Classification of Natural Language Sentences using Neural Networks

Sergio Roa and **Fernando Nino**
National University of Colombia
Department of Computer Science
Ciudad Universitaria
Bogota, D.C., Colombia
s.roa@computer.org
lfnino@ing.unal.edu.co

Abstract

In this work the task of classifying natural language sentences using recurrent neural networks is considered. The goal is the classification of the sentences as grammatical or ungrammatical. An acceptable classification percentage was achieved, using encoded natural language sentences as examples to train a recurrent neural network. This encoding is based on the linguistic theory of Government and Binding. The behaviour of the recurrent neural network as a dynamical system is analyzed to extract finite automata that represent in some way the grammar of the language. A classifier system was developed to reach these goals, using the Backpropagation Through Time algorithm to train the neural net. The clustering algorithm Growing Neural Gas was used in the extraction of automata.

Introduction

Neural networks have been widely used in classification tasks and, in particular, multilayer feedforward networks are well-known and broadly implemented. However, in the case of grammars, it may be verified that recurrent neural networks have an inherent ability to simulate finite state automata (Haykin 1999), from which grammars of regular languages are inferred. The behaviour of a recurrent neural net (RNN) as a dynamical system can be analyzed (Haykin 1999) to construct a finite state automaton (Giles *et al.* 1992; Omlin & Giles 1996; Lawrence, Giles, & Fong 2000). However, regarding the natural language processing, it must be noted that grammars of natural languages cannot be completely represented by finite state models, due to their hierarchical structures (Pereira & Shabes 1992). Nevertheless, it has been shown that recurrent networks have the representational power required for hierarchical solutions (Elman 1991), and therefore, a type of RNN for natural language processing, called Elman network, is studied in this work.

This article describes the use of a RNN to classify natural language encoded sentences by their grammatical status (grammatical or ungrammatical), using the encoding assumed by the Government and Binding theory of syntax (GB-theory) (Chomsky 1981). The objective of the RNN is to produce the same judgements as native speakers on the grammatical/ungrammatical pairs, and to infer some representation of the grammar, expecting that the neural network learns in some way grammatical features. In recent years, diverse architectures of RNNs have been developed and used to extract automata for grammatical inference (Giles *et al.* 1992). In this work, based on the research of Lawrence, Giles, & Fong, the capacity of the Elman recurrent neural network (Elman 1991) to classify correctly natural language sentences is shown. The Elman network has been used previously in some natural language processing tasks (Stolcke 1990) and is used here, because of the good results found in its training (Lawrence, Giles, & Fong 2000). An implementation of the Backpropagation through time (BPTT) algorithm, specifically adapted to the problem, was developed and used to train the network, achieving an improved optimization of the algorithm convergence with respect to previous works (Lawrence, Giles, & Fong 2000; Roa & Nino 2001). Finally, the behaviour of the net as a dynamical system was analyzed to extract the knowledge acquired by the Elman neural net, in this case finite automata that represent some of the syntactic structures found in the language. The clustering algorithm called Growing Neural Gas network (Fritzke 1997) was used to reach these goals, obtaining improved results compared to anterior works.

The rest of this article is organized as follows. First, the design of the classifier system is described, i.e., the Elman network, its topology and the training algorithm used. Then, the results of this classification are presented. Finally, the automata extraction process is explained in detail.

Design of the classifier system

The input data are encoded sentences of the English language. These examples were taken from the investigation described on the article of Lawrence and others (2000). The neural network was trained using the same examples, expecting the same judgements as native speakers on the grammatical/ungrammatical pairs. Consequently, without having any knowledge about the components assumed by the linguistic theory described above, in this case positive and negative examples are used, trying to exhibit the same kind of discriminatory power that the linguists have found using the GB-theory (Lawrence, Giles, & Fong 2000).

