A Natural Language Translation Neural Network

Nenad KONCAR
Imperial College of Science, Technology and Medicine, London, UK
N.Koncar@doc.ic.ac.uk

Dr. Gregory GUTHRIE
Maharishi International University, Fairfield, Iowa, USA
guthrie@miu.edu

Abstract

We have tested the ability of neural networks to perform natural language translation. Our results have shown a greatly improved translation accuracy in comparison to the work of R.B. Allen (1987) in translating English into Spanish. A neural network was trained on a set of 10,000 sentences from a total of 24,750 sentences using a novel training algorithm. On a test set of 100 sentences the neural network showed a 98% sentence accuracy. The neural network had 48 input nodes, 70 nodes in the first hidden layer, 1 node in the third hidden layer, and 36 nodes in the output layer (48-70-1-36). A fully connected architecture was used.

Keywords: Example-based MT, Connectionist NLP, Neural Networks

Introduction

Machine translation of natural languages has been tackled in many ways. This paper presents the usefulness of feedforward neural networks to perform such tasks. Most conventional approaches to machine translation fail to adequately address the problem of allowing a machine language translation system to learn form a human expert translator during its use. Most systems have hard coded rules (with varying depth of meaning) that can be modified only by a knowledge engineer who is an expert in expressing such rules in the language of the computer. Neural networks, having the ability to learn from examples, are a possible solution to this problem. If such a neural network translation system wrongly translates a sentence it can be corrected and taught the proper translation by a user without any expert knowledge of how the computer stores and represents rules. This paper demonstrates the utility of neural networks in precisely this area on a small scale translation problem.

Connectionist NLP

Research has already shown the usefulness of neural networks in various natural language processing tasks: (Allen, 1987), (Jain, 1991), (Waibel, 1988) and (Waibel et al, 1991).

Grammar Used

The grammar used to generate the 24,750 English and corresponding Serbo-Croatian sentences was kept simple. Each English sentence consisted of up to eight (8) words with two determiners, two nouns, a verb, an adverb, a preposition and an adjective. Each Serbo-Croatian sentence contained two nouns, a verb, an adverb, a preposition and an adjective for a total of six (6) words. The training set was generated from a simple deterministic grammar by a program. The program can grade 8 English components, with a total of (11 x 5 x 9 x 2 x 5 x 5 = 24,750) 24,750 sentences (see Table 1). In addition to learning simple word pair mappings certain words changed translation depending on context. The adjective and noun endings are different for different adjective and noun combinations in Serbo-Croatian, so this was a context rule that had to be learned by the neural network. The noun and adjective endings also changed when one of the two different prepositions was used, adding another context rule.

Determiner	Noun	Verb	Adverb	Preposition	Determiner	Adjective	Noun	
The	child	runs	quickly	to	the	large	house.	English
	Dijete	trći	brzo	do		velike	kuće.	Serbo-Croat

Figure 1: Example Translation. The first line shows the parts of speech in each of the sentences under it. The second line is the English sentence and the third line is the Serbo-Croatian sentence.

TABLE 1: Words Used For Each Part of Speech

First Noun (11 words)		**Verb** (5 words)		**Adverb** (9 words)		**Preposition** (2 words)	
English	Serbo-Croatian	English	Serbo-Croatian	English	Serbo-Croatian	English	Serbo-Croatian
child	dijete	runs	trći	quickly	brzo	to	do
man	čovjek	walks	hoda	slowly	polako	toward	prema
woman	žena	strolls	korača	carefully	oprezno		
person	osoba	skips	poskakuje	hurriedly	ubrzano		
actor	glumac	glides	klizi	tiredly	pospano		
driver	vožac			energetically	snažno		
doctor	doktor			instantane-ously	trenutno		
father	otac			insecurely	nesigurno		
mother	majka			blissfully	radosno		
brother	brat						
sister	sestra						

