguous sentences but also some ill–formed sentences that it has never seen before.

1. Introduction

Parsing is an important step in natural language processing. The main function of parsing is to produce the structural relationships among lexicons from a given input sentence. Traditional syntactic parsing methods such as chart–parsing, WASP (Wait–And–See parser), ATN (Augmented Transition Network), etc. [1,7], were somewhat successful in parsing well–formed sentences. However, they all encountered the difficulty in dealing with high complexity of ambiguities and potentially ill–formed sentences. This can be due to the reason that traditional methods are so restricted in their accuracy constraints that they do not accept any noise in their input. Nevertheless, in natural language processing, ambiguity and ill–formness are ubiquitous and unavoidable. New parsing techniques seem to be desirable to overcome these problems. This motivates us to look for more flexible parsing models that can achieve high efficiency and adaptability.

Artificial neural networks have recently raised much attention in its capabilities of carrying out computation of parallel constraint satisfaction and learning[6,9,10]. From a problem solving standpoint, parsing can be viewed as a constraint satisfaction process which must reconcile with constraints coming from both data (bottom–up from lexicons) and models (top–down from syntactic grammars). Therefore, training a neural network to perform parsing can be viewed as a process of incrementally encoding the structural relationships between lexicons and grammar rules in the inter–connections of the neural network. In this paper, we propose a system called SPARK (Syntactic PArser with Recurrent neural networK) to show the process of training a recurrent neural network to parse a subset of natural language from a context–free grammar. In section 2, we briefly summarize previous neural network approaches dealing with the problems of natural language parsing. In section 3, we describe the architecture of SPARK and the extended error backpropagation method that it adopts to achieve learning. In section 4, we explain the learning experiments that were used to train SPARK in order to make it acquire syntactic parsing skills. In section 5, we show the performance of the trained neural network by testing several different cases and discuss their implications. In section 6, we give our conclusions, discuss the limitations of SPARK and future work.

2. Previous work

Fanty [3] proposed a connectionist model which used neural networks to parse an English sentence in terms of a sequence of syntactic categories. His approach is to embed all possible parse trees in the neural network by pre–encoding a huge number of all possible phrasal links (which he called matching units). His parsing model can only handle sentences with a fixed number of lexicons. The disadvantage of this model is that the number of interconnections can be quite large for even a simple grammar. Santos [11] proposed a system called PALS which used a seven–by–six matrix of cells to represent the partial structure of a parse tree. Each cell in the matrix consists of all possible phrasal nodes. PALS needs additional rule nodes to represent the linking relationships between constituents in two adjacent cells of the same column in the matrix. PALS used the idea of snapshots with a size of seven constituents to break a long sentence into several chuncks. The disadvantages of PALS, however, are two folds: (1) its learning ability is locally restricted which can be difficult when handling embedding clauses and (2) its built–in rules prohibit the possibility of rule acquisition. Giles [4] trained a second–order single–layer recurrent neural network to recognize the languages produced by a regular grammar and a pushdown automata. Although Giles's work is not related with parsing a language, using activation patterns of context units in a recurrent network to represent the transition states seems to reflect a similar situation encountered in learning language parsing where intermediate parsing statuses are often needed to be trained and retained for subsequent parsing processes. St. John and McClelland [12] trained recurrent neural networks to achieve semantic comprehension of a natural language. Their neural network consisted of two stages of processing: a Gestalt–pattern building process which accumulates the syntactic and lexical information of a given input sentence and a role–filling process which assigns semantic roles to the corresponding constituents based on the Gestalt–patterns. The Gestalt–patterns can achieve the expectation and prediction on the semantic roles for the incoming constituents at a certain parsing context and situation. Other works such as Cottrel [2], Jain et. al. [5], McClelland et. al.[8] and Wermter [14] also discussed several different proposals and techniques to apply neural networks to problems at different levels of language parsing. We were motivated by previous work and decided to choose the recurrent neural network model to investigate the natural language parsing problems.

