INTERNATIONAL JOURNAL OF SPEECH TECHNOLOGY 5, 211–225, 2002

Basic Research and Implementation Decisions for a Text-to-Speech Synthesis System in Romanian

DRAGOS BURILEANU
Speech Technology and Signal Processing Laboratory, Faculty of Electronics and Telecommunications, "Politehnica" University of Bucharest, Romania
bdragos@mESsnet.pub.ro

Abstract. Speech synthesis is one of the most language-dependent domains of speech technology. In particular, the natural language processing stage of a text-to-speech (TTS) system contains the largest part of the linguistic knowledge for a given language. In this respect, one can state that building a high-quality TTS system for a new language involves many theoretical and technical challenges. Especially, extensive studies must exist (or be done) at the linguistic level, in order to endow the system with the most relevant language information; this requirement represents an essential condition to obtain a true naturalness of the synthesized speech, starting from unrestricted input text. This paper presents fundamental research and the related implementation issues in developing a complete TTS system in Romanian, emphasizing the language particularities and their influence on improving the language processing stage efficiency. The first section describes our standpoint on TTS synthesis as well as the overall architecture of our TTS system. The next sections formulate several important tasks of the natural language processing stage (input text preprocessing, letter-to-phone conversion, acoustic database preparation) and discuss the design philosophy of the corresponding modules, implementation decisions and evaluation experiments. A distinct section is devoted to an acoustic-phonetic study that assisted the phone-set selection and acoustic database generation. The paper ends with conclusions and a description of the work that is currently in progress at other levels of the TTS system.

Keywords: text-to-speech synthesis, natural language processing, linguistic knowledge, diphone concatenation

1. Introduction

After several decades of research in speech synthesis, many highly intelligible TTS systems are now available. At the same time, the continuously increasing number of applications involving speech synthesis (for example in telecommunications, embedded systems and personal computing) shows clearly that the industry has a proper reaction to the present quality of the TTS systems and also to the customers' constant demand for more natural and efficient communication means and access to information. One must remember that to freely converse with a machine has been for some time a steady dream of human beings.

Despite this evident market response, it is also obvious that machines are still far from talking as naturally as a human speaker. The current TTS systems are not able to produce speech indistinguishable from a human voice, let's say just "relatively close" to it. Without any doubt, we are not ready yet to build a system equipped with a "perfect" voice, and many research activities in a variety of areas (including basic understanding of the production and perception of speech) remain to be done (D'Alessandro et al., 1996).

But one may wonder if it is really necessary to attain this goal by any means, and no matter what would be the effort. The author of this study is convinced that the answer is no, at least for the time being. Even if one would get very close to this target, the resulting system would be highly inefficient in terms of costs and computational resources.

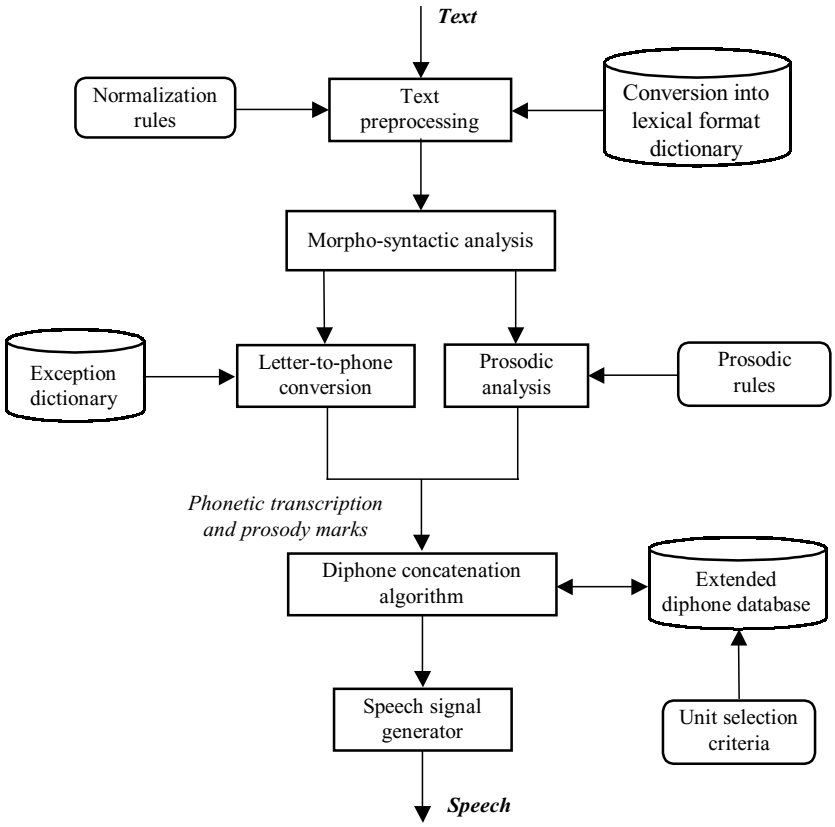

Figure 1. The architecture of the Romanian TTS system.

On the other hand, building TTS systems for new languages (as is the case for the Romanian language), has become an obligation for the speech synthesis research community; the multilingual nature of our society demands that the TTS synthesis challenge needs to be met not only for English, or French, but for multiple languages.

In this respect, our essential idea is that the design process of a complete TTS system for a new language must be conducted simultaneously in the following two directions:

1. A basic structure of the TTS system must be developed; this structure must be a framework both for fundamental research and for further increasing performance at different levels of processing.
2. Extensive linguistic studies must be completed in order to endow the system with the most relevant language information; this requirement represents an essential condition to obtain true naturalness of the synthesized speech in the given language, starting from unrestricted input text.

The above considerations guided us in developing our concatenative TTS system for Romanian. A basic structure was first built in our laboratory; then, continuous work was carried out to improve the performance of different constituent modules and consequently enhance the quality of the system.

The architecture of the system is presented in Fig. 1. It generally follows the already "classical" structure of a TTS system and consists of a (natural) language processing stage that produces a phonetic transcription of the input text and a number of symbolic marks for prosody control and a signal processing stage that transforms the information received into speech. The system includes a two-level parser for preprocessing and syntactic/prosodic analysis, a neural network-based letter-to-phone converter, a unit selection algorithm based on minimum distortion criteria, and a modified PSOLA algorithm for diphone concatenation and speech signal generation.

The aim of this paper is to present fundamental research and the related implementation issues in developing our TTS system for the Romanian language. The

following sections discuss several important tasks of the language processing stage: input text preprocessing (Section 2), letter-to-phone conversion (Section 4), acoustic database preparation (Section 5), and describe the design philosophy behind the corresponding modules, implementation decisions and evaluation experiments. A distinct Section 3 is devoted to an acoustic-phonetic study that assisted the phone-set selection and acoustic database generation. Throughout the paper, many Romanian language particularities are highlighted, and their influence on our implementation options (so as to improve the language processing stage efficiency) are also emphasized.

