HYBRID SYLLABIFICATION AND LETTER-TO-PHONE CONVERSION FOR TTS SYNTHESIS

Cătălin UNGUREAN[1], Dragoş BURILEANU[2], Vladimir POPESCU[3], Aurelian DERVIŞ[4]

Într-un sistem de sinteză a vorbirii pornind de la text (TTS), prezența etajului de prelucrare a limbajului natural este esențială dacă se doreşte obținerea unei vorbiri sintetizate cât mai naturală, pornind de la un text oarecare de intrare. În lucrarea de față abordăm două probleme importante pentru un sistem TTS în limba română, din punct de vedere al prelucrării limbajului: despărțirea automată în silabe a cuvintelor, etapă obligatorie pentru poziționarea corectă a accentelor lexicale şi în final generarea unei prozodii corespunzătoare, şi transcrierea fonetică a textului de intrare. Primul algoritm este construit într-o manieră hibridă, pe baza unui set minimal de reguli generale, urmat de o abordare statistică, iar cel de-al doilea este construit în esență pe baza unui set de reguli de conversie grafeme – alofone, pe baza formei despărțite în silabe a cuvintelor. În plus, vom arăta că predicția accentului lexical poate fi folosită pentru a îmbunătăți rezoluția transcrierii fonetice, prin eliminarea unor ambiguități de conversie.

The presence of the natural language processing (NLP) stage in a text-to-speech (TTS) synthesis system is an essential condition for obtaining a good naturalness of the synthesized speech in a given language, starting from unrestricted input text. In this paper we address two important NLP issues for a Romanian TTS system: automatic syllabification, necessary for lexical stress assignment and prosody generation, and letter-to-phone (L2P) conversion of the input text. The first algorithm is built on a hybrid strategy, using a minimal set of general rules, followed by a statistical (data driven) approach, while the second one uses a set of phonetic transcription rules that work aligned with the correctly syllabified words. Moreover, we demonstrate that lexical stress prediction can help the L2P process, by solving some additional ambiguities.

Keywords: Text-to-speech synthesis, syllabification, letter-to-phone conversion, lexical stress, *n*-grams

[1] PhD student, Faculty of Electronics, Telecommunications and IT, University POLITEHNICA of Bucharest, Romania, e-mail: catalin20_09@yahoo.com

[2] Prof., Faculty of Electronics, Telecommunications and IT, University POLITEHNICA of Bucharest, and Romanian Academy Center for Artificial Intelligence, Bucharest, Romania, e-mail: bdragos@messnet.pub.ro

[3] Assistant Prof., Faculty of Electronics, Telecommunications and IT, University POLITEHNICA of Bucharest, Romania

[4] PhD student, Faculty of Electronics, Telecommunications and IT, University POLITEHNICA of Bucharest, Romania

1. Introduction

A high-quality text-to-speech (TTS) synthesis system based on a concatenation approach usually includes a natural language processing (NLP) stage that provides the phonetic transcription of the input text together with the appropriate linguistic information for prosody generation and speech unit selection, along with a signal processing stage that transforms the information received into speech [1], [2].

Our research team works for several years in the text-to-speech synthesis field. Several versions of a Romanian language TTS system were built successively in order to improve the performance of different constituent modules and consequently enhance the quality of the synthesized speech. The current version is based on acoustic segment concatenation and uses multiple instances of non-uniform speech units (diphones and polyphones – to solve a number of difficult vowel-semivowel transitions), labeled (off-line) according to contextual and phonetico-prosodic information from a recorded speech corpus [3]. The NLP stage of our TTS system provides the phonetic transcription of the input text, together with prosodic marks and auxiliary contextual and phonetic information. In [4] and [5] we described two important modules of this stage, namely the automatic diacritic restoration (modified in the most recent version, by adding a morphological analysis stage), and respectively the lexical stress assignment.

The main purpose of this paper is to discuss two other important modules of the NLP stage. More specifically, we propose an automatic syllabification algorithm for the Romanian language, and a letter-to-phone (L2P) conversion one.

