Automatic Adaptation of a Natural Language Interface to a Robotic System

Ramón P. Ñeco, Óscar Reinoso, José M. Azorín,
José M. Sabater, and M. Asunción Vicente

Dpto. Ingeniería, Miguel Hernández University, Avda. Ferrocarril s/n, 03202 Elche (Alicante), Spain ramon.neco@umh.es
http://lorca.umh.es

Abstract. This paper shows an application of four neural networks architectures for the automatic adaptation of the voice interface to a robotic system. These architectures are flexible enough to allow a non-specialist user to train the interface to recognize the syntax of new commands to the teleoperated environment. The system has been tested in a real experimental robotic system applied to perform simple assembly tasks, and the experiments have shown that the networks are robust and efficient for the trained tasks.

1 Introduction

Learning of natural language can be divided in two different tasks: (1) learning of the syntax and (2) learning of the semantics. In the case of learning the syntax, the objective is to extract a set of grammatical rules, starting from a sequence of examples of sentences grammatically correct (and possibly of sentences grammatically incorrect). In the case of the semantics, the objective is to obtain an association between the emitted commands and the corresponding internal representation of those commands (intermediate language to a robot in the experiments presented in this paper).

The automatic learning of the syntax, also known as *grammatical inference*, has been studied from the theoretical point of view in numerous works [1], [2]. There exists two sets of learning techniques: symbolic and connectionist. The symbolic techniques try to obtain the grammatical rules directly from each learning example, while the connectionist technique obtains the grammar's model as a neural network. Most works that learn the semantics try to learn the meanings of new words from examples, in the symbolic paradigm [3], but some authors have also developed neural networks solutions [4]. In this paper, we show how neural networks architectures can be used to adapt the grammar of a natural language interface to a robotic system.

2 The system

As in all teleoperated robotic applications, the experimental system considered here consist of a remote environment, as well as a local environment that control

Fig. 1. Teleoperated system used for the experiments on natural language.

and supervise the remote environment. The devices that interact with the task have been located in the remote area (figure 1). The elements of the remote environment are the following: A robotic arm (Mitsubishi PA-10) of 7 degrees of freedom, that executes the commands emitted by the operator; a computer that acts as the robot controller; a computer for image processing; wide range area cameras; and a camera located at the end of the robotic arm, to obtain more precise visual information in the manipulation of the objects on the part of the robot.

In the local environment all the elements such that the operator can interact to send and to receive the commands to the remote environment can be found. These elements are the following: graphic computation system, by means of which the operator knows in real time the state of the task and can control in a complete way the remote system; a master device; and a computer for speech recognition that make the speech recognition together with the natural language processing so that the operator can command the teleoperated system using voice commands [5].

2.1 Formal Grammars and Grammatical Inference

We give in this section a short introduction to formal grammars, grammatical inference and natural language. For a more detailed description we recommend, for example, [6]. A grammar G is a four-tuple (N, T, P, S) where N and P are sets of terminals and nonterminals symbols comprising the alphabet of the grammar, P is a set of production rules, and S is the start symbol of the grammar. The language $L(G)$ associated to this grammar is the set of strings of the terminal symbols that the grammar recognizes. We define *grammatical inference* as the procedures that can be used to obtain the production rules of an unknown grammar G (the *syntax*) based on a finite set of strings of $L(G)$ (and possibly also a finite subset of the complement of $L(G)$). In this paper we apply grammatical inference using neural networks in order to learn the syntax of new commands in a natural language interface to a teleoperated robot.

Natural language processing has traditionally been handled using symbolic methods and recursive processes. The most used of these symbolic methods are based on finite-state descriptions such as n-grams or hidden Markov models. However, finite-state models cannot represent hierarchical structures as found in natural language commands to a robot. Recurrent and feedforward neural networks have been used for several small natural language problems [7], [8]. Also, in speech recognition some neural network models have been used to account for a variety of phenomena in phonology, morphology and role assignment [8], [9]. The main motivation for the work presented in this paper was the fact that a natural interface to a robot needs to be flexible enough to allow the users to adapt the underlying system grammar.