2. Input Text Preprocessing

Text preprocessing plays an important role in a TTS synthesis system. The correct detection and interpretation of incoming strings influence the overall system accuracy and contribute to the conversion of an unrestricted text into synthetic speech. Usually, written texts are presented as strings of ASCII characters; they consist of orthographic words and symbols such as punctuation marks, number strings, or mathematical operators. One may encounter numerals (for example, in Romanian, 'al 2-lea'—*the 2nd*, '24', '24.530', '2,453'), abbreviations ('Prof.', 'ms', 'ing.'—for *engineer*, 'dl'—for *mister*), and acronyms ('IBM', 'S.R.L.', 'TTS'). These strings may be considered "anomalous" (from a linguistic point of view) relative to the majority of text words, and have to be converted into a suitable orthographic format before any other linguistic analysis. This conversion represents the task of the preprocessing module. This task also includes word and sentence boundary detection, as well as punctuation marks and special symbols processing (Dutoit, 1997; Lindstrom and Ljungqvist, 1994).

Text preprocessing for a TTS system is a difficult task in any language. In Romanian, the difficulties include, for example, multiple functions for the period, comma and colon, and different pronunciations for numerals according to case, number and gender (Burileanu, 1999). For example, the period (.) may occur in abbreviations (e.g., 'tel.', 'etc.', 'P.S.'), acronyms ('S.R.L.'—for *Ltd.*), digit sequences ('1.234', '1.234,567', '25.03.2002'), and sequences of three indicating, for example, that a text fragment will be omitted (. . .), or may indicate the end of a sentence. Ambiguities created by the period are a major problem for the preprocessing task, especially in the case when the period carries out multiple functions (e.g., the period after an abbreviation may also indicate the sentence-end). Other difficult situations are generated by hyphens, mathematical expressions, etc.

For our TTS system, we proposed a set of definitions and preprocessing rules, based on a detailed analysis of the most common use of punctuation marks, lower and upper cases and digit sequences. The basic proposed definitions are given below.

D1. A "word" is a sequence of lower case letters.
D2. An "expression" is a sequence of characters that contains one or more of the following categories: letters sequence with at least one upper case, digit sequence; punctuation marks, and other special symbols.
D3. An "extra-textual symbol" is a symbol performing a punctuation function.
D4. An "inter-textual symbol" is a symbol that belongs to an expression and helps with its spelling.
D5. "Expansion" is the process of conversion of an expression to its full textual form.
D6. An "ambiguous character sequence" is a sequence of characters that may affiliate with more than one linguistic class.

According to those definitions, we designed a preprocessing algorithm that consists mainly of three basic steps:

1. *Text segmentation.* The input text is segmented into *character groups*, from left to right. We obtain strings of ASCII characters delimited by white space characters; punctuation is temporarily included in these groups.
2. *Conversion of anomalous symbol strings into orthographic characters.* We defined an original rule set for the Romanian language regarding the detection of basic punctuation marks (. , *? ! : ;* ... - */ ' " ( ) [ ] { }*) and mathematical operators, as well as numerals, abbreviations and acronyms expansion. For example, some fundamental rules used for the correct detection and interpretation of periods were presented in Burileanu et al. (1999b). The character groups obtained after the first step are transcribed into orthographic characters (when necessary), based on a crude contextual analysis at the word/sub-word level, and a number of dictionaries for abbreviations and unusual acronyms.
3. *Interpretation of certain punctuation marks.* The text preprocessor also detects and memorizes the

position of certain punctuation marks, which will be used by morpho-syntactic and prosodic modules.

We implemented the preprocessor using the standard *lex/flex* and *yacc/bison* lexer and parser generators. These programs generate a very efficient C code for the preprocessor and also provide a standard and eloquent conversion rules description for the three tasks to be performed (described above).

The *flex*-generated lexer performs the input text segmentation and identifies "words" and "expressions". "Expressions" are further separated according to their type (number of upper cases and inter-textual symbols), providing appropriate information to the parser. It also transforms numbers, dates and hours into their character sequences using dedicated C routines. Detection of such digit and punctuation strings that follow strict rules is best suited for lexer processing.

All "words" and "expressions" identified by the lexer are first looked up in abbreviation dictionaries according to their forms, and the corresponding information is provided to the parser. We use several dictionaries grouping abbreviations based on their forms. For example, one dictionary contains only words that are written using lower case letters (e.g., 'dna'—*misses*); another contains words that use both lower and upper case letters (e.g., measurement units like 'Hz' and 'mA'), another with dotted abbreviations (e.g., 'P.S.', 'etc.'), etc. Using more than one dictionary and grouping words based on their respective writing manner accelerates the search process. When needed, the dictionary indicates the correct spelling for each abbreviation, including case, number and gender forms. A similar processing sequence is used for acronyms. However, we must note that most acronyms do not require supplementary textual information. Usually, their pronunciation amounts to the sequential reading of their constituent letters, using simple transcription rules (for example, 'S.R.L. = serele'), or to a common reading of words, when the pronunciation in their compact form has become widely accepted (as for 'NATO', or 'TAROM'—*Romanian Airlines*).

The *bison*-generated parser identifies relationships between different types of words using the information provided by the lexer (numeral, abbreviation, acronym, etc.). This allows, for example, the correct spelling of a number followed by an abbreviated unit measure. Note that in the Romanian language, between a number greater than 20 and the numbered objects, an extra 'de' must be inserted: '19 km = nouăsprezece kilometri' and '20 km = douăzeci de kilometri'. The number spelling also differs according to the gender of the object.

Using *flex* and *bison* we were able to concentrate on specification issues, transferring parts of the C code generation task to the two programs. An important effort was devoted to balance the set of tasks to be done by each program.

An error recovery mechanism allows the preprocessor to be tolerant of some typical syntax errors, like a phrase beginning with a lower case letter, or end detection of input file before identifying a punctuation mark that concludes the phrase. Incorrect format for dates and numerals, for example, does not cause preprocessor abnormal termination; they are translated in their closest spelling form.

However, we must note that our preprocessor (together with linguistic analysis) cannot solve yet all possible text ambiguities or ungrammatical constructions (even if, for example, unexpected special symbols are ignored). We evaluated the overall system accuracy on a large set of newspaper and scientific texts; the number of errors due to the preprocessor module was lower than one percent.

The number of unusual character sequences in a text that must be translated into speech obviously depends on the type and subject of text and on the writing correctness (Ferri et al., 1997; Liberman and Church, 1992). Since the writing rules are not widely known in detail by the majority of people, we think that a way to enforce correct writing is to include our preprocessor (or even our entire TTS system) in a spelling checker program that will let the user decide in the case of any ambiguity.