The first algorithm is essentially based on a hybrid strategy that uses *n*-gram similarity measures and a limited rule-set. The statistical approach used here is similar, up to a point, to that used for the lexical stress resolution algorithm, but some new aspects exist, though. First of all, the training corpus used for the syllabification is built in a different manner, by means of a manually correctly-syllabified text corpus. Secondly, even though our approach is based on the same *Katz backoff* algorithm, we purposefully chose not to use *n*-grams *smoothing techniques* under the level of trigrams. A final classic decision is made, though, by calculating the emission probabilities and choosing the alternative which maximize it.

The L2P proposed algorithm is mainly build in a ruled-based manner, however using information from the syllabification and stress assignment modules. It is interesting to observe that in a classical NLP approach, one usually performs the L2P conversion first, followed by the syllabification task. This usually happens because each syllable is centered on a vowel and, for accurate rule-based syllable identification, vocal phonetic resolution must come first. However, in our work we considered the reversed approach, that is, after

identifying the borders of the syllables by also using lexical stress information for a number of difficult situations, the phonetic transcription task is solved in a deterministic manner, using a (minimal) set of L2P conversion rules.

There are a limited number of achievements in the field of syllabification and L2P conversion for the Romanian language. For example, in [6] the authors describe a rule-based phonetic conversion system for a reported 4.79% Word Error Rate (WER). The rule-set has been built from a phonetically-transcribed dictionary by successively running the algorithm, counting error conversions, and extracting of new rules. After 11 such iterations, the WER decreased from 21% to 4.94%, at which point the algorithm seems to saturate. The reported results have been obtained on a dictionary composed of 4,779 frequent Romanian words, extracted from a corpus of literary texts and scientific works. Moreover, using this L2P module and a supplementary rule-set (24 rules), the authors implemented a syllabification module, with a 10.55% WER reported result.

Another syllabification algorithm is described in [7], and works on a rule based principle as well. First of all, there is a general set of three rules. A more extended rule-set (exception set), composed of 100 rules, is finally applied and the authors report 15% WER results for syllable detection.

A final example of L2P conversion algorithm is described in [8]. The method relies on two distinct parts. First, the method starts with a L2P rule set built with the help of linguists. The second component tries to overcome the inconsistency of this set; to this end, the authors use a freely available software tool (WEKA) to obtain an extensive rule-set. The software is well documented over the Internet and is well known throughout the Academic world. The reported algorithm accuracy is between 78% and 94.8% for different test sets.

The paper is structured as follows: Sections 2 and 3 discuss the principles of the two proposed algorithms, Section 4 describes the experiments performed and the main results obtained, and Section 5 concludes the paper with final remarks.

2. Syllabification algorithm description

In the Romanian language, any syllable must contain one single vowel. There could also co-exist other typical vocalic letters, but these are considered to be, at the phonetic level, semivowels, and occur together in diphthongs or even triphthongs which must be spotted for a syllabification attempt. The Romanian language benefits from a good language processing framework, which is a quite up-to-the-minute achievement of the Romanian Academy [9], being generally recognized as the starting point for any linguistic effort. Other interesting ideas, as those presented in [10] may also support the scientific effort, by offering some additional points of view. Despite these large linguistics exertions, it must be

established that, for a certain written word, it is not possible to apply all the syllabification rules acknowledged by linguists (e.g., in [9]), because in the written form there is no difference between vowels, semivowels or hiatus. In other words, the hiatus cannot be directly emphasized, nor the diphthongs or triphthongs. It is necessary to perform an analysis in order to recognize different kinds of vowels, i.e. in diphthongs / triphthongs, or hiatus. In our framework this is achieved through a statistical training on a correctly syllabified corpus and is implemented in the syllabification module.

For the task of Romanian words syllabification, we extracted from [9] a minimal set of general syllabification rules, denoted by **G**, which covers only a small part of all possible situations**.** The **G** set has seven rules; we will illustrate only the first three of them:

- Two identical adjacent vowel letters are placed in different syllables, except for the **ii** pair which is placed at the end of the word.
- The letter sequence **vowel-consonant-vowel** (VCV) can be separated between the first vowel and the consonant, except for the sequence including a final **i.**
- The letter sequence **vowel-consonant-consonant-vowel** (VCCV) can be separated between the two consonants, except when the first consonant is one of the following letters: **b**, **c**, **d**, **f**, **g**, **h**, **p**, **t** and the second one is **l** or **r** – in such cases the syllables separation is made between the first vowel and the consonant.

