Word Hy-phen-a-tion by Neural Networks

Pavel Smrž and Petr Sojka

Faculty of Informatics, Masaryk University Brno
Botanická 68a, 602 00 Brno, Czech Republic
E-mail: {smrz,sojka}@fi.muni.cz**, phone: ++420-5-41512{362,352}**

Abstract
We are discussing our experiments we made to learn feed-forward neural network for task of finding valid hyphenation points in all words of given language. Multilayer neural networks were succesfully used for solving of this difficult problem. The structure of the network used is given, together with a discussion about training sets, influence of input coding and results of experiments done for the Czech language. We end up with pros and cons of tested approach—hybrid architecture suitable for a multilingual system.
Keywords: neural networks, hyphenation, back propagation, generalisation, typesetting

1 Introduction

> "The invention of the alphabet was one of the greatest advances in the history of civilisation. However, the ancient Phœnicians probably did not anticipate the fact that, centuries later, the problem of word hyphenation would become a major headache for computer typesetters all over the world."
>
> (Liang, 1983) (Liang 1983, page 39)

The problem of finding all valid hyphenation points in all words of a given language has been tackled for decades. Most of the approaches used sofar are deterministic. A rule-driven hyphenation algorithm for English was implemented in TEX78 (Liang 1981). The method was improved by Liang (Liang 1983, Knuth 1986) for use in TEX82. It is based on the generalisation of the prefix, suffix and vowel-consonant-consonant-vowel rules. The program PATGEN (Liang and Breitenlohner 1991) enables the process of pattern generation from a set of already hyphenated words to be automated. This algorithm or its derivatives are used in many DTP sys-

tems like `troff` (Emerson and Paulsell 1987), `Lout`, QuarkXpress, 3B2 and many others today.

Liang's algorithm performs well for nonflexive languages with small number of compounds like English but there is still lack of good methods for other languages, especially for flexive languages (all Slavonic languages, Dutch, German etc). Sojka and Ševeček (Sojka 1995, Sojka and Ševeček 1995) state that in Czech, on average, 20–30 different word forms—inflexions—can be derived from one word stem. This number can be almost doubled if negatives are formed from many words (adjectives, verbs, adverbs, some nouns) by adding the prefix `ne`. Thus, from a 170,000 stem wordlist about 5,000,000 inflexions may be generated in Czech.

For multilingual documents usually several separate algorithms for every language used are needed, even if the languages are dialects only, leading to high computer memory demands. Typesetting in narrow columns brings the necessity to find very high percentage of all valid hyphenation points.

From the DTP world and prominent publishers another need is being heard of: several classes of hyphenation points are called for, to make a distinction e.g. between valid and not recommended, but possible one, as published in (R. E. Allen 1990).

This leads to the stochastic approaches rather than deterministic ones – Brunak and Lautrup (S. Brunak and B. Lautrup 1990) shows that a neural network is likely to be a way leading quickly to the working solution.

2 Hyphenation Problem with Neural Networks

2.1 Hyphenation of Czech Words

We performed our experiments with multilayer neural networks trained on hyphenation for Czech language. The problem of word hyphenation in Czech is rather complex. Hyphenation rules for Czech language are described in (Zdeněk Hlavsa et al 1993) and (Haller 1956). In (Haller 1956) also a list of exceptions is given including about 10,000 words. Czech language has syllable hyphenation with "etymological" exceptions. Hyphenation is preferred between a prefix and the stem and on the boundary of compound words.

2.2 Neural Net Architecture

The architecture of the networks used in the experiments is similar to that of NETtalk (Sejnowski and Rosenberg 1987)—multilayer feedforward nets. We use usual notation here: 7-30-1 means NN topology with 7 neurons in input layer, 30 neurons in the middle layer and 1 neuron in output layer. Layers are fully interconnected.

The input of our network is a series of seven consecutive letters from one of the training words. The central letter in this sequence is the "current" one for which the output is to be produced. Three letters on either side of this central letter provide

context that helps to determine the hyphenation point. Individual words are moved through the input window so that each letter in the word with the exception of the last two is "seen" in the central position. Blanks are added before and after the word as needed.

