A PHONETIC CONVERTER FOR SPEECH SYNTHESIS IN ROMANIAN

Dragoş Burileanu[*], Mihai Sima[*], and Adrian Neagu[†]
[*]*"Politehnica" University of Bucharest, Romania,*
[†]*Institut de la Communication Parlée, Grenoble, France*

ABSTRACT

Letter-to-phone conversion, as part of the natural language processing stage, plays a very important role in text-to-speech (TTS) synthesis because it associates an appropriate phonetic transcription with each word of the sentence to be pronounced. The classical approach for the phonetic conversion is based in most TTS systems on either a dictionary or a set of rules. Because both methods have important drawbacks, we used a completely different approach: we designed a very accurate grapheme-to-allophone converter for unrestricted TTS synthesis in Romanian, based on a parallel architecture of neural networks. The percentage of correctly transcribed words is extremely high, comparatively with both classical methods and other related implementations. The paper motivates the neural approach, describes the system architecture, gives implementation details and discusses the main results obtained for Romanian language.

1. INTRODUCTION

Usually, a TTS synthesis system includes a natural language processing stage, able to produce a phonetic transcription of the text to be read, information about intonation, stress and duration, as well as a digital signal processing stage, which transforms the symbolic information it receives into speech. The presence of the phonetic transcription module is essential for a TTS system, as it strongly influences the overall accuracy and contributes to the conversion of an unrestricted text into synthetic speech [3, 6].

However, the phonetic conversion is not a trivial task for a TTS system as the pronunciation of words significantly differs from their spelling in many languages. In Romanian language, even if the coarticulation rules are fewer and less restrictive than in English or French, for example, and the heterophonic homographs are also fewer, the number of pronunciation problems and ambiguities is still big enough [4].

Some examples of difficulties for Romanian language are presented below; we mention that we used a phonetic code based on the standardized SAM phonetic alphabet - SAMPA [12].

A. Some graphemes have multiple phonetic values:

A1. The occlusive consonant 'c' is pronounced as follows:

- [k'], when it is followed by 'h': "chema" (to call) – [k'ema];
- [tS], when it is followed by 'e' or 'i': "cine" (who) – [tSine];
- [k], in all other situations: "cap" (head) – [kap].

A2. The front vowel 'e' is pronounced as follows:

- [e_X] (semivowel) in: "perdea" (curtain) – [perde_Xa];
- [je] ([e] preceded by semivowel [j]) in: "el" (he) – [jel];
- [j] (semivowel – short 'i') in: "ea" (she) – [ja];
- [0] (phonetic-zero unit, so it is not pronounced): "geam" (glass) – [dZ0am];
- [e], in all other situations: "erou" (hero) – [erow].

B. A number of consonants change some of their phonological features in a specific context; e.g., the occlusive consonant 'b' is devoiced and pronounced [p]: "absurd" (irrational) – [apsurd].

C. Some graphemes may be reduced or deleted, such as 'ci' in "cincisprezece" (fifteen) – [tSinsprezetSe].

2. MOTIVATION FOR A NEURAL APPROACH

As mentioned, two basic strategies are used for the phonetic conversion of the input text [6]. Some TTS systems store in a dictionary the phonetic transcriptions for the most used morphemes of the language, others use a complete set of grapheme-to-phoneme conversion rules. Many systems actually make a compromise between a set of rules and a pronouncing dictionary. But the development of such a set for a language is usually a very laborious task. Also, the storage of a large dictionary and the time required to identify a word raise many problems.

In order to eliminate these drawbacks, new methods were proposed in recent years, based on a "trainable" phonetic converter general concept [5, 6, 8]. For example, many attempts have been made to use artificial neural networks for solving this task [1, 7, 9, 10, 11]. Unfortunately, their performances were so poor that the general perception is this approach cannot compete with rule-based system [6].

The phonetic converter created by the authors of this paper, also based on a neural network approach, proves that very good performances may be reached, if the system architecture is optimized for this purpose and the complete phonetic features of the language sounds are used.

Our motivation is based on three fundamental reflections. Firstly, it is obvious that grapheme-to-allophone conversion may be seen as mapping the input word graphemes into the elements of a symbolic sequence (allophones), in accordance with some phonetic and phonologic principles; a feed-forward neural network can theoretically do this task.

Secondly, the neural approach is basically language independent. A dictionary of phonetically transcribed words and a phonetic description of fundamental sound units being given,

the retraining of the neural network is straightforward for the new language.

Finally, the lack of detailed phonetic studies for Romanian language and the numerous difficulties for the phonetic conversion guided us to this solution.

3. THE SYSTEM ARCHITECTURE

The phonetic converter presented here is an improved version of our first one [2]. Based on a parallel architecture of neural networks, the present system receives sequences of five letters at the input (the central one being the "target" letter) and provides the articulatory features for the allophone corresponding to the central letter at the output.