3. The Phone-Set Selection and Duration Considerations

Due to insufficient phonetic and linguistic studies for the Romanian language, important effort was directed towards establishing an optimum set of phones for our TTS system and a complete acoustic-phonetic description of them.

Many experiments had been carried out during several months. A great number of utterances and also words spoken in isolation were systematically investigated (by means of waveforms and spectrograms), with the following targets:

- to deduce a representative (and efficient for synthesis purposes) set of phones;

- to study the behavior of this phone-set under various conditions (e.g., position in word, segmental context, stress, word length, manner of pronunciation);
- to analyze the durations of the selected phones (typical values and dispersion according to contextual factors) and then to understand some of the compression/expansion phenomena at the phonetic level; and
- to infer appropriate algorithms for timing control in the speech generation stage.

This work was an invaluable aid not only for the synthesis algorithm itself, but also for handling the unit segmentation in the database generation procedure (as further discussed in Section 5). In addition, it has provided good support for present intensive prosody studies.

Starting (in a preliminary version of the system) with 38 phones, we finally chose a basic set of *33 phones*. This set was designed according to the previously outlined study and language particularities, considering the trade-off between using many phones to generate a better quality of speech and using a reduced set to significantly spare computing resources and simplify several algorithms (presented in Section 4). The decision was also made by comparing many TTS systems built for different languages.

We used for this basic set a phonetic code based on the standardized SAM phonetic alphabet—SAMPA (Wells et al., 1992). We mention that we introduced two new symbols (for the graphemes 'ch' and 'gh'), given the version adopted in 1996. This phonetic code is used throughout this paper.

Table 1 presents duration values for the basic phone-set (several hundreds of realizations for each phone were investigated), and Figs. 2 and 3 illustrate the waveforms and spectrograms for two Romanian words (phones are separated by thick vertical lines).

The study showed clearly that realizations of many phones in the Romanian language greatly depend on their context and position, and known phenomena like *assimilation* and *devoicing* frequently occur. Some of these situations are presented below:

– short vowels (like /e/, /i/, and /1/) and semivowels (/e_X/, /j/, /o_X/, and /w/) are extremely dependent or their context;
– long vowels (/a/, /@/, /o/, and /u/), unvoiced fricatives (like /f/, /h/, /s/, and /S/) and nasals (/m/ and /n/) are more stable and less context dependent, but fricatives generally influence neighboring sounds;
– semivowels lack a unique spectral representation and usually serve as transitions between two sounds; for example, they are very difficult to delimit from vowels in diphthongs;
– some consonants, like the voiced occlusives /b/, /d/, /g/, and /g'/, are strongly influenced by the next vowel;

Table 1. Duration values for the basic phone-set in the Romanian language.

			Duration (ms)		
No.	Grapheme	Phone	Minimal value	Maximal value	Mean value
1.	a	/a/	70	150	100
2.	ă	/@/	75	145	105
3.	e	/e/	70	112	92
4.	short—e	/e_X/	40	81	55
5.	i	/i/	62	119	85
6.	short—i	/j/	50	106	78
7.	î (â)	/1/	40	78	61
8.	o	/o/	70	136	99
9.	short—o	/o_X/	38	80	56
10.	u	/u/	68	131	98
11.	short—u	/w/	54	120	88
12.	b	/b/	70	115	94
13.	c	/k/	59	150	86
14.	c(e,i)	/tS/	81	220	127
15.	ch	/k'/	80	126	101
16.	d	/d/	100	163	123
17.	f	/f/	89	164	129
18.	g	/g/	70	115	96
19.	g(e,i)	/dZ/	100	152	131
20.	gh	/g'/	108	172	143
21.	h	/h/	91	153	122
22.	j	/Z/	91	140	112
23.	l	/l/	50	108	77
24.	m	/m/	72	120	98
25.	n	/n/	74	152	101
26.	p	/p/	74	133	105
27.	r	/r/	40	93	63
28.	s	/s/	94	182	136
29.	ş	/S/	123	202	169
30.	t	/t/	55	110	95
31.	ţ	/ts/	115	171	150
32.	v	/v/	67	134	93
33.	z	/z/	70	145	103

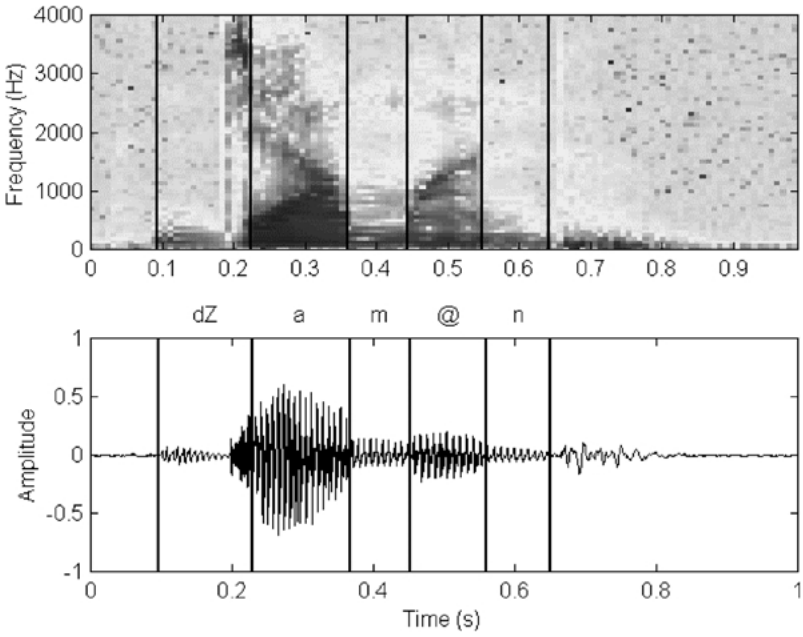

Figure 2. Waveform and spectrogram for the word 'geamăn' (*twin*)—/dZ a m @ n/.