One type of tested networks uses unary encoding. For each of the seven letter positions in the input, the network has a set of 43 input units: one for each of 41 letters in Czech, one for letters from other languages, and one for blank. Thus, there are $43 \times 7 = 301$ input units. Other tested type uses real numbers rather than binary ones for encoding the input. The letters are coded in the form of numbers from the set $\{0.02, 0.04, \ldots, 0.98\}$. The exact coding of a particular letter will be given later.

The networks have one or two output neurons. In the first case, the meaning of the output value is 0 for 'do not hyphenate' and 1 for 'insert hyphenation point'. In the second case the output 0 1 means 'hyphenate', 1 0 'do not hyphenate'.

2.3 Training Set Used

We had a set of 169,888 hyphenated Czech words to experiment with. The problem with this set was considerably large number of errors. There are two types of errors. The first type is probably the worse one—the hyphen is placed in the position where the word cannot be hyphenated. In case of errors of the second type the algorithm is not able to find an allowed hyphenation point. These errors are a big complication of typesetting in narrow columns.

3 Empirical Results—Brute Force Trial

For the first group of experiments the networks with the topologies 301-30-1 and 301-100-1 were employed. In both cases each layer was completely interconnected with the next one. The training set was divided into 170 parts each containing 1000 words. Totally, 1,581,183 training patterns were generated.

The network was trained with each part of the training set. The training was carried out until the network error dropped below the value of 0.1 or until 100 cycles was reached. The learning rate was initially set to the value of 0.7. Then it was stepwise decreased in each training by 0.1 to the final value of 0.3 which was used for the rest of learning.

Naturally, this learning process was extremely time consuming. The training of the network 301-30-1 took about 17 days of user time on Sun SparcStation 10 not taking into account the time for generating patterns. For the training of the network 301-100-1 the supercomputer Silicon Graphics POWER Challenge L was used. Despite its computing power the training took about 18 days of user time.

The results of the experiments described above are summarised in Tables 1 and 2. Although in the latter case the network contained more than three-fold num-

ber of connections, the amount of wrong patterns was almost the same. It is obvious that, in this case, the performance of the network cannot be significantly improved by increasing the number of hidden layer neurons only.

STATISTICS (1581183 patterns)
wrong: 3.43% (54282 patterns)
right:96.57% (1526901 patterns)

Table 1: Results of learning of the network 301-30-1 with 1,581,183 training patterns

STATISTICS (1581183 patterns)
wrong: 3.26% (51599 patterns)
right:96.74% (1529584 patterns)

Table 2: Results of learning of the network 301-100-1 with 1,581,183 training patterns

As stated earlier, the biggest problem with the training of word hyphenation is to obtain a good training set. It seems to be unrealistic to avoid all errors but it is necessary to try to find patterns with the least number of wrongly hyphenated words and with the maximum of correctly marked hyphenation points. The file of 169,888 hyphenated words contained many errors. Therefore, when the network was tested using this file, some correctly hyphenated words were considered erroneous by the system. Remaining errors mainly occurred in words which belong to the exceptions from hyphenation rules, especially in words adapted from foreign languages.

To test if a network can even learn all the exceptions from hyphenation rules, all 54,282 training patterns wrongly hyphenated by the network 301-30-1 were used as one big training set and presented to another network of the type 301-30-1. Learning rate decreased stepwise from the value 0.8 to 0.2. After 100 cycles the network learned all but 4 training patterns which is an excellent result.

4 The Influence of Input Coding: Use of Real Numbers

All the following experiments were carried out with the training set containing 78,809 hyphenated words beginning with the letter `m`. The amount of errors in these data was very low. The number of errors of the first type was negligible and the relative number of errors of the second type was less too. A subset of 1000 words was chosen in which the errors were corrected was used for most of shorter experiments. They were performed with the networks with 7 input layer neurons for 7 consecutive letters. Each letter was coded as a real number. In the beginning, the codes of letters were assigned according to the alphabet (see Table 3).