Adjective					**Second Noun**			
English (5 words)	Serbo-Croatian (20 words)				English (5 words)	Serbo-Croatian (10 words)		
large	velike	velikog	velikom	velikoj	house	kuća	kuće	kući
big	ogromne	ogromnog	ogromnom	ogromnoj	car	auto	auta	autu
small	male	malog	malom	maloj	swimming pool	bazen	bazena	bazenu
nice	predivne	predivnog	predivnom	predivnoj	boat	brod	broda	brodu
beautiful	lijepe	lijepog	lijepom	lijepoj	field	njiva	njive	njivi

The positioning of the verb and the adverb in the English sentence was randomly chosen in order to add more variety to the training set. The training set was randomised after generation, because it was found that the neural network refused to learn after a certain time if the training set was not properly randomised. The translation was not a simple direct mapping because of the difference in the number of words and the context sensitive change of the adjective and noun according to both which preposition was used and which adjective and noun pair was used, and also the random change of verb and adverb positions in the English sentence.

Incremental Search

Back-propagation (Rumelhart et al, 1986) seems to be the most widely used learning algorithm in feedforward neural networks today. The back-propagation algorithm, being an algorithm that can only find a local minimum in weight space, has several limitations in terms of speed of training and modelling capability when one tries to address problems of growing complexity such as natural language translation. Initially we tried to use the back-propagation training algorithm to solve our machine translation problem, but results were not satisfactory, so a new heuristic algorithm for searching for an optimal neural network topology and weights was developed which uses back-propagation as one of its subroutines.

Pseudo code for the "Incremental Search" algorithm:

1. Create a feedforward neural network (N.N.) that has only an input and an output layer with each having a number of nodes predefined by the training data set. Initialise all the variables.

2. Create a new current hidden layer and place it just before the output layer.

3. Add one more node (or use "chunks" of 'n' nodes as increments) to the current hidden layer and connect it with random weights to all nodes in all other layers.

4. FOR NUM_RANDOMIZATIONS
   (the purpose of this 'for' loop is to try different random starting weights used by the back-propagation algorithm called in 4.1.1)
   4.1. REPEAT
      4.1.1. Train the N.N. using back-propagation over 1 presentation of the entire training set. During learning keep track of RMS (root mean squared) error on the training set (**RmsTrain**).
   4.2. WHILE **RmsTrain** is decreasing (i.e. network is learning)
   4.3. Calculate RMS error (**RmsEval**) on the early test set (note: the early test set is different from the final test set).
   4.4. Update **RmsTrain_best** and **RmsEval_best** with the smallest values of **RmsTrain** and **RmsEval**. If an update was made then save the network architecture and the starting weights for that architecture.
   4.5. Re-randomise the newly added weights.

5. IF **RmsEval_best** < **RmsEval_prev_layer** (RMS error on the early test set just before a new layer was added)
   THEN
   5.1. IF **RmsEval_best** > **RmsEval_prev** (RMS error on the early test set just before a new node was added)
      THEN (memorisation has occurred)
      5.1.1. Remove the just added node and restore the state (topology and weights) of the N.N. to the state just before the node was added.
      5.1.2. Goto 2.
      ELSE
      5.1.3 Goto 3.
   ELSE
   5.2. IF **RmsEval_best** <= RMS_LIMIT
      THEN
      5.2.1. Inform the user that a network has been designed that can correctly generalise within the given error limit (RMS_LIMIT).
      ELSE
      5.2.2. Inform the user that more training data is needed in order to achieve the desired error rate (RMS_LIMIT) on the early test set. (Also ask the user if the training data was properly randomised or not.)

For a graphic representation of the neural network architecture used by the algorithm please refer to the following Figure 2.