3. The Architecture of SPARK and The Extended Error Backpropagation Learning Method

SPARK's architecture consists of four units: the input units, output units, hidden units and the context units which are shown in Fig. 3.1. The input layer for this feedforward network consists of 8 category input units (CIU), i. e. "noun","pronoun", "aux", "verb–i", "verb–t", "det", "adj" and "prep". Each unit represents a syntactic category of the input lexicon. We assume that all lexicons in a sentence have been assigned to their corresponding categories before the sentence is given to the recurrent neural network. For example, the sentence "The young boy will go with her" would be converted to its categorial form "det adj noun aux verb–i prep pronoun". When a given input lexicon has more than one syntactic category, more than one unit can be activated. When the first word category, say "det", is read in, the weight of the det–CIU will be set to 0.7 and all weights of the rest of CIU's will be set to 0.2. When a syntactic ambiguous lexicon is read in, more than one CIU's will be set to 0.7 depending on the corresponding categories of the lexicon. The output layer of the recurrent neural network also has 8 category output units (COU) which correspond to CIU's

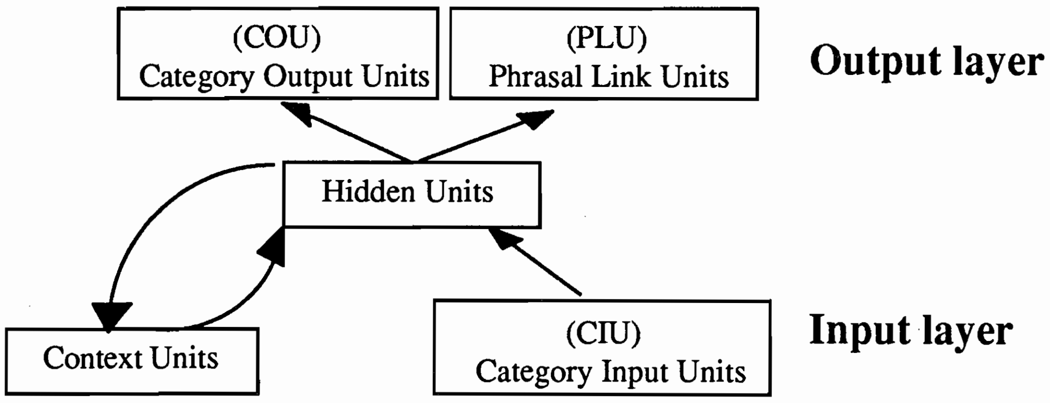


Fig. 3.1 The structure of the recurrent neural network

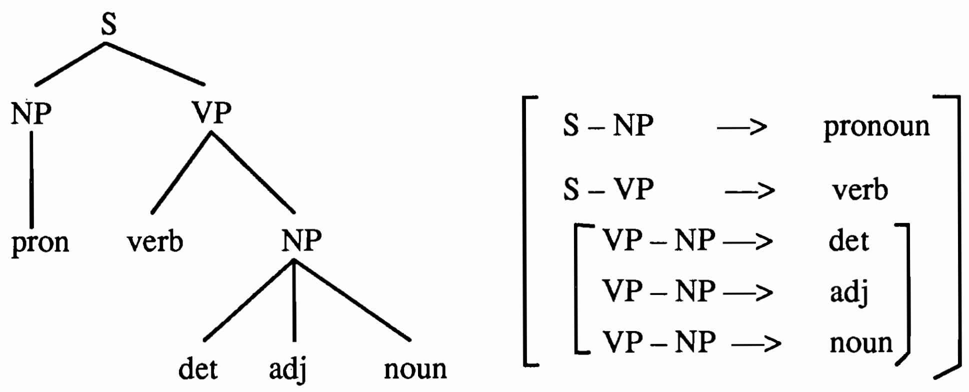


Fig. 3.2 The parsing tree and its corresponding phrasal link representation

as well as 6 phrasal link units (PLU), i. e., "S–NP" (S–N), "S–VP" (S–V), "VP–NP" (V–N),"VP–PP" (V–P),"PP–NP" (P–N) and "NP–PP" (N–P), each of which represents a piece of partial parse–tree information. An parse tree can be represented in terms of these phrasal links. For example, in Fig. 3.2, a parse tree of a sentence with five lexicons on the left can be represented by five phrasal links on the right and each of the link corresponds to a lexicon. The number of the hidden units and that of the context units are the same and can be varied.