The GB-theory assumes four primary lexical categories, which are verbs (v), nouns (n), adjectives (a) and prepositions (p). The other four classes are complementizer (c), determiner (det), adverb (adv) and marker (mrkr). Besides this categorization, a subcategorization of the classes is also defined, i.e., the categories are subcategorized depending on the context. Following the GB-theory and using a specific parser, the subcategorization was made by Lawrence, Giles, & Fong to obtain the encoded examples used in this work. For example, an intransitive verb, such as *sleep*, would be placed into a different class from the transitive verb *hit*. Similarly, verbs that take sentential complements or double objects, such as *seem*, *give* or *persuade*, would be representative of other classes. Hence, the encoding resulted in 8 classes, categorized as follows: 9 classes of verbs, 4 of nouns, 4 of adjectives, 2 of prepositions. The remaining classes do not have subcategorization. Table 1 shows some examples of sentences, their respective encoding and grammatical status, where 1 means grammatically correct and 0 incorrect.

Sentence	Encoding	Grammatical Status
I am eager for John to be here	n4 v2 a2 c n4 v2 adv	1
	n4 v2 a2 c n4 p1 v2 adv	1
I am eager John to be here	n4 v2 a2 n4 v2 adv	0
	n4 v2 a2 n4 p1 v2 adv	0
I am eager to be here	n4 v2 a2 v2 adv	1
	n4 v2 a2 p1 v2 adv	1

Table 1: Encoded sentences and grammatical status.

As shown in Table 1, some sentences have different encodings, because the parser was developed in a free-context manner (Lawrence, Giles, & Fong 2000). Some sentences presented contradictory results, i.e., different grammatical states for the same sentence, thereby these examples were eliminated. The encoded sentences are processed using a parser and encoded again to be presented properly to the neural net. A normalized encoding for the subcategories of the 4 principal categories is used. For example, the subcategories for nouns would be: Not a noun = 0, noun class 1 = 0.5, noun class 2 = 0.667, noun class 3 = 0.833, noun class four = 1. The remaining categories, which do not have subcategorization, have a value of 1. This encoding was defined using a linear order, depending on similarities between subcategories.

The output data or desired responses are originally 0 or 1 (grammatically correct or incorrect), but values of 0.9 and 0.1 were given to the network, to avoid the saturation of the activation function of each neuron (Lawrence, Giles, & Fong 2000). 365 training examples were presented, equally distributed in correct and incorrect examples.

Topology of the recurrent neural network

In this work, the Elman recurrent neural network (Elman 1991), shown in Figure 1, was used. The data were input to the neural net using a window of size 2, through which sentences are entered in blocks of each two words. Each window consists of 8 neurons, corresponding with the 8 lexical categories, i.e., 16 neurons overall. The value received by each neuron is the corresponding one for the subcategory or 0.0 if the neuron does not correspond with those category. For example, the input of the encoded sentence "n1 v1" would correspond to 0.5 for the first neuron, that represents nouns, and 0.0 for the next seven neurons of the respective window, 0.0 for the first neuron in the second window, 0.5 for the second neuron corresponding with verbs, and 0.0 for the rest.

The use of a window is relevant because permits the temporal processing of each sentence, allowing the extraction of information acquired by the net as dynamical system. In this case, some finite automata were extracted. For the case of a window of size 2, transitions of two words in the constructed automaton may be predicted, although in this work the use of a window of size 1 was also investigated, reaching good results in the convergence of the algorithm and the extraction of automata with one-word transitions. The use of a large window (input window equal to the longest sentence) was also investigated, without observing the behaviour expected as dynamical system, because the network does not have to store any information (states and rules) during the processing of the sentence; it simply maps the inputs directly to the classification. Figure 1 shows this process and the network topology.