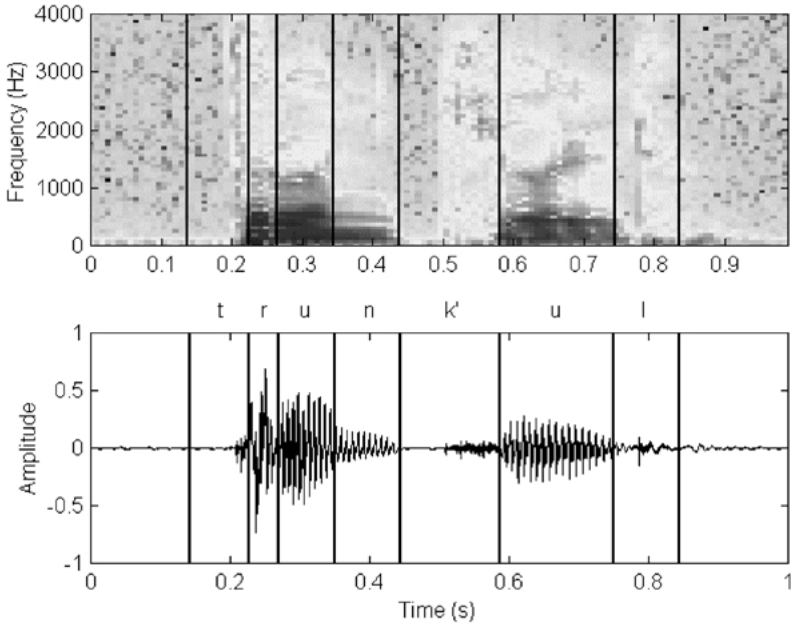

Figure 3. Waveform and spectrogram for the word 'trunchiul' (*trunk*)—/t r u n k' u l/.

– some voiced fricatives (/Z/, /v/, and /z/) are partially devoiced when preceded by an unvoiced occlusive (like /k/ or /t/); another example: the occlusive consonant /b/ before the unvoiced fricative /s/ is devoiced and pronounced /p/;
– the liquid /l/ and the vibrant /r/ are often influenced by the preceding vowel;
– unvoiced fricatives are sometimes difficult to distinguish from the silence regions at word boundaries; the spectrogram is useful here

since it show the spectral energy at higher frequencies;
- stressed vowels are usually 1.2 times longer than unstressed ones;
- fricatives are generally longer than occlusives;
- some prepalatal and velar consonants (like /tS/ and /k/), whose point of articulation is near the hard or soft palate, are lengthened when preceded by front vowels;
- the compression of some phoneme durations for long (polysyllabic) words in natural speaking, which is a known phenomenon in many languages, is also a common event in the spoken Romanian language, but it greatly depends on the speaker rate, style and emotion.

4. Letter-to-Phone Conversion

The *phonetic transcription module* is essential for any TTS system, as the pronunciation of words significantly differs from their spelling in many languages (Boula de Mareüil et al., 1998). In Romanian, even if the coarticulation rules are fewer and less restrictive than in English or French, for example, and the heterophonous homographs are also fewer, the number of pronunciation problems and ambiguities is still large (Burileanu, 1999).

A few examples of difficulties for the phonetic conversion task in Romanian are presented below.

(a). *Some graphemes have multiple phonetic values*

a1. The occlusive consonant 'c' is pronounced as follows:
- /k'/, when followed by 'h': 'chema' (*to call*)—/k'ema/, 'unchi' (*uncle*)—/unk'/;
- /tS/, when followed by 'e' or 'i': 'ceas' (*hour*, *clock*)—/tSas/, 'cine' (*who*)—/tSine/;
- /k/, in all other situations: 'cap' (*head*)—/kap/, 'acasă' (*home*)—/akas@/.

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/, 'erau' (*they were*)—/jeraw/;
- /j/ (semivowel—short 'i') in: 'ea' (*she*)—/ja/, 'aceea' (*that one*)—/atSeja/;
- /0/ (phonetic-zero unit, so it is not pronounced): 'geam' (*glass*)—/dZam/;
- /e/, in all other situations: 'erou' (*hero*)—/erow/.

a3. The occlusive consonant 'g' is pronounced as follows:
- /g'/, when followed by 'h': 'ghem' (*ball of thread*)—/g'em/, 'unghi' (*angle*)—/ung'/;
- /dZ/, when followed by 'e' or 'i': 'geană' (*eyelash*)—/dZan@/, 'agil' (*agile*)—/adZil/;
- /g/, in all other situations: 'gară' (*railway station*)—/gar@/, 'gât' (*neck*)—/g1t/.

(b). *A number of consonants change some of their phonetic features in a specific context*

b1. The occlusive consonant 'b' is devoiced and pronounced /p/: 'absurd' (*irrational*)—/apsurd/;

b2. The occlusive consonant 't' is devoiced and pronounced /d/: 'batjocură' (*affront*)—/badZocur@/;

b3. Some unvoiced consonants becomes voiced; e.g., 'c' is pronounced 'g' in 'acvariu' (*aquarium*)—/agvarju/.

(c). *Some graphemes may be reduced or deleted*

c1. 'ci' in 'cincisprezece' (*fifteen*)—/tSinsprezetSe/;

c2. 'n' in 'înnoda' (*to knot*)—/1noda/.

The classical approach for the phonetic conversion of input text is based in most TTS systems on either a dictionary or a set of rules. Some systems store in a dictionary the phonetic transcriptions for the most used morphemes of the language, while others use a complete set of grapheme-to-phoneme conversion rules (Dutoit, 1997). However, the development of a set of rules for a language is usually a very laborious task; the storage of a large dictionary and the time required identifying a word also raise many problems. Many systems actually make a compromise between a set of rules and a pronunciation dictionary.

In order to eliminate these drawbacks, new methods were proposed in recent years, based on a *trainable phonetic converter* general concept (Daelemans and van den Bosh, 1997; Dutoit, 1997; Jiang et al., 1997). For example, attempts have been made to use *artificial neural networks* for solving this task (Ainsworth and Pell, 1989; Gubbins and Kurtis, 1995; Karaali et al., 1997; Sejnowski and Rosenberg, 1987). Unfortunately, they performed rather poor, the general perception

being that such an approach cannot compete with rule-based systems (Dutoit, 1997).

Nevertheless, we used a neural-network approach for the phonetic transcription task in our TTS system. The motivation for the connectionist approach was based on three fundamental reflections:

1. Grapheme-to-phone conversion represents a typical *classification* problem: the input-word graphemes are mapped into the elements of a symbolic sequence (phones) in accordance with some phonetic and phonologic principles. In other words, phonetic conversion may be seen as a function (generally non-linear) that transforms orthographic character groups (words) into phonetic character sequences that represent words' pronunciation.

   On the other hand, artificial neural networks (and especially *feed-forward* networks) are known to be excellent alternatives to classic pattern classification problems. In this respect, their ability *to adapt*, *learning* from examples (pairs of features and desired responses), and *to generalize* the learned information is the most useful attribute.

   Consequently, a feed-forward neural network can accomplish theoretically the phonetic conversion task. However, we showed that very good performance can be reached only if the neural system architecture is optimized for this purpose, and the complete phonetic features of the language sounds are used.
2. The connectionist approach is conceptually language independent. Given a dictionary of phonetically transcribed words and a phonetic description of fundamental sounds, the retraining of the neural network is straightforward for the new language.
3. The lack of detailed linguistic and phonetic studies for the Romanian language and its numerous pronunciation difficulties make the task of establishing a complete set of rules for the phonetic conversion very complex. These facts reinforce the argumentation to use a trainable phonetic converter.