Some authors have addressed the problem of induction of simple grammars - e.g. [10] - and there has been some interest in learning more than regular grammars with recurrent neural networks, such as recursive auto-associative memories (RAAMs) [11] or recurrent neural networks tied to external trainable stacks [12]. In all these works the grammars learned were not large, while other authors such as [13] tried to learn considerably more complex grammars. However, in the last case the obtained grammars where not intuitively interpretable from a logical point of view. In the practical applications presented in this paper, the phrases to be analyzed are not too large so we expected that a connectionist architecture can learn the syntax of new commands not included initially in the designed interface.

In the next sections the results obtained in the training of two categories of recurrent neural architectures for grammatical inference in the robotic application will be described. Section 3 describes the first category of experiments, performed using simple recurrent networks. In the second category of experiments, described in section 4, a combination of recurrent nets will be used in order to obtain a "neural" interpretation of the presented command.

3 Simple Recurrent Networks

This section describes the application of three recurrent neural network architectures : (1) totally connected recurrent network [14]; (2) Elman recurrent network [15]; and (3) Back-Tsoi network [16].

3.1 Network Architectures

Totally Connected Network The architecture of this network consists of three layers, shown in Figure 2 (left): One input layer in which the code of the input sentence is introduced; one hidden layer that represents the internal state of the network; and one output layer where the network stores the result, which is the feedback toward the nodes in the hidden layer. The output of node k in the output layer is given by the following expression:

$$y_k = f\left(\sum_{y=0}^{N_l-1} w_{ki} h_i\right) \tag{1}$$

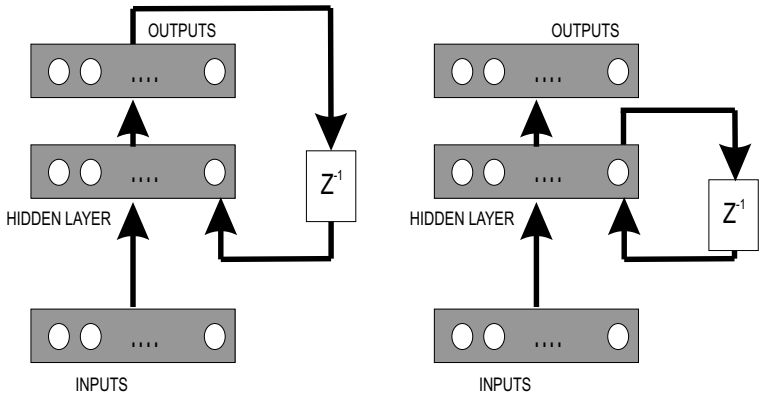

Fig. 2. Totally connected and Elman networks.

where f is a sigmoid function, N_l is the number of nodes of the hidden layer, w_{ki} is the weight of the connection between the node i of the hidden layer and the node k in the output layer, and h_i is the activation of the node i of the hidden layer which in turn is computed according to the following expression:

$$h_k = f\left(\sum_{i=0}^{N_e-1} w_{ki}e_i + \sum_{i=0}^{N_s-1} v_{ki}s_i\right) \tag{2}$$

where e_i is the input corresponding to node i in the input layer, s_i is the activation of node i in the output layer, w_{ki} is the value of the connection between the node i in the input layer to the node k in the hidden layer, and v_{ki} is the value of the connection between node i in the output layer to node k in the hidden layer.

Elman Recurrent Network The second network architecture considered for the learning of the syntax is an Elman recurrent network, also known as simple recurrent net (SRN) [15]. This network has feedback from each node in the hidden layer toward all the nodes of the same layer. In Figure 2 an scheme of this architecture is shown.

The dynamics of this net is given by the following expression:

$$s_k = f\left(\sum_{l=1}^{D} w_{kl}e_l + \sum_{l=0}^{H} v_{kl}s_l(t-1)\right) \tag{3}$$

where s_k is the activation of the node k in the hidden layer, v_{kl} is the value of the connection between the node l of the input layer and the node k in the hidden layer, v_{kl} is the value of the recursive connection between the node 4 of the hidden layer and the node k of the hidden layer, and $s_l(t-1)$ is the value

of the node l of the hidden layer at the previous discrete time step. Starting from s_k at time t, the output of the net is obtained according to the following expression:

$$y_p = f\left(\sum_{l=0}^{H} v_{lp}^{(2)} s_l(t)\right) \tag{4}$$

where v_{lp} is the value of the connection between the node l of the hidden layer and the node p of the output layer, and $s_p(t)$ is the value of the node p of the hidden layer at the current time step t. The error measure used is the quadratic error, given by the expression

$$E(t_1, t_2) = \sum_{t=t_1}^{t=t_2} E(t) = \sum_{t=t_1}^{t=t_2} \frac{1}{2} \sum_{p=1}^{M} \left(d_p(t) - y_p(t)\right)^2 \tag{5}$$