The extended error backpropagation differs from conventional error backpropagation of Rumelhart in that there is a feedback connections from hidden units to context units [14]. This feedback mechanism in SPARK is to temporalily store the current parsing status in terms of activation patterns of hidden units into the context units so that the subsequent parsing step can take it into account. The learning algorithm for training SPARK can be derived by unfolding the temporal sequence of feedforward passes into a multi–layer feedforward network that grows one layer at each pass as shown at the right hand side of Fig. 3.3. To carry out the extended error backpropagation learning procedure, we let first the network run through the time interval [t_0, t] and save all inputs, activation patterns of hidden units, and target vectors at each time step into a history buffer. Then the temporal error backpropagation according to the history proceeds. The process is described in terms of a set of equations that are defined in Fig. 3.3. First, we define the error generated over time as E(t) which

$$(1)\quad e_k(t) = d_k(t) - y_k(t) \qquad k \in U$$

$$(2)\quad E(t) = \frac{1}{2} \Sigma [\, e_k(t)]^2$$

$$(3)\quad E^{total}(t_0,t) = \sum_{\tau = t_0+1}^{t} E(\tau)$$

$$(4)\quad \nabla_w E^{total}(t_0,t) = \sum_{\tau = t_0+1} \nabla_w \; E(\tau)$$

$$(5)\quad \Delta W_{ji} = -\eta \frac{\partial E^{total}(t_0,t)}{\partial W_{ji}}$$

$$(6)\quad \underset{k \in H}{\delta_k(\tau)} = f_k'[s_k(\tau)][\sum_{j \in U} W_{jk} \cdot \delta_j(\tau) + \sum_{l \in I} W_{kl} \cdot \delta_k(\tau+1)]$$

$$(7)\quad \Delta W_{ji} = -\eta \sum_{\tau = t_0+1}^{t} \delta_j(\tau) \cdot x_i(\tau - 1)$$

$s_k(\tau)$: the net input for k–th hidden units at time τ

U : the index set for output units

I : the index set for external inputs

H : the index set of hidden units

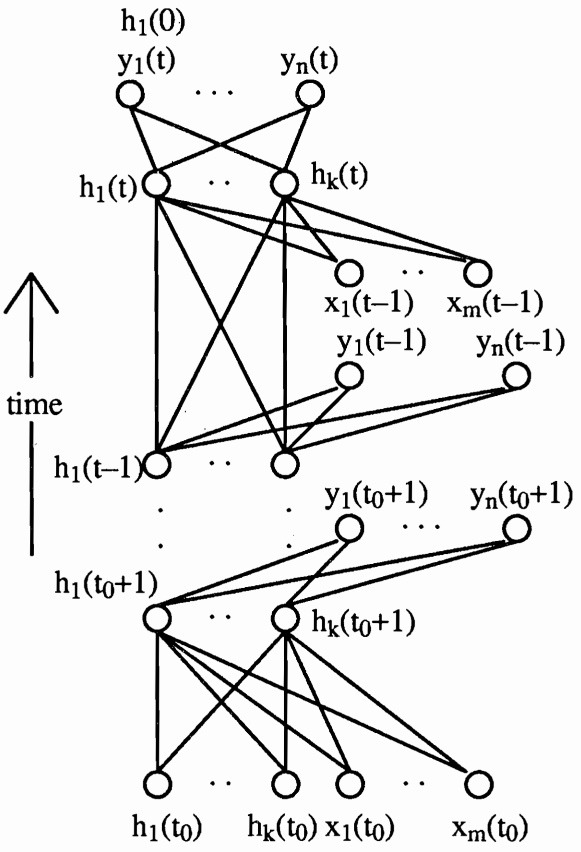


Fig. 3.3 Equations used in the extended error backpropagation and the unfolded neural network architecture for a temporal sequence

