BULETINUL Universităţii Petrol – Gaze din Ploieşti	Vol. LX No. 3B/2008	27 - 34	Seria Tehnică

An Automated System for the Vocal Synthesis of Text Files in Romanian

Adrian Moise, Alexandra Dan

Universitatea Petrol – Gaze din Ploieşti, B-dul Bucureşti nr.39, Ploieşti
e-mail: amoise@upg-ploiesti.ro

Abstract

This article's aim is to present an automated system used in obtaining audio signal by converting information in text format– only in Romanian language, into speech. Such a system may have multiple uses, from facilitating the access of sight impaired persons to practically any software, to reading highly expressive texts, such as a theater plays, for any type of auditorium. In the first part, there will be exposed the main existent methods, with their strong and weak points. Afterwards, there will be presented a technique implemented by the authors, consisting in concatenating the prerecorded syllables, stored in a database. The final part contains the experimental results, along with issues that may appear in several cases, as well as eventual solutions, with referrals to the practical application, about to be developed. The article ends by mentioning the conclusions about the facts above, as well as a few suggestions of possible later research.

Key words: *speech synthesis, concatenation, automated system, syllables, Romanian language*

Introduction

Vocal synthesis represents the artificial reproduction of human speech. Text-to-speech systems are systems that can transform a certain text to speech. They can satisfy a very large range of necessities.

Mainly, these software programs are very useful to sight impaired persons, helping them to communicate easier with the computer, by providing a “speaking interface”: Internet browsing, using operating systems or any software.

Moreover, speech-impaired persons may use such systems in order to communicate by phone. Already, most of the Internet browsers and operating systems have either incorporated text-to-speech systems, or facilities for installing such extensions.

The more the quality of the speech resulted improves, the more the text-to-speech systems begin being useful in listening the e-mail messages or RSS feeds, event notifications, movie subtitles, dictations, teaching children how to read or learning foreign languages.

Finally, there has been highly suggested, although not yet popular among implementations, that speech synthesis and speech recognition should be incorporated in the same system, facilitating each other's functionality [1], just as humans learn to produce and perceive speech “simultaneously” [2].

A speech synthesizer can be implemented either software, or hardware. There are several methods used for obtaining speech. Among them, the most important are:

● Articulator synthesis – based mainly on understanding the dynamics of vocal tract and obtaining a direct connection between this and speech acoustics [3]. At the moment, articulator synthesis is the less developed among speech synthesis methods, but on long term it may become the most efficient, in terms of accuracy.

● Rule-based synthesis - or parametric acoustic synthesis - allows a very wide range of digital audio signal processing, which leads to a very high expressiveness, provided its model, phoneme representation as well as co articulation rules are according to reality.
In this case, no prerecorded fragments of human speech are used. Therefore, the obtained speech sounds quite artificial, robotic-like. Nevertheless, words are intelligible, even at high speeds of speech reproduction. Moreover, these types of systems do not use a lot of computers' resources – as they do not a database with prerecorded speech, which would have occupied a considerable memory space, and also due to the simple algorithms, they don't need very fast CPUs either. This is the reason why very often such systems are used by visual impaired persons, in order to interact better with the computer.

● Concatenated waveform synthesis – relies on concatenating different fragments of prerecorded speech, which can vary from whole phrases, to simple phonemes, and may include diphones, triphones, morphemes, words, sentences.

Generally, for optimization, it is created an index of records on fundamental frequency, length, or contained phonemes. At runtime, the optimum chain of records is determined, based on criteria, using for example weighted decision trees.

Due to the low-level signal processing, the speech sounds very natural, especially in those cases when the text-to-speech system was adapted to a certain vocabulary. Nevertheless, the bigger the database is, the better results are obtained. In addition to this, the selection algorithms do not always “choose” the best option, certain words ending up to be “pronounced” rather unclear.

