Phonematic translation of polish texts by the neural network

A. Bielecki   I. T. Podolak   J. Wosiek *
Institute of Computer Science, Nawojki 11, Cracow, Poland
and
E. Majkut
Institute of Polish Philology, Jagiellonian University
Gołębia 14–20, Cracow, Poland

Abstract

Using the backpropagation algorithm, we have trained the feed forward neural network to pronounce Polish language, more precisely to translate Polish text into its phonematic counterpart. Depending on the input coding and network architecture, 88%-95% translation efficiency was achieved.

CSJU — 05/95
June 1995

*Supported in part by the KBN grants PB 2P03B19609 and PB 2P30225206.

1 Introduction

A human brain is a system, which is able to solve a lot of complex problems. Wide range of processes connected with learning and the way problems are solved by human brian are under investigations for years. Letter–to–sound translation of text in native language is a multistep task which is efficiently solved by a lot of human brains every day. The key step of the translation is the transformation from the highest sensory representations of the letters to the earliest articulatory representations of the phonemes.

Computer science has been interested in phonematic translations for a long time. So far the problem was approached in two different ways. A simple implementation of all transformation rules, including irregularities, is the first possibility. A DECtalk system converting English text to speech was implemented in such a way [?]. In such commercial systems a look-up table of about million bits is used to store the phonetic transcription of common and irregular words, and phonological rules are applied to words that are not in this table [?], [?]. A similar system for Polish is described in [?]. Using an artificial neural network is the second possibility. A NETtalk, described in [?] and [?], and also mentioned in [?] where its abilities and implementation difficulties in comparison with the DECtalk are discussed, is a network system that learns to convert English text to speech.

In this paper we describe few neural networks that learn to pronounce Polish text. Each of them has slightly different architecture and differs with respect to others, both in learning speed and translation efficiency.

2 Phonemes and phonematic transformations in Polish

Polish language is much simpler than English to specify all transformation rules. Bolc and Maksymienko have implemented a system based on this approach and have obtained good results [?]. We have used an artificial neural network for this purpose in order to test both approaches and also to compare results with those of Sejnowski's NETtalk implementation for the English language.

The implemented neural networks transform letter with its context (i.e. letters before and behind the letter) into a phoneme. A phoneme is the

smallest part of a language which differentiates a meaning but has no meaning itself. For instance, two Polish words:
tam [**tam**] (there) and
sam [**sam**] (alone)
differ only by the parts [t] and [s]. Therefore [t] and [s] are phonemes according to the definition given above. We have defined phoneme by its functional role it plays in a word but another definition can be put forward: a phoneme is a set of features which distinguish the given phoneme from another one. Such features are called distinctive. In this way the phoneme is defined by its structure, i.e. the way it is pronounced. As an example consider two Polish words:
dam [**dam**] (I will give) and
tam [**tam**] (there).
The phoneme [d] is voiced, dental, buccal, hard whereas [t] is unvoiced, dental, buccal, hard. It has been shown that a voice is a distinctive feature. We use the Jakobson theory [**?**] of phoneme description. In this theory all phonologic features are binary hence each phoneme is represented by a vector with binary entries. Fourteen distinctive features (binary ingredients) are needed to describe all Polish phonemes unambiguously (see Tables 1 to 3). However not all of them are required for every phoneme.

Care must be taken while implementing automatic transformation between the text and phonematic transcription. The first problem is that the transformation of a single letter into a phoneme is usually ambiguous. In most cases it depends on the context in which the letter appears, i.e. it depends on the letters (or, generally, characters) both before and after the letter in question. For example, the letter sequence *dzi* gives in effect a single phoneme [$\acute{\zeta}$] and using the rules of Polish language letter *d* is transformed into [$\acute{\zeta}$] and both *z* and *i* into empty phonemes ([**?**]). Thus the transformation of *d* in this sequence depends upon two letters following it, transformation of *z* upon both one letter before and one after it, and that of *i* upon two letters preceding it. Some letters are independent of the context (like a, j, l, ł, ń, o, and ó), some are dependent only on the neighbouring letters, and some, yet less numerous, depend on two or three letters. In addition, an interwords and midwords assimilations exist in Polish, for instance:
kosz truskawek [**koš-truskavek**] (a basket full of strawberries) and
kosz gruszek [**kož-grušek**] (a basket full of pears).