Each word of the input text is shifted in the input layer from right to left, until the complete set of articulatory features and consequently the phonetic transcription of the input text are obtained [4].

Due to the insufficient phonetic and linguistic studies for Romanian language, a great deal of effort was made for establishing an optimum set of allophones and a complete acoustic-phonetic description of them. We finally chose a basic set of 33 allophones. We also defined 29 articulatory features like "front", "back", "middle", "open", "closed", "occlusive", "fricative", "liquid", "palatal", "velar", etc., extended with 2 special codes for phonetic-zero unit/word boundaries and one for the primary stress.

The system architecture is presented in Figure 1.

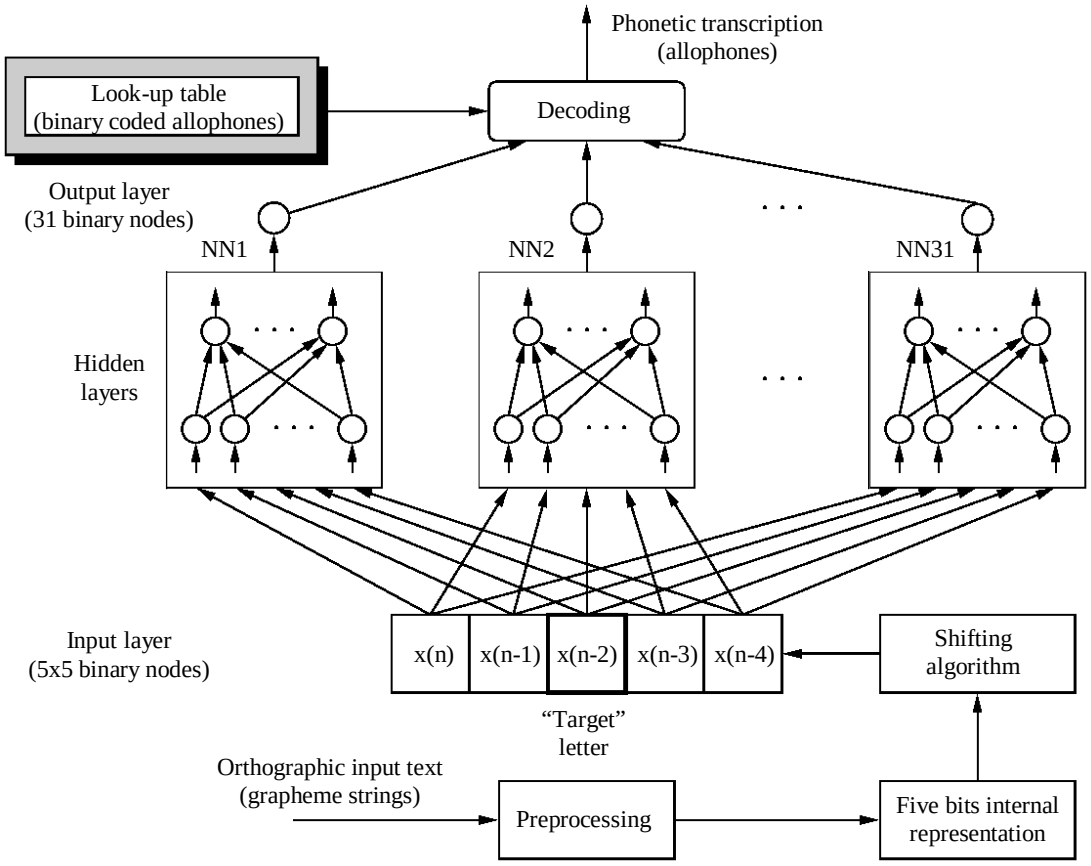

Figure 1. The architecture of the phonetic converter

The phonetic conversion covers the following steps:

1. Preprocessing the input orthographic text

A number of actions are performed in order to accomplish a primary conversion of the letter input strings into a 27 fundamental orthographic character set. Firstly, the substitution of upper cases into lower cases and hyphens removal are done. Then, the replacement of some non-native Romanian letters with graphemes corresponding to their basic phonetic values ('k' with 'c' or 'ch', 'x' with 'cs', 'q' with 'c', 'w' with 'v', and 'y' with 'i') is performed. Finally, 'ch' and 'gh' graphemes are replaced by two ASCII symbols used for the internal representation. The last two actions actually perform a complete alignment of text strings with phonetic ones.

2. Five bits internal representation

A 29 character set (the 27 previously mentioned plus word boundaries and a special character for the primary stress) is five bits coded, as opposed to the seven bits ASCII standard representation. This action led to an important reduction of the number of input nodes of the neural structure from 5×7 to 5×5.