The highest quality is obtained when dealing with domain specific synthesis, when the vocabulary used is relatively reduced, and all the words and phrases that are going to be used are prerecorded. The disadvantage consists in the fact that such systems do not have a general purpose.

On the other side, working in almost any situation, but producing a bad quality speech, are the text-to-speech systems which use diphone concatenation. Diphones are the signals from either the mid point of a phone or the point of least change within the phoneme to the similar point in the next phoneme [3]. Lately, such implementations have rarely been done in commercial applications, although they are still being used in research, due to the multitude of available open source software.

Application Description

For implementing the practical application, there has been chosen the concatenated waveform synthesis method. To be more precise, it consists in concatenating syllables. By using no digital signal processing techniques (except maybe for a digital coding method when recording the sound, and eventually, another one for decompressing the data before playing it [6]), the resulted speech will sound very natural, provided the database is large enough to include all the necessary syllables. Thereby, without these syllables, shorter groups of sounds would be concatenated, which would result in pronouncing quite unclear some words. Nevertheless, in case of using such an application when a relatively limited vocabulary is necessary, such as for example reading weather forecasts, the results are satisfactory.

This concatenation method seems to be an optimum choice for Romanian language. According to researches made on Romanian syllables and words, the 50 most frequent syllables (with represent 0.07% of the total number) occur in 50.03% of the syllables obtained by splitting all the words from the dictionary. Furthermore, a number of 1200 syllables represent 95% of the total. (Fig. 1)

All in all, by recording the most frequent 400 syllables, it is very easy to obtain a large-enough database for reproducing the most common Romanian words.

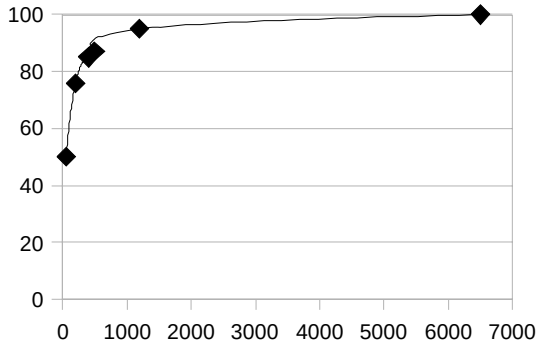

Fig.1. Dependence between the number of most-used syllables and their occurrence percent in all of the Romanian words

It is aimed to implement some classes, with an interface that can allow calling a function such as

speak(text,[options])

where *text* is a string parameter, and *options* is an optional parameter. Thereby, any other application can have access to this function, allowing a more efficient communication with the user. For example, this can be the case of a software program that connects to an e-mail service and "reads" to the user the current messages, with the help of such a function.

The process of obtaining speech, starting from a given text is depicted in Figure 2, and includes several stages, being relied on using some additional resources.
In the initial phase, the text must be processed, with the purpose of bringing it to a compatible format to the corresponding functions of the next phases. Therefore, it is recommended that at the end of this processing stage, the text will only contain Romanian characters; the words will be separated by spaces, and without capitals. The punctuation marks must be properly represented, meaning that there must be eliminated all type or errors, such as *"...."* instead of *"..."*. Moreover, this stage includes eliminating the digits – by replacing them with their corresponding Romanian numerals. There are many possibilities of making these changes, this is why it is recommended for the user to choose one of the options: for example *"12"* may be *"doisprezece"*(engl. *"twelve"*) or *"unu doi"* (engl. *"one two")*.

Splitting a word into syllables is performed by applying the rules from Romanian, on the above text. There are seven rules based on pronunciation, and two rules based on morphologic analysis [5]. The exceptions from these rules are also implemented. Before being able to apply these rules and exceptions, it is firstly necessary to establish the type of each sound: consonant, vocal, or semivowel. Finally, it is obtained a text where words are split in syllables, separated by the character "-", text that may be represented using an ordered list containing the syllables and other punctuation symbols, such as "." - dot character, for example.