Variations of transformation of Polish nasals (ą, ę) provide another ex-

ample of similar difficulties:
ręka [**re ṳka**] (hand): ę ↦ e ṳ

dębu [**dembu**] (oak – a genetive form): ę↦ em
Kęty [**kenty**] (a name of the town in Poland): ę↦ en
wzięła [**vźe ṳa**] (she gave): ę↦ e ṳ

kręcił [**kreńciw**] (he turned round): ę↦ eń

Additional difficulties appear in rare and unorthodox events (for instance: zamarzać [**zamarzać**]–to freeze) [**?**]. Transformations for which only few examples exist cause also some problems (for instance: ź ↦ empty phoneme, d ↦ [C] in Polish). Furthermore, there are no rules for a few borrowings and specialistic terms [**?**]. Moreover, the text can be pronounced correctly in different ways. There exist not only classical and common Polish but Warsaw and Cracow pronunciation as well [**?**]. We taught the implemented neural network according to the classical Cracow pronunciation basing on [**?**]. A list of various transformations together with the frequency they occurred in our training and testing sets used are given in Table 2.

A set of transformations discussed above allows to translate correctly all Polish words apart from a few borrowings, for instance *cyjanek* (cyanide) or foreign names often appear in Polish, for instance a surname Katz. However, even translation of such words using only described possibilities does not lead to misunderstanding.

The nature of the text to phoneme transformation issue predestinates it for a neural network approach. To solve it in a classic algorithmic way one would need a set of all relevant transformation rules. Finding a proper set of rules for a given language usually causes a lot of difficulties [**?**]. For instance, English is particularly difficult to master because of its irregular spelling [**?**], [**?**]. Even if a suitable set is given, it is always either huge or incomplete, which leads to impossibility of obtaining 100% accuracy [**?**] and is therefore no better than a neural net, which never accomplishes 100%. It is always an arduous task to implement it in an algorithmic way. On the other hand for a given word its phonematic transformation is well known if only a type of pronounciation is given (*eg.* literary or cockney for English). Because training of a neural network requires only a set of examples, it seems to be the convenient method to solve the problem. Finally, the similarities of the learning process of a neural network to that of a human make this approach even more interesting.

The neural network method has also some disadvantages. As it was mentioned, no artificial neural network can give a 100% accuracy. There can be problems with finding a set of training examples that would be statistically suitable, and small enough for the training part to end in a reasonable time. Additionally, finding a proper network architecture is not straightforward and usually a number of trials are needed. The same applies to choosing the right representation of both the input text and the resulting phonemes, one that would at the same time be compact for the net to be of modest size and result in easy input–output mappings. A few neural networks, which we have implemented, are described in the next Section.

3 Implementation of an artificial neural network

We have employed a multilayer feed forward network, which was trained by the backpropagation method (BPM) in a supervised mode [?]. The network consists of layers of neurons — the input layer into which the pattern being recognized is fed, then the hidden layer (or a number of them) which receives a signal from the input layer and propagates it forward, and at the end the output layer which generates the answer. The neurons are fully interconnected on the layer basis, that is every neuron in one layer is connected to each neuron in the next layer, but no neurons in one layer are connected between themselves and there are no connections between neurons in layers that are not neighbouring. During the learning (BPM) phase the generated output is compared to the given correct output (the supervised mode) and an error is computed for each output neuron. Afterwards, the information about this error is "backpropagated" to previous layers in such a way that all neurons receive only a part of the total error, the part they have contributed to. On the basis of this value they adapt themselves in order to lessen the error. The procedure is repeated many times for all examples given in the training set until the total error drops to a value that is small enough. The important part is that the neurons in hidden layers organize themselves in a way as to recognize different features of the input data so that, after training, the network would find these features in inputs that were not presented during training and most probably respond in a correct way.

Such a network would receive a letter at it's input and should respond with the correct phoneme. Since the transformation of a single letter into a phoneme depends on the context (as it was described in Section 2), it is logical to use a sliding window that will analyze simultaneously some number of characters. Then the output should represent the correct phoneme corresponding to the letter in the center of the window.

Now we address the problem of character and phoneme representation. Taking into account the network size and corresponding learning time, it may seem that the representation should be as compact as possible, e.g. binary in which each phoneme is represented by a binary number. But then two completely distinct phonemes could become very similar for the network and it would produce erroneous results. We have tried such a representation with only 6 output neurons, and even less in the hidden layer, but the network could not learn the given examples, not to say about a satisfactory generalization. It turns out that the proper solution is the natural one, i.e. the one used in the phonetics as described in the previous Section. Namely each phoneme is represented as a group of some of 14 articulatory features (in Polish language). The abundance of this description is exploited to decrease the probability of confusion of different phonemes.