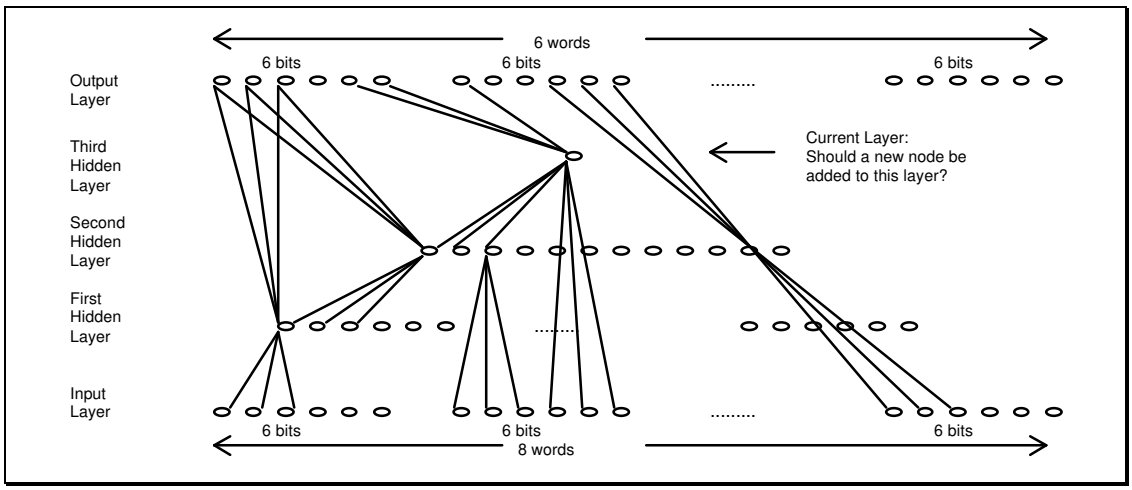

Figure 2: Language Translation Neural Network. Each circle represents one node in the neural network. The input layer consists of a total of 48 bits. Each group of 6 bits represents one word and thus there are 8 words in the input layer. These 8 words represent the English sentence. The output layer consists of a total of 36 bits which makes a total of 6 words in the output layer. These 6 words represent the Serbo-Croatian sentence. Each layer if fully connected to every previous layer.

It has been shown that the first layer of a back-propagation neural network serves as a feature detector and that successive layers try to combine these features in order to achieve the desired output (Lang & Witbrock, 1988), (Touretzky, 1989), (Mirchandani & Cao, 1989). It has also been observed that if there are too many nodes in a layer then the ability to generalise is lost (Caudill, 1990). This is the justification for why layers are constructed one after the other in the "Incremental Search" algorithm. We need to first find out all the possible features from the data by using the first hidden layer. We will know when the first hidden layer is big enough when we observe no further improvement in generalisation (i.e. when memorisation starts to occur). Back-propagation learning generally slows down by an order of magnitude every time a layer is added to a network. This is because the error signal, generated at the output layer, is attenuated each time it flows through a hidden layer. Learning progress is therefore limited by the slow adaptation of units in the early layers of a multi-layer network. To avoid this one can use short-cut connections to provide direct information pathways to all parts of the network (Lang & Witbrock, 1988). This is why our algorithm connects each new added layer to all the previous layers. During the execution of the "Incremental Search" algorithm the architecture of the neural network is changed slightly by adding a new node or a new layer the error surface (Tank & Hopfield, 1989) will also change allowing for an escape from a local minima in the error surface that the learning algorithm might be in.

Optimal Neural Network

The "incremental search" algorithm does not guarantee an optimal neural network topology and weights (Judd, 1987), (Baum, 1989) because it is a heuristic algorithm. However, our experiments show that it will produce a satisfactory network topology and weights for those problems where back-propagation fails.