Most of the times, the database containing the prerecorded sounds is not wide enough to cover all the possible cases, this is because after splitting the words may result syllables that haven't

been recorded yet. In such cases, it is used a fragmentation algorithm, that splits them in smaller groups of sounds, until the program has all the necessary data to select from the database and is able to proceed in concatenating all the syllables in a **.wav* file. Here also interfere some user options, which may decide on the duration of the pause between words. Finally, the obtained file is played, obtaining speech.

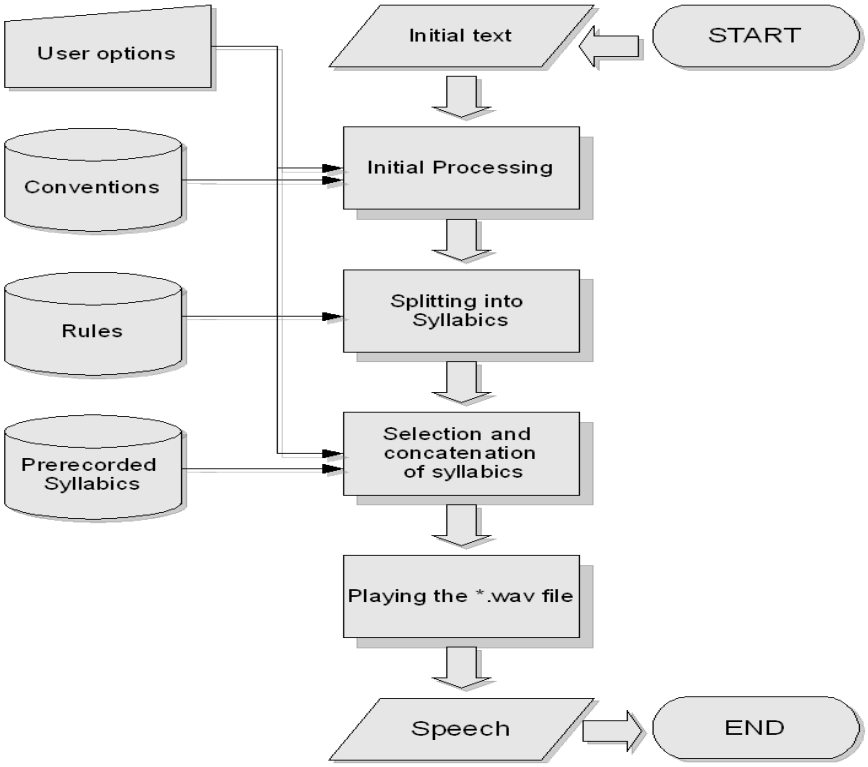

Fig.2. Application flowchart

Experimental results

It is rather difficult to establish the meaning of some character groups, depending on the context. For example, a number having this format *"123.234.190.087"* may represent an amount of money, or may be as well an IP address, in which case replacing the number with the equivalent from Romanian of *"one hundred twenty three thousand millions two hundred thirty four etc.."* would be meaningless. A solution in this case would be a text analysis and checking whether the given numeral is accompanied by a word that can identify a value, a price, a currency, or, on the contrary, to prove that it is an IP address or a phone number.

In the case of some common abbreviations in Romanian, such as "*s.a.m.d*", "*a.c.*", "*etc*", "*km*", these may be replaced with their corresponding complete words/groups of words.

As an observation, the hyphens that connect two words, must be eliminated: *"intr-un"* becomes "*intrun*", instead of *"in-tr-un"*, which wouldn't have been correct as pronunciation – because there is no syllable such as *"tr"*, but orthographically, would have been perfectly correct.

It is important mentioning that for this application, the purpose of splitting into syllables is not to obtain the Romanian syllables spelled correctly, but to obtain groups of letters as long as possible, easy to pronounce, for being recorded in the database, and as clear as possible, in order to obtain a very intelligible final output.