There is a similar problem with representation of characters. A too compact one is prone to give wrong results. We have found that the network which used a binary representation could not achieve the errors smaller than 50% ! One of the possible solutions is to represent each character as a vector of n neurons with only one activated. This method, which will be referred to as *unary* was also used by Sejnowski [**?**]. Unfortunately the number of input neurons becomes high (34*7=238) which lengthens the learning process. Additionally the input is more sensitive to the noise. A change of a value of a single neuron, which corresponds to a noise on a 3% level, results in an input character being equally similar to two different letters and the networks output may be totally wrong. For comparison we have also employed the *distributed* input technique with rather good results.

Each letter is represented with k neurons, where k is much less than n, the number of different characters. Out of 2^k possibilities we take n [1] different input vectors in_i in such a way that in each one there is approximately $k/2$ active and inactive neurons, and consequently the Hamming distances

[1] Where n is the number of letters, punctuation, and interword space.

between all pairs of vectors are similar. This choice has two advantages. First, less neurons are needed to represent a single character — 21 in our case. What follows, is that the number of synapses in the network is lower, thus the number of examples needed to achieve satisfying generalization. On the other hand, it results in a more complicated error function surface and the learning time could be longer. Second, even with a few erroneous neuron values the resulting configuration still resembles the original one, thus the noise level can be higher. This representation is called *box* by us.

4 Results

We have tested the networks with both the unary and the distributed inputs, and with the output phoneme represented by 14 articulatory features (*velar, high-pitched, low, centralized, vocal, consonantal, nasal, palatal, labial, lateral, fricative, dental, explosive, voiced*) plus 3 neurons needed to represent punctuation marks. The analyzed character was put in the middle of the seven character window. This provided the required information about the context.

The training set consisted of 13000 examples for which the correct phonematic translation was provided. The different cases appeared in the learning set with the frequency corresponding to the commonly used Polish language.

All tested networks had one hidden layer which proved to be sufficient. A network with no hidden layers could not translate text satisfactorily. The network with two hidden layers learned much slower and it had tendencies to "learn by heart", *i.e.* to remember the learning set rather than to generalize.

We have defined two kinds of recognition: one, called *perfect*, when the output contained activations that corresponded to actual phoneme representations; the other, called *correct*, where, if the actual activations did not correspond to any phoneme, than one that was closest in the sense of the Hamming distance was chosen, otherwise output consisted of the corresponding phoneme. During the training phase all networks acquired efficiency on the learning set close to $99.0\% - 99.9\%$. The translation efficiency is defined in terms of the ratio of good translations of a character to the total number of transformations performed in the sample. This has changed however when the network was presented with the unknown testing set. The situation is summarized in Table 4. We were studying the performance of a network changing the coding scheme of the input characters (second and third column) and size of the hidden layer. The number of the output neurons was kept fixed at $N_{out} = 17$ as required by our coding of phonemes by their articulatory features. The fourth column gives the length of the (BPM) training period where one epoch (iteration) is defined as the single pass through the whole learning set. In all cases the training phase was stopped when the preselected accuracy was achieved. The last two columns summarize the performance of the network on the unknown testing set which contained almost 5000 characters. The qualities and speed of learning of different networks are given in Table 4. and in Figure 1.

It can be seen that some networks had 95% of correct translations which is considered as the reasonable threshold for comprehensible spelling. Interestingly, the quality of translation was increasing with the reduction of the size of the hidden layer. This can be also seen in the Figure 1. with the learning curves. However below some critical size the performance of the network was decreasing dramatically so that no clear maximum was observed. The maximum efficiency was observed with the unary method of character coding. The distributed input leads to similar results.

The experiments were done at the Technical University of Athens on a Silicon Graphics Power Challange computer with MIPS R4000 processors. The learning times varied for different architectures (but using the same learning set) from around 59 hours to around 249 hours. However, the satisfactory performances were achieved usually much earlier. The average time needed for the networks to reach, both learning and generalization, errors near their optimal, was around 70 to 80 hours CPU time.

5 Summary

We have studied the neural network implementation of the phonematic translation of Polish text. Although the character – phoneme correspondence is simpler in Polish than in English for example, there exist a number of ambiguities which make application of the neural network suitable and interesting.

Several architectures and coding techniques were tested. Networks with more than one hidden layer were not able to generalize. This property was also observed in other applications [?]. Depending on the number of hidden neurons and on the method of coding of input characters the efficiency of reading the unknown text varied in the 88% to 95% range. In comparison, Sejnowski achieved from 77% to 91% for English language, depending on the network architecture and the training set size [?], [?]. On the other hand, the algorithm described in [?], basing on 329 rules without using any dictionary (for problems occurring with unorthodox events) gained about 96% efficiency, which is comparable to ours. The best choice results with the unary input of 238 neurons and 30 hidden neurons though the network of 147 input neurons and distributed coding performed similarly.