The results of experiments with the network 7-30-9-2 are summarised in Table 4. It is obvious that these results are not satisfactory as the network error is too high. The network wrongly hyphenated even often used words with simple syllable hyphenation. It did not recognise the rules of making syllables, did not take into ac-

␣	a	á	b	c	č	d
0.02	0.04	0.06	0.08	0.10	0.12	0.14
ď	e	é	ě	f	g	h
0.16	0.18	0.20	0.22	0.24	0.26	0.28
i	í	j	k	l	m	n
0.30	0.32	0.34	0.36	0.38	0.40	0.42
ň	o	ó	p	q	r	ř
0.44	0.46	0.48	0.50	0.52	0.54	0.56
s	š	t	ť	u	ú	ů
0.58	0.60	0.62	0.64	0.66	0.68	0.70
v	w	x	y	ý	z	ž
0.72	0.74	0.76	0.78	0.80	0.82	0.84

Table 3: Coding of letters according to the alphabet

count which letters are vowels and which consonants. Therefore, this approach proved to be inapplicable due to poor generalisation achieved.

```
STATISTICS          ( 8411 patterns )
wrong   : 14.12 %   ( 1188 patterns )
right   : 85.88 %   ( 7223 patterns )
```

Table 4: Results of experiments with the network 7-30-9-2 and coding according to Table 3

In order to improve results, learning with another input coding of letters was used. It is shown in Table 5. All the vowels in Czech were coded as small numbers in the range of $\langle 0.02, 0.28 \rangle$, all the consonants except `r` and `l` as numbers from $\langle 0.50, 0.98 \rangle$. Letters `r` and `l` are consonants but can make syllables in Czech. Therefore, they were coded using using numbers 0.40 and 0.42 separately from the other consonants. Blank was coded as 0.34, i.e. as a number between code numbers of vowels and consonants.

The results of the experiments with the network 7-30-9-2 and the coding described above are shown in Table 6. The comparison of results with both coding alternatives is given in Figure 1.

Information about the type of a letter (consonant or vowel) helped the network to generalise. The different coding of letters `r` and `l` also improved learning. The results with this network could be probably further improved by another sorting of input letter codes. Many errors were caused by a special nature of joined letters `c` and `h`. In Czech they both are used together as a two-character symbol of one sound and, in fact, they form a sort of a single letter. Thus, `c` and `h` cannot be separated by a hyphen. A solution