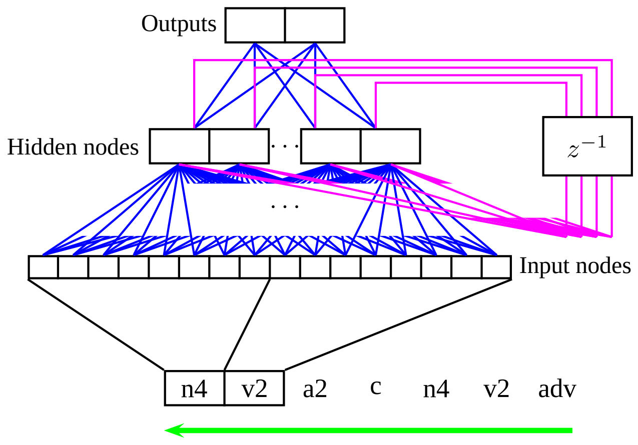

Figure 1: Use of the window in the Elman recurrent network

The architecture of the net consists of 16 neurons in the input layer for a window of size 2, 20 in the hidden layer and 2 in the output layer, using bias inputs. This topology presented the best results for the convergence of the algorithm. The first neuron in the output layer has a desired response of 0.9 if the sentence is correct and 0.1 otherwise. The second neuron gets the desired responses of 0.1 and 0.9 respectively. Thus, these neurons classify the sentences as grammatical or ungrammatical.

Network training

An adapted version of the BPTT algorithm (Williams & Peng 1990; Haykin 1999) was implemented. This algorithm is commonly used to train recurrent networks and is an extension of the standard algorithm. It is derived by unfolding the temporal operation of the net into a feedforward network, holding the synaptic weights. The algorithm presented in this article is based on the Truncated BPTT and the Epochwise BPTT algorithms (Williams & Peng 1990). Let $T(n)$ be the set of indices j of neurons for which exists a desired response d_j such that the output y_j of the j-th neuron should have at time n. Then, the error signal is defined by:

$$e_j(n) = \begin{cases} d_j(n) - y_j(n) & \text{if } j \in T(n) \\ 0 & \text{otherwise.} \end{cases} \quad (1)$$

The total error is described as follows:

$$E(n) = \frac{1}{2} \sum_j e_j^2(n). \quad (2)$$

The learning goal is the minimization of the total error over some appropriate period of time $[n_0, n_1]$, defined by

$$E_{\text{total}}(n_0, n_1) = \sum_{n=n_0}^{n_1} E(n). \quad (3)$$

For this particular task, the period $[n_0, n_1]$ is represented by a temporal pattern, i.e., the input of a whole sentence, where n_0 is the initial time step of processing (first word[1]) and n_1 is the final time step (last word(s)). Therefore, the topology of the net at the end of the forward propagation can be viewed as a multilayer feedforward network, with as many hidden layers as the magnitude of the interval $[n_0, n_1]$. This process can be observed in Figure 2, where $\vec{x}$ is the input vector, $\vec{w}$ is the weight vector, $\vec{y}$ is the output vector and there are 3 steps of processing (e.g., a sentence of three words using a window of size 1).

The backpropagation step is performed when the current sentence is processed throughout. Let $T(n)$, $H(n_1)$ and $H(n)$ be the set of indices of neurons in the output layer, in the last unfolded hidden layer and in a hidden layer, respectively. The local gradients are defined as

$$\delta_j(n) = \begin{cases} \varphi_j'(v_j(n))e_j(n) & \text{if } n = n_1 \text{ and } j \in T(n) \\ \varphi_j'(v_j(n)) \sum_k w_{kj}\delta_k(n) & \text{if } n = n_1, j \in H(n_1) \\ & \text{and } k \in T(n) \\ \varphi_j'(v_j(n)) \sum_k w_{kj}\delta_k(n+1) & \text{if } n_0 \leqslant n < n_1, \\ & j \in H(n), k \in H(n+1) \end{cases} \quad (4)$$

where $\varphi_j'(v_j(n))$ is the derivative of the activation function at time n for neuron j, given the activation potential v_j. In this case, the logistic function was used. It can be observed