The phonetic converter developed for our TTS system is based on a parallel structure of neural networks. In fact, the actual configuration represents an improved version of the one described in Burileanu et al. (1999a). It uses the same basic strategy, but we increased the length of the dictionary used for training and testing. Also, we added a small exception dictionary as well as a user-friendly interface that facilitates retraining and allows an easier change of system parameters.

The system receives sequences of orthographic characters (grapheme strings), uses a fixed-length frame of five letters (the central one being the "target" letter) as the (common) input layer of the neural structure, and provides the *articulatory features* for the phone corresponding to the central letter at the outputs. Each word of the input text is then shifted through the input layer from right to left, until the complete set of articulatory features and then the phonetic transcription of each word are obtained.

The architecture of the phonetic converter is depicted in Fig. 4.

We used the basic set of 33 phones described in Section 3. We also defined 29 articulatory features like *front*, *back*, *middle*, *open*, *occlusive*, *fricative*, *liquid*, *palatal*, etc., together with a special code for phonetic-zero unit and word boundary.

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-item fundamental orthographic character set. Initially, 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. Then, 'ch' and 'gh' graphemes are replaced by two ASCII symbols used for the internal representation. These actions perform a complete alignment of text strings with their phonetic counterparts. Finally, a 28-character set (the 27 previously mentioned plus a special character for word boundary) is five-bit coded, as opposed to the seven-bit ASCII standard representation. This action led to an important reduction of the number of input nodes in the neural structure from 5×7 to 5×5.

   Because the phonetic converter was also designed to work independently of the synthesis system (for research purposes), the preprocessor can execute a few additional operations: the substitution of upper cases into lower cases, the removal of hyphens and other punctuation marks, input text segmentation into words, etc.
2. *Word-shifting through the input layer.* Each letter of a word is shifted to the left at each step, both in the training phase and the testing phase. This algorithm starts with the first letter in the middle position and ends with the final letter in the same middle position; the vacant places are filled with the '#' symbol (word boundary). Each sequence of

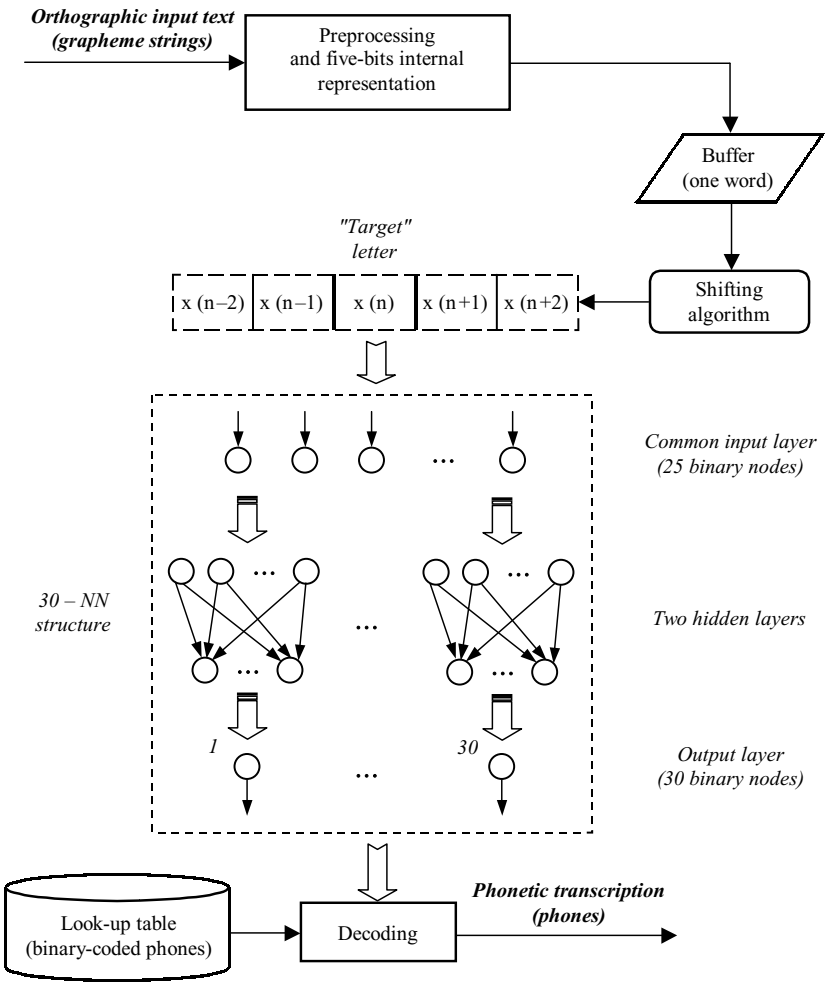

Figure 4. The architecture of the phonetic converter.

five characters 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 2. The second column shows the input vectors corresponding to the word and the third one the phone corresponding to the central letter.

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.

3. *Running through the neural network structure.* The neural structure is a parallel architecture including 30 fully connected feed-forward neural networks (*Multilayer Perceptron* type), one for each articulatory feature. The 25 binary inputs are common to all the networks. 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 phone corresponding to the central letter, using all the vectors in

Table 2. An example of the word-shifting procedure.

Step	Input vector	Phone
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** # #	/@/

the training database. The presence of this feature in the articulatory description of the phone is delineated as "1" at the output of the network, while the absence of this feature is delineated as "0".

The networks work together in the testing (or recognition) phase and provide as output at each step the complete articulatory description for the phone corresponding to the central letter in the form of a 30 binary-coded vector.

4. *Obtaining the final phonetic transcription.* In the final step, after obtaining the binary codification for a phone, a look-up table of coded phones is used to extract the graphic character of the proper phone.

Each phone was temporarily coded using its articulatory features with decimal numbers from "2" to "30". The symbols representing the phonetic-zero unit and the word boundary were coded as "1".

We must emphasize that we initially 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 unacceptably large. We give the following interpretation: those features may not describe the corresponding phones very well, because they cannot fully be separated in the n-dimensional space of training vectors. In order to allow a better physical modeling of these sounds, some features need to be removed, and a few new features are necessary.

For those specific phones, we proposed 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 3 shows the complete (decimal) codification table for the phone set.

In the next stage, we built a binary codification table for the phones. As mentioned previously, the presence of the feature in the articulatory description of the phone was marked with "1" and the absence of the feature with "0". For example:

/a/ (2, 10):	01000000010000000000000000000
/e/ (3, 11, 30):	00100000001000000000000000001
/r/ (9, 17, 25, 28):	00000000100000001000000010010
/0/, # (1):	10000000000000000000000000000

This binary array was separately stored. During the training phase, the correct phone 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 phone.

We then built a 7,000-word dictionary phonetically coded by SAMPA symbols, containing the most used words in Romanian language. This dictionary was partitioned into a 5,000-word training database and a 2,000-word testing database.