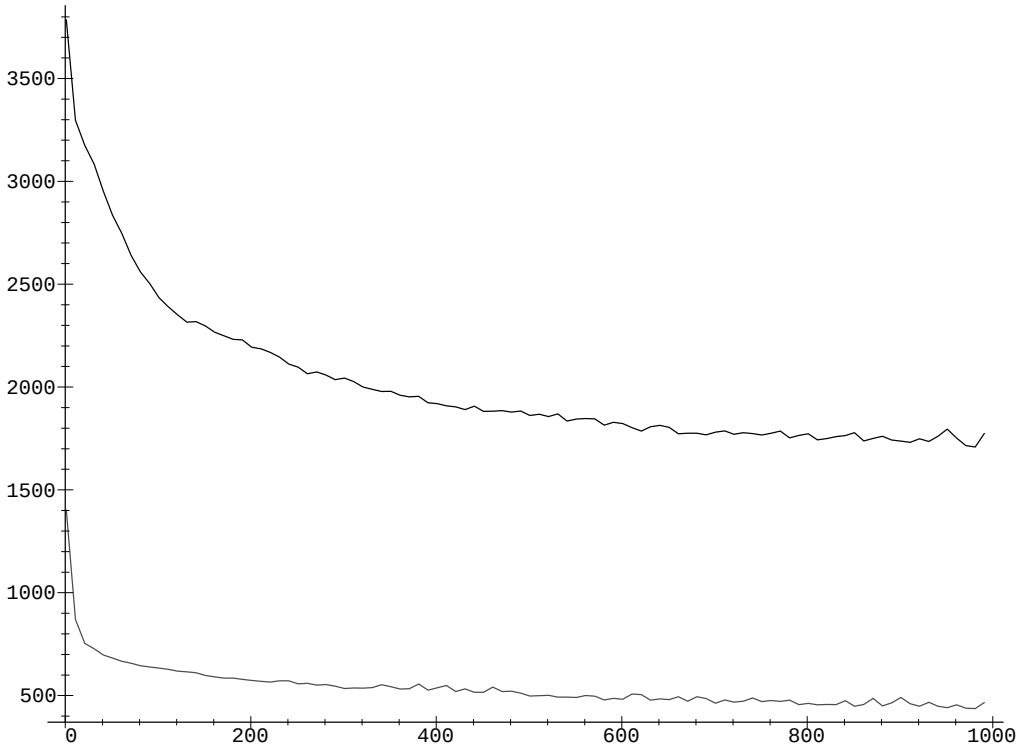

Figure 1: Comparison of the results of experiments with the network 7-30-9-2 using both coding alternatives. Upper curve: Coding according to Table 3. Lower curve: Coding according to Table 5.

a 0.02	á 0.04	e 0.06	é 0.08	ě 0.10	i 0.12	í 0.14
o 0.16	ó 0.18	u 0.20	ú 0.22	ů 0.24	y 0.26	ý 0.28
		␣ 0.34			r 0.40	l 0.42
			b 0.50	c 0.52	č 0.54	d 0.56
ď 0.58	f 0.60	g 0.62	h 0.64	j 0.66	k 0.68	m 0.70
n 0.72	ň 0.74	p 0.76	q 0.78	ř 0.80	s 0.82	š 0.84
t 0.86	ť 0.88	v 0.90	w 0.92	x 0.94	z 0.96	ž 0.98

Table 5: Alternative coding of letters

```
STATISTICS          (8411 patterns)
wrong   :  3.41 %   ( 287 patterns)
right   : 96.59 %   (8124 patterns)
```

Table 6: Results of experiments with the network 7-30-9-2 and coding according to Table 5

of this problem may be to code joined `c` and `h` by a special number differing from the codes of both single letters.

5 Comparison of Various Topologies

Next series of experiments was designed to compare the abilities of networks with different topology. Networks 301-30-1, 301-60-1, 7-30-1, 7-30-9-2, and 7-60-2 were compared. In case of networks with 7 input neurons the alternative coding was used (as described in Table 5). Detailed description of results can be found in (Smrž 1995).

No significant difference in learning performance was observed between the networks with topologies 301-30-1 and 301-60-1. Similar result was obtained earlier using the networks 301-30-1 and 301-100-1 and the other training set (see Section 3).

The results obtained with the network 301-30-1 are distinctly better than those with networks consisting of a less number of neurons and synapses for which different coding was necessary. On the other hand, this network needs much more memory for weight storage.

Learning and generalization performance of a network is significantly influenced not only by the number of hidden layer neurons but also by the network topology. Using the network 7-30-9-2 with two hidden layers, better results were obtained though the total number of neurons and connections was less than that of the network 7-60-2 with only one hidden layer.

Next, the generalisation ability of the networks 7-30-9-2 and 301-30-1 was studied. The networks were trained with the subset of 1000 words. Then the whole set of 78,809 words was used for testing. The results are given in Tables 7 and 8. The percentage of wrongly hyphenated words can be considered very low if the number of errors in the set used for testing is taken into account.

STATISTICS	(648928 patterns)
wrong:	4.91% (31886 patterns)
right:	95.09% (647042 patterns)

Table 7: Generalisation ability of the network 7-30-9-2

STATISTICS	(648928 patterns)
wrong:	2.78% (18057 patterns)
right:	97.22% (630871 patterns)

Table 8: Generalisation ability of the network 301-30-1

To sum up our observation: to train a neural network to perform hyphenation in a language one should:

1. create a training set with proofreaded hyphenated words without errors with all exceptions from generalizable rules
2. make a clever ordering of letters in given language reflecting “similarity”/“exchangability” of letters
3. use a topology with two hidden layers might be cheaper in memory consumtion but learning is then harder

The learning process itself can be used for finding errors in training data (proofreading of words that were not learnt). This gradual bootstrapping process may lead to a perfect network.