[1]Or 2 words for the case of a window of size 2

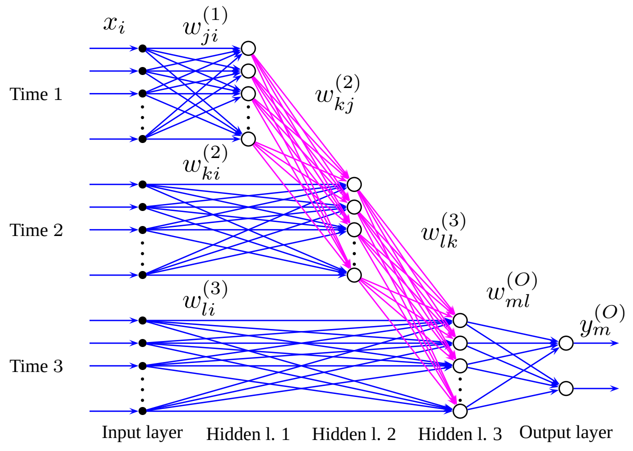

Figure 2: Forward propagation in the BPTT algorithm.

that the backpropagation starts in the output layer and continues until the first layer, only taking into account the unfolded layers for the current sentence. Let $I(n)$ denote the set of indices for the neurons in the input layer. Then, the weight updates are calculated as follows:

$$\Delta w_{ji} = \begin{cases} \eta \sum_{n=n_0}^{n_1} \delta_j(n)x_i(n) & j \in H(n), i \in I(n) \\ \eta \sum_{n=n_0+1}^{n_1} \delta_j(n)y_i(n-1) & j \in H(n), i \in H(n-1) \\ \eta\delta_j(n)y_i(n) & j \in T(n), i \in H(n_1) \\ & \text{and } n = n_1. \end{cases} \quad (5)$$

The equation 5 shows that, for the feedback connection (second case), the gradient at n_0 is not considered, because the output $y_i(n-1)$ corresponds with the processing of the previous sentence. Likewise, for the last case, i.e., the connection between the hidden layer and the output layer, only the local gradient for the output layer is considered. This may be explained because of the processing of the output layer just at time n_1.

The average total error was used as performance measure at the current training epoch:

$$E_{\text{avg}} = \frac{1}{N} \sum_{t=1}^{N} E_{\text{total}}(t) \quad (6)$$

for the t-th training example from N in total. The learning rate η at time t is adjusted using a *search-then-converge* schedule of the form $\frac{\eta_0}{1+\frac{t}{\tau}}$, where $\eta_0 = 0.2$ is the initial learning rate and τ is a constant. This schedule is used to avoid parameter fluctuation through a training cycle, preventing a decrease in the network training performance (Haykin 1999; Lawrence, Giles, & Fong 2000). The stochastic update[2] was used and, in addition, the training examples were randomly presented, avoiding the convergence to local minima (Haykin 1999). The weights were initialized

[2]Each pattern (the whole sentence) is presented and the backpropagation is executed immediately

using uniformly distributed random numbers in the range $[-2, 2]$, which presented better results.

Experimental results

Acceptable results were found in the classification of sentences. 365 examples were used in the training phase, 184 grammatical and 181 ungrammatical. The results in correct classification percentage and final error (Table 2) were very similar using windows of size 1, 2 and 15 (the longest sentence size). The values were taken after 4000 epochs of training. The same analysis was performed for the test set,

Window	Classification	Error
1	100%	0.00013
2	100%	0.00019
15	100%	0.00031

Table 2: Results over the training set.

consisting of 135 sentences (96 correct and 39 incorrect). The results are shown in Table 3, also for both correct and incorrect examples.

Window	Percentage of correct classification		
	Grammatical ex.	Ungrammatical ex.	Total
1	59.38%	69.23%	62.22%
2	56.25%	82.05%	63.70%
15	56.25%	79.49%	62.96%

Table 3: Results over the test set.

These results prove that the recurrent network has the ability to learn grammatical structures involved in its neurodynamics. However, the results for the test set were inferior, because the training examples possibly do not represent much of the syntactic structures found in the English language. It may be concluded that the recurrent network learns some representation or part of the grammar, due to the small proportion of training examples used in this investigation[3]; additionally, the separation into the training and test sets can create a test set which lacks some grammatical structures.