is a sum of square of the difference between desired target $d_k(t)$ and output $y_k(t)$ [eq. (1) and (2)]. The learning goal of SPARK is to minimize the total error function $E^{total}(t_0, t)$ [eq. (3)]. The weight updating formula is computed by taking the negative gradient of $E^{total}(t_0, t)$ with respect to weights [eq. (4) & (5)]. Since the errors are propagated through the whole time sequence, the error at hidden layer must take into account both errors from the output layer at current time step and those from subsequent step [eq. (6)]. Once the temporal error backpropagation has been propagated to the time t_0+1, the actual connection weights can then be updated by the generalized delta rule [eq. (7)].

4. The Training Experiments

In this section, we show how to train a recurrent neural network to acquire the parsing skills given a set of training sentences. We create training sentences according to a set of context–free grammar rules as shown in Table 4–1. Note that the (adj)* represents that the number of adjectives can vary from zero to several while (aux) represents that an auxiliary is optional.

Table 4.1 A set of phrase structure rules

S	←	NP VP			
NP	←	det (adj)* noun PP	VP	←	(aux) verb–t NP
NP	←	pronoun	VP	←	(aux) verb–t NP PP
NP	←	det (adj)* noun	VP	←	(aux) verb–i PP

We tentatively prepare two training sets S1 and S2. S1 consists of 16 sentences shown in Table 4.2 whose lexicon categories are all uniquely assigned while the number of the input lexicons ranges from 2 to 11. Training set S2 includes S1 and has additional 9 sentences shown in Table 4.3 whose lexical categories can be ambiguous. For example, "n/v" represents a word whose category can be either a noun, an intransitive verb (vi) or a transitive verb (vt). In the second columns of Table 4.2 and Table 4.3, the corresponding partial parse–tree information in terms of phrasal links for each sentence is also shown.

Table 4.2 Training set 1

Training sentences with lexicon categories	partial parse tree information (phrasal links)
1. det noun vi *The child laughed.*	S–N S–N S–V
2. det noun vt det noun *The girl saw every movie.*	S–N S–N S–V V–N V–N
3. det noun vi prep det noun *The baby cried in the bedroom.*	S–N S–N S–V V–P P–N P–N

4. det noun prep det noun vi *The stranger with a hat disappeared.*	S–N S–N N–P P–N S–V
5. det noun prep det noun vt det noun *The girl with the unbrella broke her leg.*	S–N S–N N–P P–N P–N S–V V–N V–N
6. det noun prep pron vt det noun prep det noun *The man over there drank some wine in the afternoon.*	S–N S–N N–P P–N S–V V–N V–N V–P P–N P–N
7. pron vi *He succeeded.*	S–N S–V
8. pron vt pron *I like you,*	S–N S–V V–N
9. det adj noun vt det adj noun *A young girl found this little cat.*	S–N S–N S–N S–V V–N V–N V–N
10. det adj noun prep det adj noun vt det adj noun *The old man with the wooden stick discovered a new life.*	S–N S–N S–N N–P P–N P–N P–N S–V V–N V–N V–N
11. pron vt pron prep pron *He saw her with me.*	S–N S–V V–N V–P P–N
12. pron vi prep pron *He sat over there.*	S–N S–V V–P P–N
13. det noun prep pron aux vi *The clerk over there will manage.*	S–N S–N N–P P–N S–V S–V
14. det noun aux vi *Their dog won't bite.*	S–N S–N S–V S–V
15. det noun aux vt det noun *The students must read this textbook.*	S–N S–N S–V S–V V–N V–N
16. det noun prep det noun aux vt det noun *The students in the classroom must took an examination.*	S–N S–N N–P P–N S–V S–V V–N V–N