Furthermore, another problem encountered was that applying the rules and the exceptions does not always lead to favorable results. The algorithm works for most of the Romanian words, as they appear in the dictionary, but, by applying these rules to some alternative forms of the same word (i.e.: declination, case, number), errors might be obtained. For example, the word *"sera"* would be correctly split as "*sera*", whereas for *"seri",* it is obtained *"se-ri",* which is wrong. The solution is simple, and it consists in temporarily eliminating the final "*i*"(that marks the plural of the noun *"sera")* , splitting into syllables the so-called word "*ser*", and finally adding at the end of the last syllables the initially eliminated *"i".* Nevertheless, this solution must me applied only for those words, nouns or adjectives, that when at plural forms, finish in "*-i*". Otherwise, splitting any other word finishing in *"-i"* but not respecting the other requirements would result in bad results.

Another problem is represented by the diphthongs and triphtongs, consisting in a vocal and a semivocal, respectively in a vocal and two semivocals. The groups of characters that may form diphthongs or triphtongs in Romanian are well-known, but there are cases when the same group of characters can be either a diphtong or not, and there are no rules that can establish this.

For example, the group *"io"* is a diphthong. By applying the rule of splitting into syllables for diphthongs, the word *"spaniol"* can be split as *"spa-niol"* or as "*spa-ni-ol*", in case of not considering it a diphthong. Moreover, in Romanian there are twentytwo diphthongs and eleven triphtongs, which can be at the beginning, the middle or the end of the words, combined with different other sounds. This makes it a very wide set of possible situations and stating rules concerning this cases is very difficult.

Due to the fact that there is no efficient method of making distinction between all these situations, and considering the mention above, choosing the variant where word fragmentation is minimum, and therefore, the speech resulted is the most clear, seems to be the best solution.
Furthermore, the next problem would be whether the obtained syllables are already recorder in the database, or not. If not, an alternative must be found.Let *M* be the set of the prerecorded syllables in the database. Let *S* be a syllable in the text that must be transformed into speech. If $S \notin M$, then there must be a method of discomposing *S* in shorter groups of characters S_i, $S_i \in M$. Let *L* be a list of integer numbers x_i, with possible values from *1* to *n-1*, satisfying the following property:

$$\sum_{i=1}^{n} x_i = \sum_{i=1}^{n} length(S_i) = length(S) = n \tag{1}$$

These integers may be generated by using a certain method, for example backtracking.

Supposing that *xi* represents *length(S_i)- the number of characters of Si.* Then we may say that *L* will be containing the numbers representing the lengths of the groups in which the fragmentation of *S* has been done. In addition to this, if *L* is an ordered list, it it will be containing the groups' lengths in the same order as they appear in the syllables. If *L={2,3},* this implies that the first group contains two characters, and the second three, where obviously, the whole syllable will be five characters long. Thus, it can be established one possible partition of the syllable that must be "pronounced". The validity condition is that each group of characters that has been obtained to be previously recorded in the database. Otherwise, the *"L"* list has absolutely no utility, and must be found other solution.

All these possible sets, noted by L_j, can be generated. Let $T'=\{L_j, j=\overline{1,m'}\}$-where m' is the number of possible generated sets L_j. From these, there can be eliminated all those variants that imply using some groups of characters that do not exist in the database. We will obtain $T=\{L_j, j=\overline{1,m}\}$, with $1 \le m \le m'$ and $T \subset T'$. Note that T may never be an empty set. In the worst case, S will be split in groups of length one each, that means $|L_j|=length(S)=n$, and $T'=\{T\}$. Nevertheless, there are cases when $|T| \le 2$, therefore, there are multiple ways of decomposing S, from which the optimum one must be chosen: L_{final}.