Automata Extraction

The knowledge acquired by a RNN may be extracted, analyzing the net as a dynamical system. The ordered triple of a discrete Markov process ({state; input (word(s)) $\rightarrow$ next state}) can be extracted from a recurrent network, constructing equivalent finite automata which represent some of the language features. This can be done by clustering the activation values of the recurrent state neurons (Omlin & Giles 1996). In this case, the state of an Elman network is represented by the hidden layer neurons (Haykin 1999). However, it is firmly established (Chomsky 1956) that the syntactic structures of natural language cannot be parsimoniously described by regular languages. Certain phenomena are more compactly described by context-free grammars, recognized by push-down automata, or context-sensitive grammars, recognized by linear bounded automata (Pollack 1991). Some researches, for example, have dealt with the problem of analyzing a RNN as a chaotic dynamical system, studying the fractal attractors found in its neurodynamics, trying to understand how RNNs can represent more than a regular language and how they can learn a complex set of grammatical rules (Pollack 1991).

Once the states and transitions are extracted from the RNN (propagating the set of sentences), a clustering algorithm was used to categorize these states into different classes, searching a reduction of dimensionality and expecting that the recurrent network quantize its state space during training. The algorithm used in this case is the Growing Neural Gas network (Fritzke 1997). It is an incremental network model which is able to learn important topological relations in a given set of input vectors, in this case the state vectors, by means of a simple Hebb-like learning rule. The main idea of this method is to successively add new units (neurons) to an initially small network by evaluating local statistical measures gathered during previous adaptation steps. The network topology is generated incrementally by Competitive Hebbian learning (Martinetz 1993): at each adaptation step a connection between the winner and the second-nearest unit is created. The resulting graph is a subgraph of the Delaunay triangulation corresponding with the set of reference vectors of the neurons. This subgraph, which is called the induced Delaunay triangulation, is limited to those areas of the input space where data are found. The induced Delaunay triangulation has been shown to optimally preserve topology in a very general sense (Martinetz 1993). Furthermore, since the reference vectors are adapted to the input space, a mechanism to remove edges that are not valid is needed. This is done by an aging mechanism. The algorithm ends depending on a given maximal count of neurons or a minimal quantization (or distortion) error (Fritzke 1997).

When the clustering phase is finished, automata are constructed using the reference vectors of the winner neurons, which categorize the extracted states of the RNN, by storing the new ordered triple {corresponding reference vector of a state; input (word(s)) $\rightarrow$ corresponding reference vector of the next state}. In general, non-deterministic finite state automata were extracted. In order to obtain more information, automata were extracted using both training set and complete set. The results for the case of one-word transitions (use of a window of size 1) and two-word transitions are shown in Table 4 and in Table 5, respectively. Columns 3 and 4 show the percentages of correct classification. In addition, these tables show the number of states extracted, the set of examples used to extract automata, and the final quantization errors. When the complete set is used to construct an automaton, a better classification for both training and test sets is obtained, because the automaton includes more knowledge.

[3]The dataset has been hand-designed by GB linguists (Lawrence, Giles, & Fong 2000)

Nr. of states	Used set	Classification		Error
		Training set	Test set	
57	Training	81.42%	47.79%	0.15
57	All	81.69%	69.12%	0.17
94	Training	88.25%	47.06%	0.09
94	All	86.89%	69.12%	0.09
173	Training	91.53%	41.91%	0.03
181	All	89.62%	60.29%	0.04

Table 4: Results (six automata with one-word transitions).

Nr. of states	Used set	Classification		Error
		Training set	Test set	
57	Training	96.45%	44.12%	0.09
58	All	95.36%	63.24%	0.10
87	Training	97.27%	43.38%	0.03
89	All	96.45%	63.24%	0.04
143	Training	98.36%	41.91%	0.002
150	All	96.99%	63.24%	0.007

Table 5: Results (six automata with two-word transitions).