In Table 4.4 and 4.5, we show the effects of learning speed (in terms of epochs) versus various number of hidden/context units. It can be seen that when the number of hidden units reach around 30's, the efficiency of learning seems to become stable. Since the number of the hidden/context units might slightly affect the performance, for comparing the testing results in this paper, however, we kept the number at 10.

Table 4.3 Additional sentences with ambiguous lexicon categories for Training set 2

Training sentences with lexicon categories	partial parse tree information (phrase links)
1. det n/v vi *The program halted.*	S–N S–N S–V
2. det noun vi/n *The animals escaped.*	S–N S–N S–V
3. det noun vt det n/v *The boy caught one fish.*	S–N S–N S–V V–N V–N
4. det noun vt/n det noun *The tanks attacked the city.*	S–N S–N S–V V–N V–N
5. det noun vi/n prep det noun *His wife worked in the company.*	S–N S–N S–V V–P P–N P–N
6. det noun prep det n/v vi *The book with no cover disappeared.*	S–N S–N N–P P–N P–N S–V
7. pron vi/n *He danced.*	S–N S–V
8. pron vt/n pron *She helped me.*	S–N S–V S–N
9. det adj/v noun vt det adj n/v *Her close friend told a funny joke.*	S–N S–N S–N S–V V–N V–N V–N
10. det n/v aux vt/n det n/v *This report may influence his score.*	S–N S–N S–V S–V V–N V–N

Table 4.4 The convergence rates in terms of number of training epochs vs number of hidden units for Training set S1. The threshold is set at 0.7 for total sum of square error.

No.of hidden units	**8**	**10**	**15**	**20**	**30**	**40**	**50**
Epochs	**>3000**	**575**	**187**	**84**	**62**	**56**	**56**

Table 4.5 The convergence rates in terms of number of training epochs vs number of hidden units for Training set S2. The threshold is set at 1.0 for total sum of square error.

No.of hidden units	**10**	**15**	**20**	**30**	**40**	**50**	**60**
Epochs	**814**	**158**	**117**	**86**	**79**	**78**	**56**

5. Testing Results

After training, we used several testing sentences to evaluate the performance of SPARK.We found that SPARK could successfully parse many sentences with ambiguous categories. In particular, SPARK can tolerate syntactic ill–formed sentences and can produce a plausible parsing structure to account for a given input sentence. Since it is impossible to explore all possible legal and illegal sentences to evaluate the performance of SPARK, we show only a few testing cases to explain how SPARK performs parsing. This will be discussed from three different aspects in (A), (B) and (C) respectively.

(A) Testing Results Using New and Syntactic Ambiguous Sentences

SPARK can parse successfully those sentences with ambiguous categories which it has never seen before. Although SPARK can handle many sentences of this kind, we only illustrate one example in detail. In Fig. 5. 1, we show the activation patterns which were taken from run–time execution results for sentence ”det adj/v n/v aux vi prep det n/v”.

We found that the ambiguous categories for three lexicons were assigned correctly and the corresponding parse tree directly constructed from the PLU patterns shown in Fig. 5.2. Similarly, for the

	COU								PLU					
	n	pn	ax	vi	vt	d	a	p	S–N	S–V	V–N	V–P	P–N	N–P
1	.	.	.	.	.	#	.	.	#	.	.	.	.	.
2	.	.	.	.	.	.	#	.	#	.	.	.	.	.
3	#	.	.	.	.	.	.	.	#	.	.	.	.	.
4	.	.	#	.	.	.	.	.	.	#	.	.	.	.
5	.	.	.	#	.	.	.	.	.	#	.	.	.	.
6	.	.	.	.	.	.	.	#	.	.	.	#	.	.
7	.	.	.	.	.	#	.	.	.	.	.	.	#	.
8	#	.	.	.	.	.	.	.	.	.	.	.	#	.