3. Words shifting through the input layer

Each letter is leftward shifted at each step, both in training phase and in testing phase. This algorithm starts with the first

letter on middle position and ends with the final letter on the same middle position; the vacant places are filled with the '# ' symbol (word boundary).

Each sequence of five letters at the input layer will be called the "input vector" hereafter.

An example of this procedure for the word "această" (this) is presented in Table 1. In the second column we show the input vectors corresponding to the word and in the third one the allophone corresponding to the central letter.

Step	Input vector	Allophone
1	# # a c e	[a]
2	# a c e a	[tS]
3	a c e a s	[0]
4	c e a s t	[a]
5	e a s t ă	[s]
6	a s t ă #	[t]
7	s t ă # #	[@]

Table 1. Word shifting through the input layer

4. Running through the neural network structure

The neural structure is a parallel architecture including 31 fully connected feed-forward neural networks (Multi-Layer Perceptron type), one for each articulatory feature. The networks have 25 binary nodes in common as inputs. Each network has a different number of nodes in two hidden layers and one binary output.

The networks are separately trained to learn the attached articulatory feature of the allophone corresponding to the central letter, using all the vectors in the training database. The presence of this feature in the articulatory description of the allophone is pointed out by "1" at the output of the network, while the absence of the feature is pointed out by "0".

The networks work together in the testing (or recognition) phase and provide at the outputs, at each step, the complete articulatory description for the allophone corresponding to the central letter under the form of a 31 binary coded vector.

By means of experiments, we established that including for each letter the influence of four surrounding letters is sufficient to overcome all the difficulties in Romanian language.

5. Getting the final phonetic transcription

In the final step, after obtaining the binary codification for an allophone, a look-up table of coded allophones is used to extract the graphic character of the desired allophone.

4. IMPLEMENTATION DETAILS

4.1. Allophone coding

The allophone set was preliminary coded using their articulatory features with numbers from '2' to '30'. The symbols representing the phonetic-zero unit and the word boundaries were coded with '1' and the primary stress with '31'.

We must emphasize that we firstly used a set of articulatory features based on the available bibliography. But some features were hardly learned by the corresponding neural networks. Even by increasing the number of nodes in hidden layers the number of errors remained big enough. We give the following interpretation: those features may not describe the corresponding allophones very well, as they cannot be fully separated in the n-dimensional space of training vectors. In order to allow a better physical (acoustic and articulatory) modeling of these sounds, additional features are necessary.

For those specific allophones, we consequently propose a new set of articulatory features labeled "type 1", "type 2", etc. The errors for this set of features drop off to insignificant values.

Table 2 shows the first eight lines of the complete codification table [4].

No.	Grapheme	Allophone	Articulatory features	Codes
1.	a	a	open, central	2, 10
2.	ă	@	medium, central	3, 10
3.	e	e	medium, front, type 1	3, 11, 30
4.	short - e	e_X	medium, front, type 4	3, 11, 23
5.	i	i	closed, front, type 2	4, 11, 21
6.	short - i	j	closed, front, type 5	4, 11, 27
7.	î (â)	1	closed, central	4, 10
8.	o	o	medium, back, type 3	3, 12, 22

Table 2. Decimal codification for some allophones

In the next stage, we built a binary codification for the allophones; as mentioned, the presence of the feature in the articulatory description of the allophone was marked with "1" and the absence of the feature with "0". For example:

[a] (2, 10): 01000000010000000000000000000000
[e] (3, 11, 30): 00100000001000000000000000000010
[0], [#] (1): 10000000000000000000000000000000

This binary array was separately stored. During the training phase, the desired allophone is presented and the corresponding line is extracted. By comparing it with the network outputs, the line is further used to compute the error. During the testing phase the network outputs are compared with the array lines in order to extract the corresponding allophone.

4.2. Training and testing the system

Training a neural network-based phonetic converter demands a proper database, which must carry out two basic criteria:

- it must be large enough to cover the usual language words as much as possible and also all the pronunciation difficulties of the language;
- it must be limited so that the training time remains reasonably low.

Therefore, we built from scratch a 5,000-word dictionary phonetically coded by SAMPA symbols, containing the most used words in Romanian language. This dictionary was partitioned into a 4,000-word training database and a 1,000-word testing database.

The neural networks were trained with an improved back-propagation (BKP) algorithm, the back-propagation of errors from the output layer and weighs modification being done after

an entire epoch (“batch”-type training). We used momentum, an adaptive learning rate and an error criterion based on the global mean-square error. For most neural networks, the training stopped after an acceptable global error was reached. However, for some of them it was necessary to reinitialize the weights or to apply a "cross-validation" method (training was stopped when the number of erroneous vectors from the testing database begun to grow up, so when the networks started to lose their generalization ability).