Table 1. Syllables' length

Nr. crt.	Length	% from the total	No. of possible variants
1	1	9%	1
2	2	58%	2
3	3	27%	4
4	4	6%	8

Statistically, the syllables' lengths and their incidence are presented in Table 1 [4].
By taking into consideration that the maximum length of a syllable in Romanian is 6 – extremely rare, but possible, we can assume that in the worst case, due to a bad splitting, *length(S)*= 6·2=12. By mathematical induction, it can by easily prove that for an *n*-letter *S*, there will be obtained 2^{n-1} possible splittings. So, in the worst case, for *length(S)*=12, we will obtain 2^{11} = 2048 variants. However, for an average-length *S*, we will have 2^{4-1} = 8 variants to choose from.

The criteria used in choosing L_{final} is represented by the lengths of the characters groups, that must be as large as possible, in order to obtain at the end a quality speech, that does not sound "fragmented".

In conclusion, the number of such groups must be minimum:

$$L_{final} = \{L_j \mid |L_j| = \min\{|L_k|, k=\overline{1,n}\}\} \quad (2)$$

If there are still multiple sets L_j with this property, there will be eliminated firstly those sets containing group x_i consisting only of consonants. The reason is because it is very hard to concatenate such groups of characters, without vocals. As a proof, in Romanian, all syllables must contain at least one vocal [5]. Finally, it is chosen any of the remaining possibilities.

Another problem when using a concatenated waveform synthesis is the database size, which increases dramatically when dealing with a wide vocabulary. It is always possible to compress the data, but this implies compromising the sound's quality.

Moreover, when it comes to prosody and expressive effects, the method of sound wave concatenation has a further inconvenient. It seems that there is no linear correlation between intonation, stressing, prominence, or rhythm on one hand, and signal properties on the other

hand. Although nowadays digital signal processing techniques are very advanced, they can not be applied to sounds for such purposes, say, in order to change their intonation, due to the lack of a mathematical model, which could have made the link between the prosody and the sound properties [3].

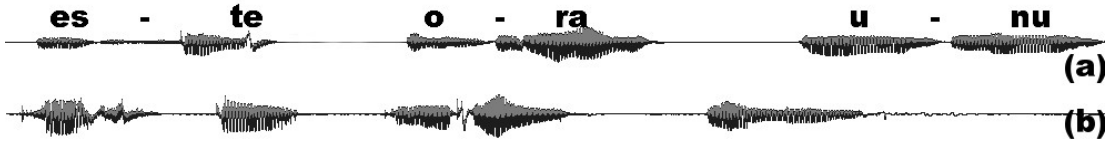

Fig. 3. Comparison between (a) concatenated speech and (b) natural speech, both pronounced by the same voice.

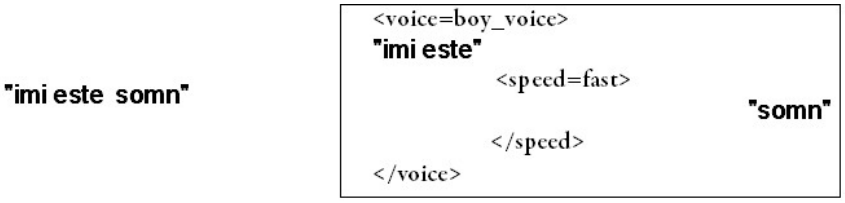

Fig. 4. Example of using a markup language

In Figure 3, there are two possible waveform representations of the same sentence in Romanian: "*este ora unu*". The first one was obtained using the application described in the previous section, by concatenating six syllables, and inserting between the words a "silent" wave form, of constant duration: "*es-te-'silence'-o-ra-'silence'-u-nu*". All the syllables have been previously recorded, being pronounced in a neutral way. The second one was pronounced by the same voice, in the same conditions, but this time all the sentence at once. It is obvious that the waveforms not only look different, but also sound different when played, because of the lack of expressiveness of the first one.