Fig. 5.1 The activation patterns for the sentence ”det adj/v n/v aux vi prep det n/v”, the # represents the activation values ≥ 0.8 and the . represents values ≤ 0.2.

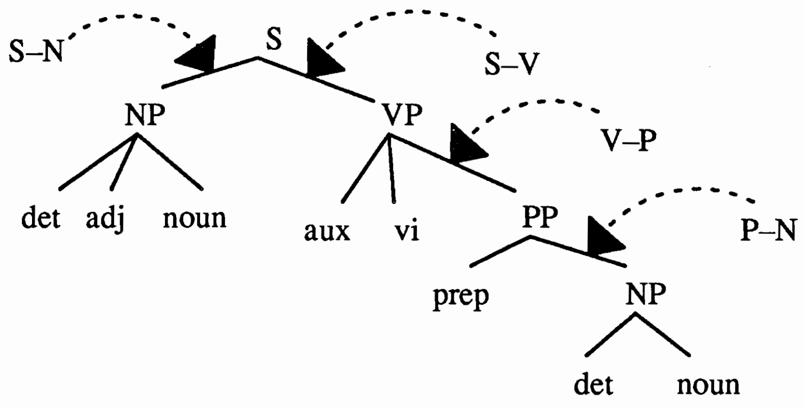


Fig. 5.2 The corresponding parse tree for the sentence "det adj/v n/v aux vi prep det n/v". The dashed arrows point to the corresponding phrasal links.

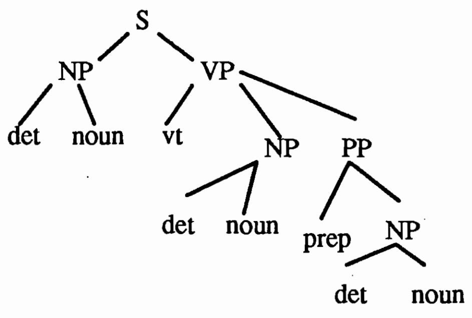
sentence with more categorial ambiguities like "det adj/v n/v vt/n det adj/v n/v", SPARK also produced the correct category assignment "det adj noun vt det adj noun" as well as a correct parsing tree. For a longer sentence such as "det adj/v adj adj n/v prep det adj adj noun vi prep det adj noun" SPARK also performed well and produced "det adj adj adj n prep det adj adj noun vi prep det adj noun" as a result as we desired.

(B) Testing Results Using Sentences with Syntactic Noises

In this experiment, we show that SPARK can tolerate some syntactic ill–formed sentences. In Fig. 5.3, we illustrate three sentences and their activation patterns of the output units generated by SPARK. The first sentence is a well–formed sentence with respect to the context–free grammar in Table 4.1, while the second and the third sentences are ill–formed. In the second sentence, a noun follows a determiner is tentatively omitted, SPARK can still produce the plausible parse tree with an expectation for a noun after the determiner. In the third sentence, both a noun and a preposition are omitted, the most plausible parse tree shows an expectation of a noun after the determiner, however with a erroneous preprositional link which shows up as indicated in the right bottom parse tree in Fig. 5.3.