It is important to note that the **G** rule-set can not cover all the particular situations that can arise in the Romanian language. Nevertheless, this proved to be useful because it can solve some unclear situations while also lessening the overall algorithm complexity. Consequently, every word entering the algorithm is first passed through the general syllabification rule-set, for solving a limited but clear number of situations; another advantage of this approach is that the number of possible syllabifications variants, generated throughout the algorithm and amongst which the resolution must be done by computing the emission probabilities, is significantly reduced. Then, one uses a statistical approach based on *n*-gram similarity measures. The complete training and testing procedures are presented next.

2.1. The training procedure

1. From each word **w** of a manually-built training corpus, T1 (composed of correctly-syllabified words, with start and stop markers for every input attached to the words), extract the *n*-grams of length 1,…,*n*, at character level; the word is written as <S1-S2-…-Sn> where by S_i we

denote the inner word syllables. The syllable delimiter " - ", as well as the start " < " and stop " > " word markers are considered to be internal characters.

2. For an *n*-length history, also extract *n*-grams of order *n*-1, *n*-2,...,1.
3. Count the frequencies of all *n*-grams.
4. Calculate and store the resulting *maximum likelihood* (ML) probabilities, for 1,…,*n*.
5. Calculate the probabilities according to the *Katz backoff* method [5], [11] down to the level of trigrams. Discard previous data and store the results.

2.2. The testing procedure

1. Each test word **w** is passed through the **G** syllabification rule-set. Start and stop markers are placed at the beginning of the resulted pre-syllabified word, and the $\underline{\mathbf{w}}$ form is obtained:
$$\underline{\mathbf{w}}=l_1 l_2 \cdots l_i \cdots l_N \text{, where } l_1 = < \text{ and } l_N = > .$$
2. From every $\underline{\mathbf{w}}$ generate all possible syllabification variants $\mathbf{g_j}$, $j = 1,...,2^L$, by inserting the syllable marker " - " after each character from $\underline{\mathbf{w}}$, where by L we denote the $\underline{\mathbf{w}}$ length, given in number of characters.
3. For every $\mathbf{g_j}$ extract the *n*-grams with history 1,…,*n*.
4. Calculate the emission probabilities with the *Katz backoff* formula, where the conditional probabilities are those obtained during the training stage.
5. Choose $\mathbf{g_{j,max}}$ which yields a maximal word probability, and return it at the output as the correctly syllabified version for **w**.

The correct variant is chosen from the forms which successfully passed through all the levels of the algorithm so that it yields the maximal emission probability. This is calculated as a product of the conditional probabilities of the *n*-grams extracted from a test word, according to *Katz* algorithm and presented in [5] and [11]. The modification used by our method consists in not using the smoothing technique for the unseen *n*-grams, under the level of trigrams. This approach relies on a good coverage for the *n*-grams training corpus and considers the impossible *n*-grams, under the 3rd level, as carrying no probability. The drawback is that for longer histories the effect of data sparseness becomes more visible and impacts the final statistical results.

The training and testing phases of the syllabification algorithm are depicted in Fig. 1.