Back-Tsoi Network The third architecture used in the experiments is the FIR net of Back-Tsoi [16] (FIR, Finite-duration Impulse Response). The basic idea of this architecture is the inclusion of a FIR and a gain term in each synapsis. The net has L processing layers, excluding the input layer, without feedback. In the layer l there exists N_l nodes. The output of the node k in layer l at the time step t, $k = 1, 2, \cdots, N_l, l = 0, 1, \cdots, L$, is given by the following expression:

$$y_k^l(t) = f\left(x_k^l(t)\right) \tag{6}$$

where f is a sigmoid function, computed according to the expression:

$$x_k^l(t) = \sum_{i=0}^{N_l - 1} c_{ki}^l(t) \sum_{j=0}^{N_b} w_{kij}^l(t) y_i^{l-1}(t-j) \tag{7}$$

3.2 Training and Experiments

The experiments with these three architectures used sets of 124 training commands, positives and negatives. All these commands have been or imperative sentences or declarative sentences describing situations of the robot's environment. The training commands are shown to each one of the nets one to one, applying the two learning algorithms that are described next for each time of training. The learning is considered successful when one of the following conditions is completed:

1. The quadratic error is less than an established value.
2. A maximum number of steps is reached, considering in this case that the net has failed in the learning of the grammar corresponding to the training samples.

The number of units in the output layer of each net was set to 2. One of these two units is called the *acceptance unit*, while the other one is called the *rejection*

unit. The network is trained in such a way that the desired value of the output units for a grammatically correct sentence (a positive example) is a value close to 1 in the case of the acceptance unit and a value close to 0 in the case of the rejection unit.

For the input to each net pattern, the data are encoded in a window of fixed size constituted by segments included in c possible clusters, where c is the number of different grammatical categories (noun, verb, adverb, etc.). The training samples are labelled using thesse categories. For the training of the nets two types of algorithms have been used: Backpropagation through time (BPTT) [18] for the recurrent nets and the classic backpropagation for the non-recurrent network.

The results obtained for these three networkss are described next. In Table 1 the rates of correct recognition cases are shown on the group of commands used for training the networks, and in Table 2 the same rates are shown for commands not presented in the training samples. The experiments corresponding to Table 2 have been carried out generating 500 commands syntactically correct or incorrect. The maximum number of iterations in the learning algorithms has been set to 3500 in all the cases.

Table 1. Successful rates for training commands

	Large input window	**Small input window**
Totally connected network	90	95
Elman network	100	100
BT FIR network	100	70

Table 2. Sucessful rates for generalization commands

	Large input window	**Small input window**
Totally connected network	50	60
Elman network	70	85
BT FIR network	65	55

The results have been obtained for a window size of 2 words in the case of the small window, and a size of 8 words for the case of the large window. In all the cases 10 nodes have been used in the hidden layer. What is interesting in these experiments is that for the natural language processing task in the robot's teleoperation is feasible the use of one of these three nets to extract new grammatical rules which are added to the rules initially considered in the

teleoperation interface. This property is very useful to make the interface easily adaptive for the operator.

4 Compound Recurrent Network

Another possibility of using recurrent neural nets for grammatical learning consists on applying the capacity that these systems have to learn distributed codes of input sequences. Some authors called this approach of obtaining an internal "neural" code of a data structure *holistic computation* [1].