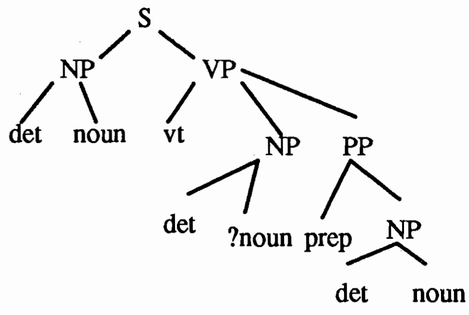
1– det noun vt det noun prep det noun

	n	pn	ax	vi	vt	d	a	p	S–N	S–V	V–N	V–P	P–N	N–P
1	.	.	.	.	.	#	.	.	#	.	.	.	.	.
2	#	.	.	.	.	.	.	.	#	.	.	.	.	.
3	.	.	.	.	#	.	.	.	.	#	.	.	.	.
4	.	.	.	.	.	#	.	.	.	.	#	.	.	.
5	#	.	.	.	.	.	.	.	.	.	#	.	.	.
6	.	.	.	.	.	.	.	#	.	.	.	#	.	.
7	.	.	.	.	.	#	.	.	.	.	.	.	#	.
8	#	.	.	.	.	.	.	.	.	.	.	.	#	.



2– det noun vt det prep det noun

1	.	.	.	.	.	#	.	.	#	.	.	.	.	.
2	#	.	.	.	.	.	.	.	#	.	.	.	.	.
3	.	.	.	.	#	.	.	.	.	#	.	.	.	.
4	.	.	.	.	.	#	.	.	.	.	#	.	.	.
5	=	.	.	.	.	.	.	#	.	.	.	#	.	.
6	.	.	.	.	.	#	.	.	.	.	.	.	#	.
7	#	.	.	.	.	.	.	.	.	.	.	.	#	.



3– det noun vt det det noun

1	.	.	.	.	.	#	.	.	#	.	.	.	.	.
2	#	.	.	.	.	.	.	.	#	.	.	.	.	.
3	.	.	.	.	#	.	.	.	.	#	.	.	.	.
4	.	.	.	.	.	#	.	.	.	.	#	.	.	.
5	*	.	.	.	.	#	.	.	.	.	=	*	,	.
6	#	.	.	.	.	.	.	.	.	.	.	.	#	.

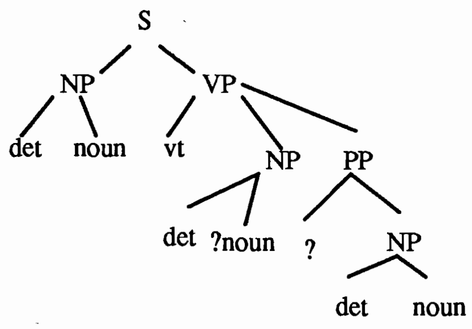


Fig. 5.3 ## 2 and ## 3 are examples of syntactic ill–formed sentences derived from sentence ## 1. The notations for "#", "*", "=", ",", and "·" represent ranges of activation values [0.8,1], [0.6, 0.8), [0.4, 0.6), [0.2, 0.4), and [0, 0.2) respectively. On the right hand side, the corresponding parse trees are shown. The ? in the parse tree represents the expectation of a constituent reflected in the activation patterns of COU.

(C) The Limitations and Problematic Cases

Of course, when there is too much noise involved, SPARK might fail too. However, we could argue that for certain bad data even human experts might be confused too. SPARK would maintain those that were successfully parsed and only fail at places where troubles got in. Here we illustrate a case to explain how SPARK might fail. For example, the sentence illustrated in Fig. 5.4 has "n/v" ambiguities at three places. As we can see in Fig. 5.4, SPARK handled well on the first entry of

```
## 15    det n/v n/v prep det n/v

        n  pn  ax  vi  vt  d  a  p        S–N   S–V   V–N   V–P   P–N   N–P
   1    .   .   .   .   .  #  .  .         #     .     .     .     .     .
   2    #   .   .   .   .  .  .  .         #     .     .     .     .     .
   3    .   .   .   *   #  .  .  .         .     #     .     .     .     .
   4    .   .   .   .   .  .  ,  #         .     .     ,     .     .     .
   5    .   .   .   .   .  #  .  .         .     .     =     .     *     .
   6    #   .   .   .   .  .  .  .         .     .     ,     .     *     .
```

Fig. 5.4 A case where SPARK cannot produce a complete parse tree.