Table 3. Decimal codification for the phone set.

No.	Phone	Articulatory features	Codes
1.	a	Open, central	2, 10
2.	@	Medium, central	3, 10
3.	e	Medium, front, type 1	3, 11, 30
4.	e_X	Medium, front, type 4	3, 11, 23
5.	i	Closed, front, type 2	4, 11, 21
6.	j	Closed, front, type 5	4, 11, 27
7.	1	Closed, central	4, 10
8.	o	Medium, back, type 3	3, 12, 22
9.	o_X	Medium, back, type 6	3, 12, 24
10.	u	Closed, back, type 3	4, 12, 22
11.	w	Closed, back, type 6	4, 12, 24
12.	b	Occlusive, bilabial, voiced	5, 13, 25
13.	k	Occlusive, velar, unvoiced	5, 14, 26
14.	tS	Semiocclusive, prepalatal, unvoiced	6, 15, 26
15.	k'	Occlusive, palatal, unvoiced	5, 16, 26
16.	d	Occlusive, dental, voiced	5, 17, 25
17.	f	Fricative, labio-dental, unvoiced	7, 18, 26
18.	g	Occlusive, velar, voiced	5, 14, 25
19.	dZ	Semiocclusive, prepalatal, voiced	6, 15, 25
20.	g'	Occlusive, palatal, voiced	5, 16, 25
21.	h	Fricative, laryngeal, unvoiced	7, 19, 26
22.	Z	Fricative, prepalatal, voiced	7, 15, 25
23.	l	Liquid, lateral, voiced, oral	8, 20, 25, 28
24.	m	Occlusive, bilabial, voiced, nasal	5, 13, 25, 29
25.	n	Occlusive, dental, voiced, nasal	5, 17, 25, 29
26.	p	Occlusive, bilabial, unvoiced	5, 13, 26
27.	r	Vibrant, dental, voiced, oral	9, 17, 25, 28
28.	s	Fricative, dental, unvoiced	7, 17, 26
29.	S	Fricative, prepalatal, unvoiced	7, 15, 26
30.	t	Occlusive, dental, unvoiced	5, 17, 26
31.	ts	Semiocclusive, dental, unvoiced	6, 17, 26
32.	v	Fricative, labio-dental, voiced	7, 18, 25
33.	z	Fricative, dental, voiced	7, 17, 25
34.	0	(Phonetic-zero unit)	1
35.	#	(Word boundary)	1

The neural networks have been trained with an improved *back-propagation* algorithm, the back-propagation of errors from the output layer and weight modification being done once an 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 began to increase and therefore the networks started to lose their generalization ability).

It is important to observe that by separately training each network, the performance 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); and
- we were able to determine precisely which of the primarily proposed articulatory features generated large errors; we proposed a new set of features in these cases.

After the training phase and weight saving, the system runs in the normal (testing) phase. By running forward through the neural structure, the phone 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 made by means of a *minimum-distance criterion*. Therefore, only the errors resulting from confusions between phones (i.e., a feature combination corresponding to an incorrect phone) will remain.

We evaluated the overall performance of the phonetic converter by testing both the training database and the testing database. We obtained a very high rate of correctly transcribed words: over 99.4% correct word transcription for the training set and over 98.3% for the testing set; generally, no more than one error per word (an incorrect phone) was obtained. A small exception dictionary recently developed can actually handle most of the common errors generated after the automatic phonetic transcription.

We appreciate that these results are highly valuable and prove the validity of our approach. However, we demonstrated that such results could be achieved only with the aid of complete knowledge of the acoustic and phonetic features of the language sounds and by designing an optimized neural system architecture.

5. Diphone Database Design and Generation

Speech synthesis systems based on concatenation of pre-recorded speech segments have grown in popularity in recent years, and they generally yield speech of good quality. However, perceptual differences between the synthetic and natural speech are still easy to distinguish due to frequent acoustic discontinuities at segment junctions and inappropriate prosodic contours (Wouters and Macon, 2001).

One of the major steps in constructing a concatenative text-to-speech system is the design of the acoustic unit database; a good quality of the database is a fundamental requirement for highly natural synthetic speech. The development of the acoustic inventory is usually a complex process and involves important decisions: the structure of the database (natural sentences or words in isolation), the size and type of the units stored (phones or polyphones), the extraction method (manual or automatic), etc. To the best of our knowledge, no systematic methods have been found to treat all these subjects.

In our opinion, the adopted strategies must take into consideration at least four fundamental criteria:

- the acoustic and phonetic particularities of the given language;
- if a large corpus of natural speech with accurate phonetic transcription is available or not;
- the concatenation algorithm used in the signal generation stage; and
- the desired quality of the synthetic speech and the final destination of the TTS system.

After a particular strategy has been chosen, the database generation generally consists of the following basic steps: determine the synthesis units, create and record the speech corpus, and extract the units from the corpus (Burileanu et al., 2000).

Considering the arguments posed in Section 3, we decided to use *diphones* as the basic acoustic units for our TTS system; we found them a good compromise in meeting the general requirements for segmental units in concatenative TTS systems for the Romanian language. Taking into account the 33 phones described in

the last sections (and also the '#' symbol—word boundary), there is a theoretical number of 1,156 diphones. Of course, not all possible diphones occur in a given language. By automatically searching for all possible diphones in the 7,000-word dictionary introduced in Section 4, a final number of 822 diphones has been assessed (thus far, transitions between words have not been examined).

We then designed a 1,200 isolated-word corpus containing at least one realization for each diphone of the set. The words were carefully chosen to be diphone-rich and phonetically balanced, to make sure that all main acoustic events as well as all known coarticulation issues were captured.

The speech corpus was recorded using a close-speaking microphone (*Andrea Electronics, Anti Noise PC Headset ANC-500*) in a room with low ambient noise, in a one-day session. Due to our specific synthesis algorithm (PSOLA-type), the speaker was asked to utter the words very intelligibly, with a constant pitch and as little intonation as possible and, moreover, maintaining a constant rhythm. After the main recording session, all utterances were reviewed, and the speaker was asked to repeat the words with abnormal pronunciation or quality problems (e.g., clipped signals, microphone proximity distortions). Speech signals were sampled at 16 kHz, PCM 16-bit and stored in the standard Microsoft© *WAVE* format.

The acquired speech corpus was processed to obtain a rough amplitude equalization (there were inherent variations in voice loudness or distance from the microphone throughout the entire session); further diphone normalization was performed within the synthesis algorithm.

It is well known that concatenative TTS systems generally are not able to simulate all contextual variations and natural coarticulations of phonemes as they appear in normal speech. Consequently, the design of the acoustic database for such a system must be accompanied by methods that attempt to reduce the undesirable effects of unit concatenation.