Other Growth/Trimming Algorithms

The Cascade-Correlation Learning Architecture (Fahlman & Lebiere, 1990), The Tiling Algorithm (Mezard & Nadal, 1989), Optimal Brain Damage (Le Cun et al, 1990), "Optimization of the Architecture of Feed-forward Neural Networks with Hidden Layers by Unit Elimination" (Burkitt, 1991) are some examples of existing growth and trimming algorithms for training neural networks and automatic neural network topology formation. Our "incremental search" algorithm differs in very important details from all the above listed algorithms. Our algorithm is capable of generalising well when real valued numbers are used as outputs whereas the Cascade-Correlation Architecture does not do this well and the Tiling Algorithm can deal with only binary valued outputs. The Optimal Brain Damage algorithm and Burkitt's algorithm are both trimming algorithms which are fundamentally different from ours so we cannot compare the them.

Results

Each word in each language was assigned a unique binary code. Because the total number of words in one language did not exceed sixty four (64) it was possible to use six (6) bits to encode all the words in one language. Please refer to Figure 2 to see how this encoding scheme produced a total of 48 bits in the input layer and a total of 36 bits in the output layer of the neural network. The evaluation scheme used was very thorough. Both word errors, sentence errors and part of speech errors were evaluated and recorded. Any output greater than .5 was interpreted as a 1 , otherwise it was interpreted as a 0.

The results that are shown in the series of tables in this section were achieved after training each neural network on a training set of 10,000 English sentences and their Serbo-Croatian translations. Each neural network was then evaluated on a set of 100 English and Serbo-Croatian sentences. It was shown that using the "Incremental Search" algorithm that it was possible to achieve a high accuracy rate on the test set. The neural network architectures produced at different stages of the execution of the algorithm are listed in the tables 2,3 and 4 shown below. The left-most column of each table shows the network architecture that was evaluated. It was found that the results achieved were extremely sensitive to randomisation of the input/output vector training pairs. If no randomisation was present the neural network would learn certain words in the translation very quickly but would refuse to learn those words with a low frequency of change in the training set. It was also found that the random starting weights of each neural network architecture tested had a major influence on the accuracy of the resulting neural network solution.

Note in table 2 that as the size of the first hidden layer grows toward 70 nodes that the performance of the network improves, but then falls after further increasing the number of nodes. This is a clear indication that memorisation of the training data is occurring. Thus a good approximation to the optimal number of nodes in the first hidden layer is 70 from the results observed. Because of the structure of the grammar used, it was easiest for the neural networks to learn how to translate the preposition (there were only two prepositions used). It was hardest to learn the grammar context sensitive rules between the adjective and the second noun and this was also observed. Furthermore, it was also slightly harder to learn the adverb because of the fact that the grammar randomly chose to interchange the positions of the verb and adverb in the training and test sets.

It can also be observed that a marked increased in network performance occurs each time a new layer is added. It must be noted that this performance increase is greatly due to the fact that the weights of the previous layer are preserved and that the learning rate of the previous layer(s) is changed to zero (frozen) when the new layer is added to the neural network architecture.

All of the following neural networks had these parameters in common:

- Training File: 10,000 sentences
- Test File: 100 sentences
- Learning Rule: Delta Learning Rate: 0.6 Momentum: 0.9
- Number of Presentations: 10,000

The results achieved with the different architectures on the test set follow:

TABLE 2: Performance Measurement of Three (3) Layer Neural Networks

Network Structure	Word Error %	Sentence Error %	Parts of Speech Error %					
			Noun 1	Verb	Adverb	Preposition	Adjective	Noun 2
48-2-36	72.92	100	15.34	9.47	14.77	0.0	16.67	16.67
48-5-36	57.39	100	14.96	9.09	15.53	0.19	16.67	0.95
48-10-36	58.14	100	7.20	6.06	7.77	4.36	16.67	16.1
48-15-36	36.17	100	5.11	6.06	6.25	0.0	15.34	3.41
48-20-36	20.08	100	4.36	5.87	5.49	0.0	4.36	0.0
48-25-36	30.30	100	5.11	5.30	5.49	0.0	14.39	0.0
48-30-36	17.80	100	4.55	3.03	4.55	0.0	5.68	0.0
48-35-36	19.89	100	2.27	3.60	2.84	0.0	11.17	0.0
48-40-36	18.18	100	4.55	7.58	5.49	0.0	0.38	0.19
48-45-36	27.27	100	0.95	3.03	5.30	0.38	15.91	1.70