All training curves, as it can be seen in Figure 1., are steeply rising within first seven hundred iterations and then slowly saturate during the next few thousand epochs; this feature was also reported in [?]. Therefore it may often suffice in practice to limit the training process to this period of rapid learning.

Many questions remain open. In particular, we would like to study the performance of the distributed networks in the present context. Such networks were found to give satisfactory results when applied to other problems [?]. The choice of the training set and its effect for the generalization ability of a network requires also further investigations. Finally, it would be very interesting to study in detail the relation between the learning process of the artificial and the natural neural network, e.g. by exploiting the existing pedagogical expertise.

Acknowledgments

Two of us, I. P. and A. B., thank A. Stafylopatis for valuable advises and the hospitality and warm atmosphere that we have enjoyed during our stay at the National Technical University of Athens.

We would like to thank A. Kotański for discussions and for the critical reading of the manuscript.

feature	u	o	i	e	a	y
velar	+	+	-	-	-	-
high-pitched	+	-	+	-	-	+
low	-	-	-	-	+	-
centralized	-	-	-	-	-	+

Table 1: The vocal phonemes with their features. Other possible features (*ie. vocal, consonantal, nasal, palatal, labial, lateral, fricative, dental, explosive,* and *voiced*) are not distinctive among them.

feature	ń	m	n	ṳ	ų̤	i̤	l	r
velar	-	-	-	+	+	-	-	-
nasal	+	+	+	-	+	-	-	-
palatal	+	-	-	-	-	+	-	-
labial	-	+	-	-	-	-	-	-
lateral	-	-	-	-	-	-	+	-
dental	-	-	+	-	-	-	-	-

Table 2: The consonantal phonemes with their features. Other possible features (*ie. vocal, consonantal, high-pitched, low, centralized, fricative, explosive,* and *voiced*) are not distinctive among them.

feature	v	f	b	p	x	g	k	ź	ś	ǵ	$\acute{k}$	$\acute{\zeta}$	ć	z	s	ž	š	d	t	ζ	c	$\check{\zeta}$	č
velar	-	-	-	-	+	+	+	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
palatal	-	-	-	-	-	-	-	+	+	+	+	+	+	-	-	-	-	-	-	-	-	-	-
labial	+	+	+	+	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
fricative	+	+	-	-	+	-	-	+	+	-	-	-	-	+	+	+	+	-	-	-	-	-	-
dental	-	-	-	-	-	-	-	-	-	-	-	-	-	+	+	-	-	+	+	+	+	-	-
explosive	-	-	+	+	-	+	+	-	-	+	+	-	-	-	-	-	-	+	+	-	-	-	-
voiced	+	-	+	-	-	+	-	+	-	+	-	+	-	+	-	+	-	+	-	+	-	+	-

Table 3: The consonantal phonemes with their features. Other possible features (*ie. vocal, consonantal, high-pitched, low, centralized, nasal,* and *lateral*) are not distinctive among them.

	type	#input	#hidden	#epochs	time in s	%perfect	%correct
1	unary	238	40	2304	596e3	81.3	88.0
2	unary	238	30	3103	515e3	90.3	95.4
3	unary	238	15	2963	497e3	92.2	94.8
4	box	147	40	4741	898e3	81.6	91.0
5	box	147	35	3989	628e3	83.6	92.2
6	box	147	30	4421	692e3	83.6	92.8
7	box	147	25	2247	395e3	79.5	91.9
8	box	147	20	2108	213e3	86.1	93.6
9	box	147	15	4589	241e3	90.9	94.8
10	box	147	10	3131	258e3	83.0	92.0

Table 4: The results obtained with different network architectures (different number of hidden neurons)

_	˜ (1023)							
a	a (851)							
ą	o (4)	om (3)	on (50)	oń (8)	o u̯ (80)			
b	b (164)	p (31)						
c	ζ (17)	$\check{\zeta}$ (38)	null (141)	c (109)	ć (55)	č (77)		
ć	$\acute{\zeta}$ (23)	ć (28)						
d	ζ (29)	$\acute{\zeta}$ (51)	$\check{\zeta}$ (17)	c (50)	ć (37)	č (18)	d (203)	t (19)
e	e (176)							
ę	e (84)	em (11)	en (27)	eń (7)	e u̯ (16)			
f	f (52)	v (25)						
g	g (150)	ǵ (15)	k (61)					
h	x (149)							
i	null (309)	i (432)	i̯ (139)					
j	i̯ (218)							
k	g (34)	k (227)	ḱ (63)					
l	l (262)							
ł	u̯ (226)							
m	m (304)							
n	n (415)	ń (177)	ź (1)					
ń	ń (39)							
o	o (742)							
ó	u (88)							
p	b (40)	p (302)						
r	r (344)	š (74)	ž (49)					
s	s (268)	ś (68)	š (167)	z (76)	ž (83)			
ś	ś (113)	ź (82)						
t	d (13)	t (408)						
u	u (192)	u̯ (14)						
w	f (145)	v (355)						
y	i̯ (13)	y (473)						
z	null (322)	s (29)	z (140)	ź (23)				
ź	null (48)	ś (14)	ź (26)					
ż	null (35)	š (10)	ž (67)					