Traditional concatenative systems adopt a "standard" synthesis method: a single unit is used for each phonetic specification, and its pitch and duration are modified in the signal processing stage to meet the target specification. On the other hand, many recent TTS systems perform "dynamic unit selection" using a large pre-recorded speech corpus that covers widespread phonetic sequences in various prosodic contexts (Beutnagel et al., 1999; Taylor and Black, 1999).

Both methods have their advantages and drawbacks:

- the standard method has the advantage of allowing one to build and maintain the database quickly but usually results in relatively poorer output speech quality;
- the unit selection method uses a costly database and requires a very large corpus of natural speech to extract the units. The run-time choice of the best instance makes this method inherently slower, but a better speech quality usually results.

Taking into account these considerations, we adopted for our TTS system a flexible approach: a basic set of diphones is enlarged and several tokens for each diphone are stored; then, *spectral-distance measures* are computed and also stored. At synthesis-time, an optimal unit sequence that minimizes global acoustic discontinuity at unit boundaries is selected.

Diphone segmentation was performed manually using a dedicated signal editor showing both speech waveforms and spectrograms (as per the procedure described in Section 2). This is an expensive method but known to guarantee good quality.

A three-step procedure was used to create two distinct diphone databases:

1. Considering one sample of each diphone from the basic set, we first created a *reduced-size database* (RDB); it contains 822 units, each of them being carefully extracted from a different word in the recorded speech corpus (central position, if possible, and avoiding some difficult situations described in Section 3). As much as possible, the segmentation was performed on stable regions of sounds. This stored database served as the reference for the evaluation tests.
2. A *large-size database* was then built by extracting all possible diphones from each word in the speech corpus.
3. The large-size database was thereafter pruned with the aid of a program that automatically removed too many appearances of a diphone in exactly the same context. Consequently, an *extended database* (EDB) containing 3,970 units was obtained and stored.

Finally, the two acquired databases (the RDB and the EDB) were processed in order to mark the exact position of the boundary between phones in each segment.

There are several types of perceptual distortions that occur in concatenative TTS systems (Campbell and Black, 1997; Conkie and Isard, 1997). Due to the specificity of the PSOLA algorithm (duration and pitch controlled by the algorithm), we were only interested in the *spectral discontinuity* at the junction of the segments selected for concatenation.

Several definitions of spectral distance between two speech patterns are widely used in speech recognition. However, the problem is somehow different in speech synthesis, because it is not clear how these objective criteria match subjective perception of spectral dissimilarity between adjacent segments in the output speech.

For our experiments, we estimated the *continuity distortion* at diphone junctions by calculating the *truncated warped cepstral distance* between pairs of warped cepstral vectors (Rabiner and Juang, 1993). The examination was done on 10 ms frames and 12 *mel-scale cepstral coefficients* were derived from a short-time FFT analysis (the mel frequency scale was used because it represents a more psychological subjective measure of the spectrum than does the normal scale).

The distortion measure around a concatenation point of two diphones was calculated as follows:

$$d_C^2(L, P) = \sum_{j=1}^{P} \sum_{i=1}^{L} (c_{ij} - c'_{ij})^2, \qquad (1)$$

where c_{ij} and c'_{ij} represent the jth mel-scale cepstral coefficients (in our case, $L = 12$) of the ith frame to the left and right of the concatenation point, respectively.

The frame number, P, is calculated based on a minimum *segment length* S_k:

$$S_k = \min(s_k, s_{k+1}), \qquad (2)$$

where s_k and s_{k+1} represent the right part of the diphone k (ranging between the phones boundary and the right margin of the diphone) and the left part of the diphone $k + 1$ (ranging between the left margin of the diphone and its phones boundary), respectively. We used many frames instead of only one, because we wanted to capture the difference between the parameter values at segment junctions as well as between the (possible) parameter change rates on a number of frames.

The cepstral distances were calculated between all possible adjacent diphones and stored in two look-up tables, both for the RDB and the EDB.

At synthesis time, two different procedures are used in order to perform an *optimum unit selection*:

L–C. A *local dissimilarity criterion*: for each diphone k one automatically selects the most favorable diphone $k+1$ from all possible candidates, based on the minimum spectral-distance measure found in the look-up table. For the selected $k + 1$ diphone, the best $k + 2$ diphone is selected following the same criterion, and so on.

G–C. A *global dissimilarity criterion*: since there are several candidates for each junction, there are many possible combinations of these segments. We used a *dynamic time warping* (DTW) algorithm to find the best path between all possible combinations of diphones for a word and thus minimize the sum of all distortions across the whole word.

Six perceptual (formal listening) tests were conducted to evaluate the elected distortion criteria. Twelve subjects were selected for these experiments, all of them being unfamiliar with speech synthesis systems. Dedicated interactive software was used during the tests.

We carefully chose a set of 150 words with known pronunciation issues. The words were synthesized in six different ways:

Test 1: by simple diphone concatenation (and using an amplitude smoothing function at diphone boundaries to avoid usual "clicks") with units from the RDB;

Test 2: by simple diphone concatenation with units from the EDB and diphones selected using the procedure L–C (local dissimilarity criterion);

Test 3: by simple diphone concatenation with units from the EDB and diphones selected using the procedure G–C (global dissimilarity criterion);

Test 4: by diphone concatenation using the complete TTS system (through the PSOLA-type algorithm) with units from the RDB;

Test 5: by diphone concatenation using the complete TTS system with units from the EDB and diphones selected using the procedure L–C; and

Test 6: by diphone concatenation using the complete TTS system with units from the EDB and diphones selected using the procedure G–C.

For each test, the word set was presented in a random order. The subjects listened to each word of the set successively and were asked to pronounce what they heard (we therefore avoided expressions like "It seems to be ...").

The program computed the average recognition rates for the six tests. The results are presented in Table 4.

Table 4. Intelligibility scores for the six tests.

	Test 1	Test 2	Test 3	Test 4	Test 5	Test 6
Recognition rate (%)	84.7	89.2	90.7	91.3	94.9	95.8

The scores show an improvement in recognition rates for tests 2 and 5 compared to tests 1 and 4. However, there were a number of words correctly heard in tests 1 and 4 that were not recognized in tests 2 and 5. This means that (1) perceptual consistency is hard to obtain among human listeners, and (2) minimizing the spectral discontinuities do not necessarily lead to a better perceptual quality. The scores also show a slighter speech quality improvement using the global dissimilarity criterion (tests 3 and 6).

The final conclusion of these tests was that using an extended acoustic database and performing an optimum unit selection based on spectral distortion criteria clearly translates into better speech quality. Evaluation experiments showed that minimizing distortion at diphone junctions generally increased the naturalness of the output speech. However, we noticed exceptions to this rule for some particular words. It is obvious that other distance measures (more relevant for speech synthesis) need to be tested in order to obtain more objective discontinuity measures (Klabbers and Veldhuis, 2001).