6 Syllable Hyphenation

Finally, it was tested how well a network would perform if only the type of letters (consonants or vowels) was given. The network 7-30-1 was used. Consonants were coded as `0`, vowels as `1` and blank as `0.5`. The results of these experiments are given in Table 9. The results clearly show that the syllable hyphenation plays a dominant rôle in Czech language. However, as the error was about 6%, it was obvious that if only the syllable hyphenation was included in the algorithm, the results would be unsatisfactory for everyday use.

```
STATISTICS          ( 8411 patterns )
wrong    :  6.08 %  (  511 patterns )
right    : 93.92 %  ( 7900 patterns )
```

Table 9: Results of experiments with the network 7-30-1 and consonant/vowel coding

7 Discussion

The results obtained for Czech hyphenation are close to those showed in (Sojka 1995, Sojka and Ševeček 1995) for "classical" approach. Testing the "syllable hyphenation neural network" on "close" languages (e.g. syllable ones), preprocessed for accents, gives similar results. This fact allows to build a modular hybrid system, in which separate neural networks will be trained to cover "close" languages, and hyphenation of words not covered by them will be stored in the exception tries in the PATGEN fashion. Such a system is able not only perform well if properly tuned up—in addition—it can be trained to give a measure of suitability of hyphenation points found for the [DTP] system.

8 Conclusion and Acknowledgements

We showed that solving word hyphenation problem with neural networks is possible and that generalisation abilities of neural networks allow to build a working system for given task. Combining with exception lists, we can build a quality system which is able to store hyphenation points for several languages with moderate memory needs.

We acknowledge the possibility to use computer facilities of Supercomputing Centre Brno.

References

Emerson, S. L. and Paulsell, K.: 1987, *troff Typesetting for* UNIX*™ Systems*, Prentice-Hall, Inc., Englewood Cliffs, New Jersey.

Haller, J.: 1956, *Jak se dělí slova (How the words get hyphenated)*, Státní pedagogické nakladatelství Praha.

Knuth, D. E.: 1986, *The TEXbook*, Vol. A of *Computers and Typesetting*, Addison-Wesley, Reading, MA, USA.

Liang, F. and Breitenlohner, P.: 1991, PATtern GENeration program for the TEX82 hyphenator, Electronic documentation of PATGEN program version 2.0 from UNIXTEX distribution at ftp.cs.umb.edu.

Liang, F. M.: 1981, TEX and hyphenation, *TUGboat* **2**(2), 19–20.

Liang, F. M.: 1983, *Word Hy-phen-a-tion by Com-put-er*, PhD thesis, Department of Computer Science, Stanford University.

R. E. Allen: 1990, *The Oxford Spelling Dictionary*, Vol. II of *The Oxford Library of English Usage*, Oxford University Press.

S. Brunak and B. Lautrup: 1990, *Neural Networks: Computers with Intuition*, World Scientific, Singapore.

Sejnowski, T. J. and Rosenberg, C. R.: 1987, Parallel networks that learn to pronounce english text, *Complex Systems* **1**, 145–168.

Smrž, P.: 1995, *Learning algorithms of neural networks*, Master's thesis, Masaryk University, Brno.

Sojka, P.: 1995, Hyphenation in TEX — Quo Vadis?, *TUGboat* **16**(3), 280–289.

Sojka, P. and Ševeček, P.: 1995, Notes on Compound Word Hyphenation in TEX, *TUGboat* **16**(3), 290–297.

Zdeněk Hlavsa et al: 1993, *Pravidla českého pravopisu (The rules of the Czech spelling)*, Academia Praha.