Table 5: Transformations of letters into phonemes in the training set. The number of occurrences is given in brackets

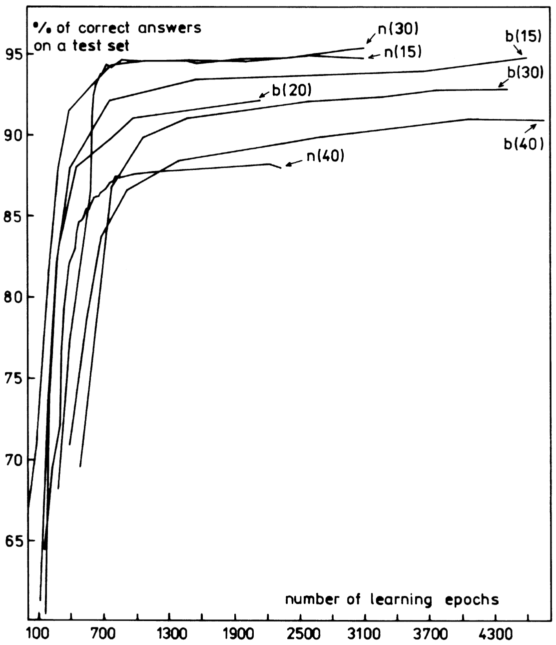

Figure 1: Figure 1. Learning curves: percent of correct answers versus number of training epochs; u(n) is a unary coded network with n hidden neurons, b(n) is a box coded one

References

[1] Digital Equipment Corporation, *DECtalk DTC01 Owner's Manual*, Digital Equipment Corporation, Maynard, Mass., document number EK-DTC01-OM-002.

[2] Allen J., *From Text to Speech: The MITalk System*, Cambridge University Press, 1985.

[3] Klatt D., *Software for a Cascade/Parallel Formant Synthesizer*, Journal of the Acoustical Society of America, 67, 1980.

[4] Bolc L., Maksymienko M., *Komputerowy system przetwarzania tekstów fonematycznych,* Wydawnictwa Uniwersytetu Warszawskiego, Warszawa, 1981.

[5] Sejnowski T.J., Rosenberg C.R., *NETtalk: A Parallel Network that Learns to Read Aloud,* Johns Hopkins University Department of Electrical Engineering and Computer Science Technical Raport 86/01, 1986.

[6] Sejnowski T.J., Rosenberg C.R., *Parallel Network that Learns to Pronounce English Text,* Complex Systems 1, 1987.

[7] Hertz J., Krogh A., Palmer R., *Introduction to the Theory of Neural Computation,* Addison–Wesley Publishing Company, 1991.

[8] Jakobson R., Halle M., *Podstawy języka,* Wrocław, 1969.

[9] Doroszewski W, *Mowa a pismo-Poradnik Językowy,* 1969.

[10] Steffen-Batogowa M., *Automatyzacja transkrypcji fonematycznej tekstów polskich,* Warszawa, 1975.

[11] Stratyński J., *Zarys fonetyki i fonologii*

[12] Karaś M., Madejowa M., *Słownik wymowy polskiej,* Warszawa, Kraków, 1977.

[13] Haas W., *Phonographic Translation*, Manchester Univ. Press, 1970.

[14] Venetzky R.L., *The Structure of English Ortography*, The Hague, Mouton, 1970.

[15] Elovitz H., Johnson R., McHugh A., Shore J., *Letter–to–Sound Rules for Automatic Translation of English Text to Phonetics,* IEEE Trans. on Acoustics, Speech, and Signal Processing, vol. ASSP–24, No.6, 1976.

[16] Müller B. and Reinhardt J., *Neural Networks An Introduction*, Springer-Verlag, Berlin Heidelberg 1990.

[17] B. Denby et al., *Neural Networks for triggering*, presented at the 1989 IEEE Science Symposium, San Francisco, January 1990; Fermilab report FERMILAB-Conf-90/20.