4.1 Network Architecture

The general idea of the use of autoassociative nets for syntactic analysis is summarized in Figure 3. Connectionist or neural representations are obtained for the input sentence and for its syntactic tree. The analysis is then made by a correspondence between the representation of the sentence with the representation of its syntactic structure (holistic transformation). The input sentence is encoded

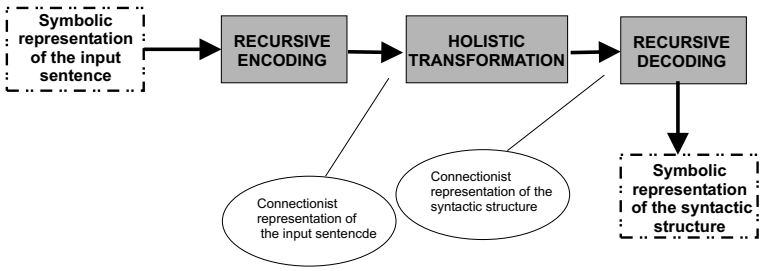

Fig. 3. Autoassociative nets for syntactic analysis.

using RAAM networks (Recusive Autoassociative Memories) [17]. The encoder is, therefore, a recursive function that takes as input a vector of $[0,1]^N \times [0,1]^K$. The implementation of the encoder is carried out as a simple perceptron:

$$r_i(t) = g\left(\sum_{j=1}^{N} W_{ij}^{rr} r_j(t-1) + \sum_{j=1}^{N} W_{ik}^{ru} u_k(t) + W_i^r\right) \tag{8}$$

where W_{ij}^{rr} are the recursive weights, W_{ik}^{ru} are the weights connecting the input and hidden units, W_i^r is the bias corresponding to the i-th hidden unit, and g is a sigmoid function.

The net has, therefore, $N + K$ input units, and N output units. With this formulation of the encoder, an input sentence i with length L_i can be encoded placing a binary representation of the first terminal symbol in the K first input

units, and a representation of the empty string in the N remaining units. Then, the encoder produces an internal representation for the first symbol in its N hidden units. The activations of the hidden units are copied recursively in the last N input units, placing the representation of the second terminal symbol of the sentence in the K first input units. This process is repeated until the encoder has processed all the terminal symbols of the input sentence, obtaining a representation for the sentence in the N hidden units of the net. The encoder of the syntactic tree operates in a similar way.

4.2 Training and Experiments

This architecture has been trained with sentences used in the natural language interface to the robot. A total number of 80 sentences have been used together with their corresponding syntactic analysis trees. A random subset of 60 sentences has been chosen for training, while the 20 remaining sentences have been used to check the generalization capacity. The length of the longest sentence has been of 15 words, and the more complex tree analysis had 5 levels. The network has been trained using the BPTT algorithm [18] for the encoders, and the standard backpropagation algorithm the net that performs the transformation between the representation of the input sentence and the representation of the syntactic analysis tree. The generalization obtained in the experiments has been of 100 % in all the cases.

Experiments to show the error recovery capability of the network have also been made. The types of errors that have been introduced in the sentences for the recovery capacity experiments are the following:

1. *Substitution* of a input terminal for another terminal that does not make sense in the sentence. This substitution simulates the situation in which the voice recognizer fails to recognize a word in the sentence.
2. *Insertion* of a new terminal symbol in the sentence. This simulates the inclusion of some incorrect sound, emitted by the user, and that the voice recognizer has recognized as the nearest word to the signal received by the microphone.
3. *Deletion* of a terminal symbol of the sentence. The recovery of this error depends on the importance of the terminal eliminated in the semantics of the sentence.

With these modifications types on the 60 original training sentences, a test set of 40 sentences has been obtained for each type of errors. These input sentences are entered to the network that has been trained. In the experiments a sentence is considered correctly recovered if the two following conditions are completed: [1]:

1. The generated tree corresponds to a correct sentence from the syntactic point of view.
2. The sentence corresponding to the tree doesn't differ much from the erroneous sentence.

The second condition is justified since if a small recognition error had taken place in the sentence, the analysis tree should not differ much from the analysis tree corresponding to the original sentence without errors. In the experiments, the condition 2 is considered satisfied if two (sub)conditions are completed:

1. The length of the sentence corresponding to the generated tree is correct or differs in one symbol from the correct sentence.
2. The maximum number of terminal symbols with errors is 2.

The percentages of recovery errors following the previous conditions is shown in the Table 3. The best generalization results have been obtained for a total of $N = 9$ input neurons.

Table 3. Error recovery rates

Substitution	**Insertion**	**Deletion**
91%	71%	76 %

5 Conclusions

The experiments presented in this paper have shown that the use of autoassociative networks is useful to obtain additional grammatical rules to those that exist originally in the grammar, with the objective that the voice interface can adapt its syntactic structure to new environments or new users. This method is not considered appropriate to obtaining the initial grammar (and, therefore an initial grammar design is needed). The experiments also have shown that sentence encoding using RAAM networks is a quite robust technique for the experienced task.