It is important to observe that by separately training each network, the performances of the entire system dramatically increased for the following reasons:

- each network was optimized in terms of training method and number of nodes in the hidden layers;
- due to the small number of nodes and weights, the training time was reasonably low for each network (hours);
- we were able to precisely determine which of the initially proposed articulatory features generated large errors; we consequently proposed a new set of features.

We also mention that the training time ranged from dozen minutes for the small networks (14×7 nodes in hidden layers and a medium of 100 training epochs) to 20 hours for the large networks (37×27 nodes in hidden layers and a medium of 4,000 training epochs).

After the training phase and weights saving, the system runs in the normal (testing) phase accordingly with the procedure described in section 3. By running forward through the neural structure, the allophone corresponding to the central letter of each input vector is obtained by comparing the network outputs with the binary coded look-up table. For the feature combinations not found in the table, the decision is taken by means of a minimum-distance criterion. Therefore, only the errors resulting from confusions between allophones (i.e. a feature combination corresponding to an incorrect allophone) will remain.

4. RESULTS AND CONCLUSIONS

We further present (Table 3) the overall performances of the phonetic converter. We show all the allophones that were not correctly identified, by testing both the training database (Tr-DB) and testing database (Te-DB).

We generally obtained no more than one error per word (an incorrect allophone or a wrong position of the primary stress). The errors in Table 3 and the errors for stress positioning (9 in training + 4 in testing corpora) add up to a total of 61 in training and 29 in test. This is a very high rate of correctly transcribed words: over 98.4 % of good word transcription for the training set and over 97.1 % for the testing set. Error rate will be further diminished by a small exception dictionary. The phonetic conversion time of a medium 6-letter word is less than 80 ms for a standard PC configuration. We think that these results are very favorable and prove the validity of our approach.

We must emphasize that the overall TTS system built by our team is structured as a set of C-modules and runs in real time under Windows. Based on a diphone-concatenation scheme, the TTS system accepts Romanian language sentences for input and generates good quality speech.

Desired allophone	Obtained allophone	Number of errors	
		Tr-DB	Te-DB
i	j	12	7
j	i	7	3
e_X	e	10	3
e	e_X	—	1
u	w	2	2
w	u	5	2
(tS)i	(tS)0	6	3
(tS)0	(tS)i	1	—
(dZ)i	(dZ)0	2	2
(k')i	(k')0	1	—
(k')0	(k')i	—	1
1	0	2	—
@	0	1	—
tS	0	1	1
dZ	0	2	—

Table 3. Phonetic transcription errors

REFERENCES

[1] Ainsworth, W. A. and B. Pell. 1989. Connectionist Architectures for a Text-to-Speech System. In *Proceedings of EUROSPEECH*, p.125-128.

[2] Burileanu, D. and S. Marinescu. 1996. A Neural System Architecture for a Text-to-Speech System in Romanian language. In *Proceedings of ICSPAT*. Boston, p.1363-1367.

[3] Burileanu, D. et al. 1997. Text-to-Speech Synthesis for Romanian Language: Present and Future Trends. In Tufis, D. (ed.), *Recent Advances in Romanian Language Technology*. Bucharest: Romanian Academy.

[4] Burileanu, D. 1998. Contributions on Speech Synthesis from Text in Romanian Language. *PhD Dissertation*. Bucharest: "Politehnica" University.

[5] Daelemans, W. and A. V. D. Bosh. 1997. Language-Independent Data-Oriented Grapheme-to-Phoneme Convertion. In Van Santen, J. P. et al. (ed.), *Progress in Speech Synthesis*. New York: Springer-Verlag.

[6] Dutoit, T. 1997. *An introduction to Text-to-Speech Synthesis*. Kluwer.

[7] Gubbins, P. R. and K. M. Kurtis. 1995. Neural Network Solutions for Improving English Text-to-Speech Transcription. In *Proceedings of the International Conference on Phonetic Science*. Stockholm, p.314-317.

[8] Jiang, L., H. W. Hon and X. Huang, 1997. Improvements on a Trainable Letter-to-Sound Converter. In *Proceedings of EUROSPEECH*. Rhodes, p.605-608.

[9] Karaali, O. et al. 1997. Text-to-Speech Conversion with Neural Networks: A Recurrent TDNN Approach. In *Proceedings of EUROSPEECH*. Rhodes, p.561-564.

[10] Lucas, S. M. and R. I. Damper. 1990. Text-to-Phonetics Translation with Syntactic Neural Nets. In *Proceedings of the ESCA Workshop on Speech Synthesis*. Autrans, p.87-91.

[11] Sejnowski, T. J. and C. R. Rosenberg. 1987. Parallel Networks that Learn to Pronounce English Text. In *Complex Systems*, vol.1, p.145-168.

[12] Wells, J. 1993. *Computer-Coding the IPA: A Proposed Extension of SAMPA*. University College, London.