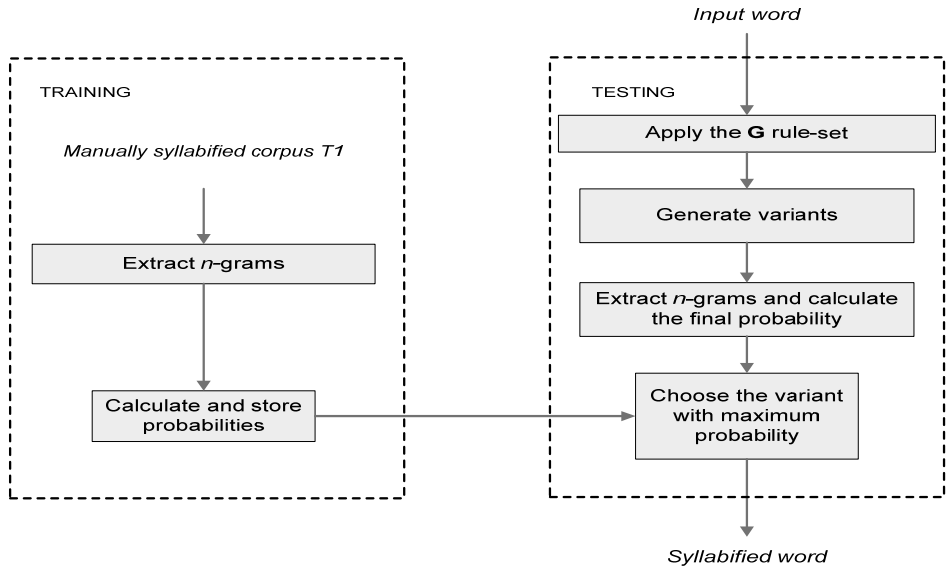

Fig. 1. The syllabification algorithm

Example: the word **sora** (*sister*), without passing through the **G** rule-set, will generate all the 64 possible syllabified variants (the " - " marker is inserted after each character):

g_1= <sora>, g_2 =<-sora>, g_3 = <s-ora>, g_4= <s-o-ra>,…,g_{64}=<s-o-r-a->

For this example, the word passes through the algorithm according to the graph depicted in Fig. 2.

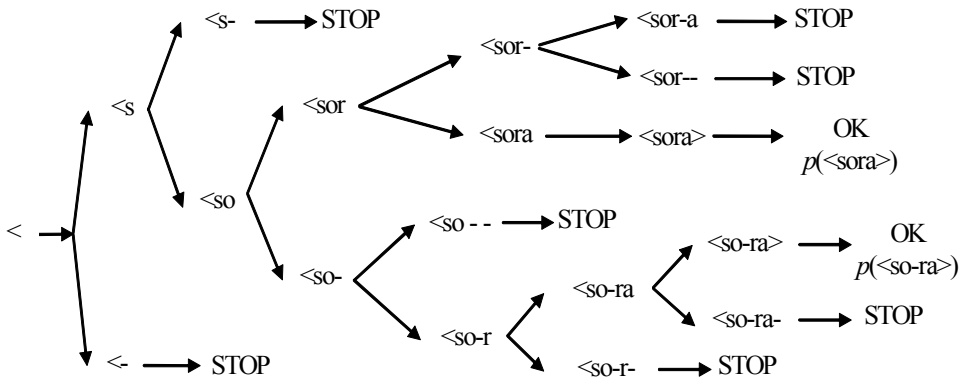

Fig. 2. The principle of the syllabification algorithm for the word **sora** (*sister*)

It can be noticed that, without passing through the **G** rule-set, the number of all possible variants grows exponentially with *L*, where *L* is the character length of a test word, taking into account the start and stop delimiters.

We observed that for the words with a length greater than 10, the method becomes time-consuming. Therefore, we decided to address this problem by stopping the algorithm for the unseen *n*-grams. Thus, our process will be stopped in a controllable manner, by ignoring theoretically-impossible variants.

In the previous example, the process stopped after only 9 branches, which is significantly smaller that the 64 theoretically-possible number. Also, one can observe that there are only two variants which will first pass through the algorithm, which are **<sora>** and **<so-ra>**. The algorithm will finally decide that the variant **<so-ra>**, that returned the maximal probability, is the correct form and will provide it at the output.

We point out that the preceding example was chosen only for theoretical reasons, disregarding (for the ease of discussion) the effects of applying the **G** rule-set. As we already mentioned, passing through the **G** set first, will substantially reduce the number of choices among which a decision is made.

3. L2P algorithm description

The following main difficulties are specific for the Romanian letter-to-phone conversion task: the presence of diphthongs, triphtongs, and hiatus, and the graphemes **ce**, **ci**, **che**, **chi**, **ge**, **gi**, **ghe**, **ghi** which have multiple phonetic values. For our study we used a basic set of 34 phones which is composed of 29 basic phonemes and five allophones for the vowels **e**, **i**, **o**, and **u**: four semivowels for each vowel of the mentioned group and the **final (unvoiced)-i** which carries no lexical stress and always follows a consonant (as in **comori** – *treasures*, **poți** – *you can*). The semivowels appear inside the syllables near vowels, resulting in phonetic diphthongs or triphtongs.