One solution could consist in using multiple databases, containing samples of exactly the same voice and same units, but with different prosody. As a consequence, more disk space will be used. An additional condition is to have all the sounds recorded in the same conditions, with no background noise, using the same microphone and exactly the same human voice, with no variations whatsoever. Still, one more fact must be taken into consideration: there are many possible combinations of prosodic variants. Sometimes it is difficult for the algorithm to establish which one to use in a given situation; this is why markup languages can prove to be very useful in speech synthesis (Fig. 4). Such an approach involves using tags that indicate the database from which the current unit should be selected. Next, an interpreter must detect and analyze them, in order to retrieve the speech unit from the right database.

There may be used different databases for several types of voices, depending on the dialect, expressiveness, intonation, speaker's gender, style, even speech speed. Of course, this is not a very practical solution, mostly because the storage space is limited, but also because of the technical difficulties encountered when working with a large number of waveforms: all recording must be done in the same conditions, the background noise must be eliminated, the units need to be separated [7], and afterwards, because of their large number, at run-time, searching them might take a longer time than accepted.

Conclusions

So far, there have been presented the main speech synthesis methods. The method chosen for implementation consists in concatenated waveform synthesis. The main disadvantage is represented by its lack of flexibility when it comes to expressiveness. Still, using multiple databases could be a solution. Markup languages help implementing this solution.

Nevertheless, in case of not using multiple databases, such an application could still prove to be useful in simple contexts, where there is no need for intonation, or even where no type of expressiveness is recommended at all – say, for example, "reading" the button captions of a software's interface when the mouse is passed over a window control. The obvious advantage is represented by the naturalness of speech. Moreover, as previously shown, when dealing with a limited vocabulary, such a method is most recommended, taking into consideration that we already know which syllables to prerecord. All in all, we consider that a better concatenated waveform synthesis relies either on finding a useful mathematical model, that can show the dependence between wave sound and phonetic properties, or on using a markup language together with multiple databases, in order to specify different text particularities.

References

1. Dutoit, T and all. - *Synthèse Vocale et Reconnaisance de la Parole: Droites Gauches et Mondes Parallèles*, Faculté Polytechnique de Mons, 2002
2. Fallside, F.- *Spoken language acquisition by computer*, The Journal of the Acoustical Society of America, Cambridge Univ, 1992
3. Tatham, M., Morton, K.- *Developments in Speech Synthesis*, John Wiley & Sons, 2005
4. Dinu, L., *Despartirea Automata in Silabe a Cuvintelor din Limba Romana. Aplicatii in Constructia Bazei de Date a Silabelor Limbii Romane – Raport de Cercetare*, Universitatea Bucuresti, 2004
5. Arhanghelovici, C. - *Indreptar ortografic, ortoepic si morfologic al limbii romane*, Saeculum, 2007
6. Rabinier, L. R., Schafer, R. W.- *Difital Processing of Speech Signals*, Prentice Hall, 1978
7. Teodorescu, H. N.- *http://www.etc.tuiasi.ro/sibm/romanian_spoken_language/index.htm, Site-ul 'Limba Romana Vorbita'*, 2006

Sistem automat pentru sinteza vocală a fişierelor text în limba română

Rezumat

Scopul acestui articol este prezentarea unui sistem automat folosit pentru obţinerea semnalelor audio prin conversia informaţiei din format text, în Limba Română. Un astfel de sistem are utilizări multiple, de la facilitarea accesului persoanelor cu disabilităţi de vedere la orice produs software, până la citirea foarte expresivă a textelor, cum ar fi în piesele de teatru. În prima parte sunt expuse principalele metode de sinteză vocală existente, accentuând avantajele şi dezavantajele fiecăreia. Apoi, se prezintă metoda implementată de autori; aceasta constă din concatenarea silabelor preînregistrate, memorate într-o bază de date. În partea finală sunt prezentate rezultate experimentale împreună cu problemele care pot să apară şi cu soluţiile propuse. Articolul se încheie cu concluzii şi sugestii pentru cercetări ulterioare.