In general, the connectionist analyzers can learn the grammatical regularity that exists in the training sentences in a inductive way. As a consequence, the application of any explicit analysis algorithm is not assumed. This characteristic is relevant in natural language phenomena which are difficult to capture with formal grammars or with transition nets and difficult to analyze with symbolic algorithms.

The main problem in the applicability of these nets is its lack of scalability to adapt to complex problems. In spite of this important disadvantage, these techniques are still very useful in the adaptation of the grammars initially designed in the system. Also, in the teleoperation application, the problem of scalability lack is more limited because the natural language expressions (commands to the robot) are not too long.

Acknowledgements

This work has been supported by the Spanish Government inside the Plan Nacional de Investigacion Cientifica, Desarrollo e Innovacion Tecnologica 2000-2003 through project DPI2001-3827-C02-02.

References

1. Shin Ho, E.K., Wan, L.C.: How to Design a Connectionist Holistic Parser. Neural Computation 11, p. 1995-2016 (1999).
2. Lawrence, S., Giles, C.L., Fong, S.: Natural Language Grammatical Inference with Recurrent Neural Networks. IEEE Transactions on Knowledge and Data Engineering (2000).
3. Regier, T.: A Model of the Human Capacity for Categorizing Spatial Relations. Cognitive Linguistics 6-1 (1995), pp. 63-88.
4. Stolcke, A., Ries, K., Coccaro, N., Shriberg, E., Bates, R., Jurafsky, D., Taylor, P., Martin, R., Van Ess-Dykema, C., Meteer M.: Dialogue Act Modeling for Automatic Tagging and Recognition of Conversational Speech. Computational Linguistics 26(3), pp. 339-373 (2000).
5. Ñeco, R.P., Reinoso, O., Garcia, N., Aracil, R.: A Structure for Natural Language Programming in Teleoperation. In: Proc. of the 6th International Conference on Control, Automation, Robotics and Vision, Singapur, December 2000.
6. Jurafsky, D., Martin, J.H.: Speech and Language Processing. Prentice Hall. (2000).
7. Stolcke, A.:Learning feature-based semantics with simple recurrent networks, TR-90-015, ICSI, Berkeley, California (1990).
8. St. John, M.F., McClelland, J.: Learning and applying contextual constraints in sentence comprehension, Arificial Intelligence 46 (1990) 5-46.
9. MacWhinney, B., Leinbach, J., Taraban, R., McDonald, J.: Language learning: cues or rules?, Journal of Memory and Language 28 (1989) 255-277.
10. Watrous, R., Kuhn, G.: Induction of finite-state languages using second-order recurrent networks, Neural Computation 4(3) (1992).
11. Sperduti, A., Starita, A., Goller, C.: Learning distributed representations for the classification of terms. Proceedings of the International Joint Conference on Artificial Intelligence (1995) pp. 509-515.
12. Zeng, Z., Goodman, R., Smyth, P.: Discrete recurrent neural networks for grammatical infence. IEEE Transactions on Neural Networks 5(2) (1994) 320-330.
13. Giles, C.L, Horne, B., Lin, T.: Learning a class of large finite state machines with a recurrent neural network. Neural Networks 8(9) (1995) 1359-1365.
14. Narendra, K.S., Parthasarathy, K.: Identification and control of dynamical systems using neural networks. IEEE Trans. on Neural Networks, 1(1):4-27 (1990).
15. Elman, J.L.: Distributed representations, simple recurrent networks and grammatical structure, Machine Learning, 7(2/3):195-226 (1991).
16. Back, A.D., Tsoi, A.C.: FIR and IIR synapses, a new neural network architecture for time series modelling Neural Computation, 3(3):337–350 (1991).
17. Pollack, J.B.: Recursive distributed representations, Artificial Intelligence, 46, 77-105.
18. Williams, R.J., Zipser, D.:Gradient-based learning algorithms for recurrent connectionist networks, in Chauvin, Y., Rumelhart, D.E. (eds.), Backpropagation: Theory, Architecture, and Applications. Erlbaum, Hillsdale, NJ, (1990).