"n/v" (row 2)which it predicted as a noun. However, SPARK hesitated as for whether the second entry of "n/v" (row 3) was a transitive or an intransitive verb. This influenced the next entry when a lexical category "prep" was entered (row 4), SPARK got loss and could not predict any plausible phrasal link (no high enough activation values for PLU in row 4).

6. Discussions and Conlcusions

We have demonstrated a way of using recurrent neural network to perform syntatic parsing. Although we cannot claim that current SPARK can outperform traditional parsing methods, we do show the potentials of the approach. The advantages of SPARK are its ability to cope with ambiguities and ill–formness and its ability of learning (without explicitly specifying the grammar rules). Using only a few training sentences, we have obtained a plausible parser to parse many sentences that are generated by a context–free grammar. There are several extensions that can further enhance SPARK to become a truly natural language parser. First, the classification of categories and the phrasal links can be further elaborated. Second, semantic features or case roles can be included in the training in order to resolve those ambiguities such as a prepositional phrase attachment problem.

Reference

[1] Allen, James (1987). Natural Language Understanding, Benjamin/Cunmmings.

[2] Cottrel, G. W. (1989). A Connectionist Approach to Word Sense Disambiguation, Morgan Kauf mann Publishers.

[3] Fanty, Mark (1985). Context Free Parsing in Connectionist Networks, Technical Report 96, Computer Science Department, University of Rochester.

[4] Giles, C. L., Sun, G. Z., Chen, H. H., Lee, Y. C. & Chen, D. (1990). Higher Order Recurrent Net works & Grammatical Inference, In D.S. Tourestzky (ed), Advances in Neural Information Sys tems 2, Morgan Kaufmann, pp. 381–387.

[5] Jain, A. N. & Waibel, A H. (1990). Incremental Parsing by Modular Recurrent Connectionist Networks, In Tourestzky, D.S. (ed), Advances in Neural Information Systems 2, Morgan Kauf mann, pp. 364–371.

[6] Khanna, Tarun (1990). Foundations of Neural Networks, Addison–Wesley.

[7] Marcus, Mitchell P. (1980). A Theory of Syntactic Recognition for Natural Language, MIT Press.

[8] McClelland, J. L. & Kawamoto, A. H. (1986). Mechanisms of Sentence Processing: Assigning Roles to Constituents, In J.L. McClelland, D.E. Rumelhart & the PDP Research Group (eds), Par allel Distributed Processing: Explorations in the Microstructure of Cognition vol–II: Applications, MIT Press, pp. 272–325.

[9] Pao, Yoh–Han (1989). Adaptive Pattern Recognition and Neural Networks, Addison–Wesley.

[10] Rumelhart, D. E., Hinton, G. E. & Williams, R. J. (1986). Learning Internal Representations by Error Propagation, In J.L. McClelland, D.E. Rumelhart & the PDP Research Group (eds), Parallel distributed processing: Explorations in the microstructure of cognition 1: Foundations , MIT Press.

[11] Santos Jr, E. (1989). A Massively Parallel Self–Tuning Context–Free Parser, In D.S. Tourestzky (ed), Advances in Neural Information Systems 1,Morgan Kaufmann, pp. 537–544.

[12] St. John, M. F. & McClelland, J. L. (1990). Learning and Applying Contextual Constraints in Sentence Comprehension, Artificial Intelligence 46, pp. 217–257.

[13] Wermter, S. (1989). Integration of Semantic and Syntactic Constraints for Structural Noun Phrase Disambiguation, Proceeding of the 11th IJCAI, vol. 2, pp. 1486–1491.

[14] Williams, R. J. and Peng, I. (1990). An Efficient Gradient–based Algorithm for On–line Train ing of Recurrent Network Trajectories, Neural Computation 1, pp. 490–501.