Conclusions

In this work, recurrent neural networks were used to classify natural language sentences. The goal was the classification of the sentences as grammatical or ungrammatical. Once the neural network was trained, automata that represented in some sense the grammar of the language recognized by the neural network were extracted. The results demonstrated acceptable performance of the extracted automata, very close to 100% for the case of two-word transitions. These results were similar to those obtained by the respective RNNs, proving the capacity of RNNs to simulate finite state automata. Using a higher level of quantization, better results were found. Therefore, it was also shown the ability of the Growing Neural Gas algorithm to discover properly statistical features found in the state space of this RNN. Additionally, it may be deduced that automata represent some of the syntactic characteristics, although it is known that they cannot broadly represent the structures found in natural language. It is observed that automata with two-word transitions presented better results in classification, though automata with one-word transitions could possibly describe more effectively the grammar. This could be related to the complexity of the clustering process. Furthermore, analyzing the recurrent net as a chaotic dynamical system might help to understand its ability to learn complex languages, in order to obtain better solutions.

In general, the results proved the capacity of RNNs to learn at least some of the grammatical structures found in the natural language, considering a correct classification percentage of 100% in the training examples both for the case of a window of size 1 and size 2 when the improved learning algorithm was developed, specially due to a correct chosing of the weight adjustments and the process of unfolding the temporal operation of the net. However, the results in the test set were inferior, possibly due to the lack of representative structures found in the training set. It is expected a better generalization using more examples, but perhaps an increased difficulty in training. Furthermore, this problem cannot be treated as a common pattern classification task, considering the way in which sentences are presented to the network, using a temporal processing. RNNs can successfully represent a complex set of syntactic structures if they are able to incorporate internal representations of the data.

References

Chomsky, N. 1956. Three models for the description of language. *IRE transactions on information theory* 2(3):113–124.

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

Elman, J. 1991. Distributed Representations, Simple Recurrent Networks, and Grammatical Structure. *Machine Learning* 7(2/3):195–226.

Fritzke, B. 1997. Some competitive learning methods. Technical report, Institute for Neural Computation, Ruhr-Universitaet Bochum.

Giles, C.; Miller, C.; Chen, D.; Sun, G.; Chen, H.; and Lee, Y. 1992. Extracting and Learning an Unknown Grammar with Recurrent Neural Networks. In Moody, J.; Hanson, S.; and Lippmann, R., eds., *Advances in Neural Information Processing Systems 4*. San Mateo, Calif.: Morgan Kaufmann. 317–324.

Haykin, S. 1999. *Neural Networks: A Comprehensive Foundation*. Upper Saddle River, N.J.: Prentice Hall.

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.

Martinetz, T. 1993. Competitive Hebbian learning rule forms perfectly topology preserving maps. In *ICANN'93: International Conference on Artificial Neural Networks*. Amsterdam: Springer. 427–434.

Omlin, C., and Giles, C. 1996. Extraction of Rules from Discrete-Time Recurrent Neural Networks. *Neural Networks* 9(1):41–52.

Pereira, F., and Shabes, Y. 1992. Inside-Outside Re-Estimation from Partially Bracketed Corpora. In *Proceedings of the 30th Annual Meeting of the Association for Computational Linguistics*, 128–135.

Pollack, J. 1991. The Induction of Dynamical Recognizers. *Machine Learning* 7:227–252.

Roa, S., and Nino, L. 2001. Clasificacion de Frases del Lenguaje Natural. In Hernandez, J., ed., *Congreso Internacional en Inteligencia Computacional*. Medellin, Col.: Universidad Nacional de Colombia, Sede Medellin. 147–150.

Stolcke, A. 1990. Learning Feature-Based Semantics with Simple Recurrent Networks. Technical Report TR-90-015, International Computer Science Institute, University of California at Berkeley.

Williams, R., and Peng, J. 1990. An Efficient Gradient-Based Algorithm for On-Line Training of Recurrent Network Trajectories. *Neural Computation* 2(4):490–501.