48-50-36	29.36	100	2.08	4.92	6.82	3.79	9.85	1.89
48-70-36	2.84	18.0	0.95	0.0	0.95	0.0	0.95	0.0
48-90-36	13.26	78.0	0.57	0.19	1.89	0.57	10.04	0.0
48-110-36	36.36	97.73	3.41	16.29	16.29	0.0	0.38	0.0
48-130-36	18.94	100	0.76	6.76	2.27	0.0	15.15	0.0

Table 2. The second layer i.e. the first hidden layer was incrementally given more nodes and the resulting neural network performance was measured for each of these networks.

TABLE 3: Performance Measurement of Four (4) Layer Neural Networks

Network Structure	Word Error %	Sentence Error %	Parts of Speech Error %					
			Noun 1	Verb	Adverb	Preposition	Adjective	Noun 2
48-70-1-36	1.33	12.0	0.57	0.0	0.0	0.0	0.76	0.0
48-70-5-36	0.95	6.0	0.38	0.0	0.0	0.0	0.56	0.0
48-70-10-36	0.57	3.41	0.38	0.0	0.0	0.0	0.19	0.0
48-70-15-36	0.76	4.55	0.38	0.0	0.0	0.0	0.38	0.0
48-70-20-36	0.76	4.55	0.57	0.0	0.0	0.0	0.19	0.0

Table 3. The third layer i.e. the second hidden layer was incrementally given more nodes and the resulting neural network performance was measured for each of these networks.

TABLE 4: Performance Measurement of Five (5) Layer Neural Networks
With Different Random Starting Weights

Network Structure	Word Error %	Sentence Error %	Parts of Speech Error %					
			Noun1	Verb	Adverb	Prepositio n	Adjective	Noun 2
48-70-10-1-36	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
48-70-10-1-36	0.19	1.14	0.0	0.0	0.0	0.0	0.19	0.0

Table 4. Each neural network had the same structure 48-70-10-1-36 only different random starting weights distinguished one network form the other.

Discussion

Neural networks have been shown to have the capability to translate sentences from English to Spanish (Allen, 1987). Sentences used in Allen's research were generated using a highly limited vocabulary and format. All sentences included a subject, verb, direct object, and indirect object. The verb was either 'to give' or 'to offer' and three different verb tenses were used (present, past, and past perfect). In the English sentences, the order of direct object and indirect object was randomly selected, while in the Spanish sentences the preferred sentence structure always places the indirect object after the direct object. Nouns referring to people (including two first names) and animals were used as subjects and indirect objects, while nouns referring to things were used for direct objects. Nouns were randomly modified by one of the two adjectives. A training set of 3310 sentences and 33 test sentences were used to train and test a multi-hidden-layer 50-150-150-150-66 back-propagation network with learning rate=0.01 and momentum=0.9. After 2 million presentations training produced a root mean squared error rate of 0.027 over all the outputs. On the test set this trained network was incorrect on average 2.5 bits and 1.3 words.

The Scalability Problem

Terence K. Sejnowski in his work "Parallel Networks that Learn to Pronounce English Text", (Sejnowski, 1987) used a context of seven (7) characters (including the space) from a word as input to a neural network and the phoneme corresponding to the middle character as output. This same approach could be taken in using neural networks to perform real-world machine translation. The neural network would be given a sliding window of words from the source language and the neural network output would be a single word in the target language text.

Multiple Meaning Words

Context rules (i.e. multiple meaning words) of language translation would be captured by using Sejnowski's method (Sejnowski, 1987) just the same as for the scalability problem just discussed. Requirements for the neural network size in terms of the number of neurons and

hidden layers would be also reduced. If the sliding window of words from the source language is increased sufficiently to cover three or more sentences then inter-sentence information could be captured and used in translation.