Further improvement of quality (a better fluency of the synthesized speech) is also expected by increasing the size of the present database (the EDB) with cross-word diphones; this task is currently under way.

6. Conclusions

This paper described the present status of the TTS synthesis system developed in our laboratory, presenting basic achievements and also recent improvements made at certain processing levels. Several tasks of the natural language processing stage, implementation details and experimental studies were discussed. We emphasized many Romanian language particularities and pointed out the fact that high-quality synthesized speech can be obtained only if one takes into account to a deep extent the language-specific linguistic knowledge.

We would like to mention that important work is currently in progress to improve the system performance, especially the naturalness of the generated speech. For example, extensive linguistic and prosodic studies are now being conducted in order to replace the present (simple) manually-created prosodic rules. The part-of-speech analysis and the phrase-level parser will be significantly improved, and a new strategy for stress assignments is being investigated. Finally, we recently started to develop a different synthesis technique based on a harmonic-plus-noise model of speech.

The present version of the TTS system provides a very useful framework both for fundamental research and for further increasing system performance. Its modular approach and the design philosophy of the constituent modules permit easy modifications based on research advances or the demands of specific applications.

References

Ainsworth, W.A. and Pell, B. (1989). Connectionist architectures for a text-to-speech system. *Proceedings of Eurospeech'89*, Paris, France, pp. 125–128.

Beutnagel, M., Mohri, M., and Riley, M. (1999). Rapid unit selection from a large speech corpus for concatenative speech synthesis. *Proceedings of Eurospeech'99*, Budapest, Hungary, vol. 2, pp. 607–610.

Boula de Mareüil, P., Yvon, F., D'Alessandro, C., Aubergé, V., Bagein, M., Bailly, G., Béchet, F., Foukia, S., Goldman, J.-P., Keller, E., O'Shaughnessy, D., Pagel, V., Sannier, F., Véronis, J., and Zellner, B. (1998). Evaluation of grapheme-to-phoneme conversion for text-to-speech synthesis in French. *Computer Speech and Languages*, *12*(4):393–410.

Burileanu, D. (1999). Natural language processing for speech synthesis in Romanian language. *Proceedings of the 12th International Conference on Control Systems and Computer Science (CSCS12)*, Bucharest, Romania, vol. 2, pp. 1–6.

Burileanu, D., Sima, M., and Neagu, A. (1999a). A phonetic converter for speech synthesis in Romanian. *Proceedings of the XIVth Congress on Phonetic Sciences (ICPhS)*, San Francisco, CA, vol. 1, pp. 503–506.

Burileanu, D., Dan, C., Sima, M., and Burileanu, C. (1999b). A parser-based text preprocessor for Romanian language TTS synthesis. *Proceedings of Eurospeech'99*, Budapest, Hungary, vol. 5, pp. 2063–2066.

Burileanu, D., Burileanu, C., and Neagu, A. (2000). Diphone database development for a Romanian language TTS system. *Proceedings of the International Symposium "State-of-the-Art in Speech Synthesis"*, London, pp. 9/1–9/6.

Campbell, N. and Black, A.W. (1997). Prosody and the selection of source units for concatenative synthesis. In J.P.H. van Santen, R.W. Sproat, J.P. Olive, and J. Hirschberg (Eds.), *Progress in Speech Synthesis*. New York: Springer-Verlag, pp. 279–292.

Conkie, A.D. and Isard, S. (1997). Optimal coupling of diphones. In J.P.H. van Santen, R.W. Sproat, J.P. Olive, and J. Hirschberg (Eds.), *Progress in Speech Synthesis*. New York: Springer-Verlag, pp. 293–304.

Daelemans, W.M.P. and van den Bosh, A.P.J. (1997). Language-independent data-oriented grapheme-to-phoneme conversion. In J.P.H. van Santen, R.W. Sproat, J.P. Olive, and J. Hirschberg (Eds.),

Progress in Speech Synthesis. New York: Springer-Verlag, pp. 77–89.

D'Alessandro, C., Rizet, M.G., and Boula de Mareüil, P. (1996). Synthèse de la parole à partir du texte. In H. Méloni (Ed.), *Fondements et perspectives en traitement automatique de la parole*. Aupelf-Uref, pp. 81–96.

Dutoit, T. (1997). *An Introduction to Text-to-Speech Synthesis*. Dordrecht: Kluwer.

Ferri, G., Pierucci, P., and Sanzone, D. (1997). A complete linguistic analysis for an Italian text-to-speech system. In J.P.H. van Santen, R.W. Sproat, J.P. Olive, and J. Hirschberg (Eds.), *Progress in Speech Synthesis*. New York: Springer-Verlag, pp. 123–138.

Gubbins, P.R. and Kurtis, K.M. (1995). Neural network solutions for improving English text-to-speech transcription. *Proceedings of the International Conference on Phonetic Science*, Stockholm, Sweden, pp. 314–317.

Jiang, L., Hon, H.W., and Huang, X. (1997). Improvements on a trainable letter-to-sound converter. *Proceedings of Eurospeech'97*, Rhodes, Greece, pp. 605–608.

Karaali O., Corrigan, G., Gerson, I., and Massey, N. (1997). Text-to-speech conversion with neural networks: A recurrent TDNN approach. *Proceedings of Eurospeech'97*, Rhodes, Greece, pp. 561–564.

Klabbers, E. and Veldhuis, R. (2001). Reducing audible spectral discontinuities. *IEEE Transactions on Speech and Audio Processing*, *9*(1):39–51.

Liberman, M.Y. and Church, K.W. (1992). Text analysis and word pronunciation in text-to-speech systems. In S. Furui and M.M. Sondhi (Eds.), *Advances in Speech Signal Processing*. New York: Marcel Dekker, pp. 791–831.

Lindstrom, A. and Ljungqvist, M. (1994). Text processing within a speech synthesis system. *Proceedings of the ICSLP'94*, Yokohama, Japan, pp. 139–142.

Rabiner, L. and Juang, B.H. (1993). *Fundamentals of Speech Recognition*. New Jersey: Prentice-Hall.

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

Taylor, P. and Black, A.W. (1999). Speech synthesis by phonological structure matching. *Proceedings of Eurospeech'99*, Budapest, Hungary, vol. 2, pp. 623–626.

Wells, J., Barry, W., Grice, M., Fourcin, A., and Gibbon, D. (1992). *Standard Computer-Compatible Transcription*. Esprit project 2589 (SAM), Doc. no. SAM-UCL-037. London: Phonetics and Linguistics Dept., UCL.

Wouters, J. and Macon, M.W. (2001). Control of spectral dynamics in concatenative speech synthesis. *IEEE Transactions on Speech and Audio Processing*, *9*(1):30–38.