Text Attributes

There are some attributes that can be given to an entire text. If a text is written in old English then its attribute would be old English. If the text is a legal document then its attribute would be that of a legal document, etc. Giving such attributes as input to the neural network during training and during its use by a skilled operator would greatly enhance the accuracy and versatility of translations that the neural network could perform.

Acknowledgements

We would like to thank all of those who have contributed ideas to this paper and have reviewed its manuscript giving many helpful comments.

References

Allen, R.B. (1987). Several Studies on Natural Language and Back-Propagation: Natural Language Translation. Proc. First International Conference on Neural Networks, vol. 2, pp. 338-339.

Baum, E.B. (1989). What Size Net Gives Valid Generalization? *Neural Computation*, vol. 1, pp. 151-160.

Burkitt, A.N. (1991). Optimization of the Architecture of Feed-forward Neural Networks with Hidden Layers by Unit Elimination. *Complex Systems*, vol. 5, pp. 371-380.

Caudill, M. (1990). Using Neural Nets: Diagnostic expert Nets, Pruning Trained Networks. *AI Expert*, September 1990, p. 45.

Fahlman, S., Lebiere, C. (1990). The Cascasde-Correlation Learning Architecture. Carnegie Mellon University Technical Report, CMU-CS-90-100.

Jain, J.S. (1991). Parsing Complex Sentences with Structured Connectionist Networks. *Neural Computation* vol. 3, pp. 110-120.

Judd, J.S. (1987). Learning in networks is hard. In Proc. First Int. Conf. Neural Networks, pp. 685-692.

Lang, K.J., Witbrock, M.J. (1988). Learning to Tell Two Spirals Apart. Proceedings of the 1988 Connectionist Models Summer School, pp. 52-59.

Le Cun, Y., Denker, J., Solla, S. (1990). Optimal Brain Damage. *Advances in Neural Information Processing Systems 2*, Morgan Kaufman Publishers, San Mateo, CA.

Mezard, M., Nadal, J.P. (1989). Learning in Feedforward Networks: The Tiling Algorithm. *Journal of Physics A*., vol. 22, pp. 2191-2203.

Minsky, M., Papert, S. (1969). *Perceptrons.* MIT Press, Cambridge, MA.

Mirchandani, G., Cao, W. (1989). On hidden nodes for neural nets. *IEEE Trans. Circuits Syst*., vol. 36, pp. 661-664.

Nadal, J.P. (1989). Study of a Growth Algorithm For a Feedforward Network. *International Journal of Neural Systems*, vol. 1, no. 1, pp. 55-59.

Rosenblatt, F. (1962). *Principles of Neurodynamics.* Spartan Books, New York.

Rumelhart, D.E., Hinton, G.E., Williams, R.J. (1986). Learning internal representations by error propagation. *In Parallel Distributed Processing: Explorations in the Microstructure of Cognition.* Cambridge, MA: MIT Press.

Sejnowski, T.J., Rosenberg, C.R. (1987). Parallel Networks that Learn to Pronounce English Text. *Complex Systems*, vol. 1, pp. 145-168.

Tank, D.W., Hopfield, J.J. (1989). Collective Computation in Neuronlike Circuits. *Scientific American*, pp. 104-114.

Touretzky, D.S. (1989). What's Hidden in the Hidden Layers? *Byte*, August 1989, pp. 227-233.

Waibel, A. (1988). Connectionist Glue: Modular Design of Neural Speech Systems. Proceedings of the 1988 Connectionist Summer School, pp. 417-425.

Waibel, A., Jain, A.N., McNair, A.E., Saito, H., Hauptmann, A.G., Tebelskis, J. (1991). Janus: A Speech-to-Speech Translation System Using Connectionist And Symbolic Processing Strategies. IEEE Proceedings of the 1991 International Conference on Acoustics, Speech, and Signal Processing, April 1991.