Even though a one-to-one correspondence between letters and phones does not exist, obviously, the correspondence is bi-univocal for consonants and for the vowels **a**, **ă**, **â**, **î**. As previously noticed, the diphthongs, triphthongs, and hiatus, which may result from the juxtaposing of several letters from the set **a**, **e**, **i**, **o**, **u**, **ă**, **î(â)**, are extremely difficult to differentiate in an automatic manner.

However, we observed that a relatively regular structure still exists. For example, we distinguished that from all the 49 possible pairs associations, the following distinct situations could occur: impossible type associations (14), hiatus type association (13), vocalic pairs that can form both hiatus and ascendant diphthongs (7), vocalic pairs which can form both hiatus and descendant diphthongs (14), vocalic pairs which can form hiatus and both ascendant or descendant diphthongs (1) – as for the **iu** vocalic pair.

It is further evident that by using an accurate syllabification algorithm that can highlight the correct structure of the vocalic sequences (diphthongs, triphtongs, and hiatus), a subsequent L2P conversion can be much more easily done, since the number of the remaining ambiguous pairs is limited and straightforward to distinguish.

The same study was made for triphthongs. We distinguished 16 different triple vocalic association types: 14 semivowel-vowel-semivowel (SVS) types and 2 semivowel-semivowel-vowel (SSV) types. We must re-emphasize that al these distinct cases are transformed in different L2P conversion rules, at the syllable level. Also, it must be mentioned that the triphthong generated rules must be applied before those obtained for diphthongs, but after hiatus resolution accomplished by means of the syllabification algorithm.

Our L2P converter, depicted in Fig. 3, uses syllables in order to solve most of the ambiguities which could occur between hiatus and different diphthongs or even triphthongs. Furthermore, we mention that the syllable information (more precisely the vowel / semivowel distinction in diphthongs), also enabled us to successfully address the correct phonetic transcription for the letter groups **ce/ci**, **che/chi**, **ge/gi**, **ghe/ghi**.

The remaining possibly unclear situations are solved by using the lexical stress assignment algorithm described in [5], e.g. the diphthong **iu**, which can be both ascendant or descendant; namely, the lexical stress placed on one of the two letters will determine that letter to be treated as a vowel, otherwise as a semivowel. In addition, lexical stress information proved to be a good approach for the **final-i** ambiguity resolution.

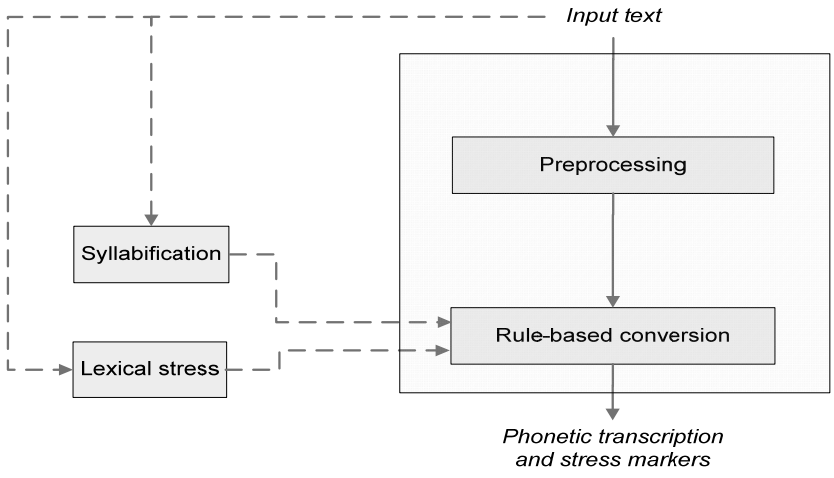

Fig. 3. The L2P algorithm

Then, the phonetic transcription can be considered deterministic – we used a set of 45 rules to accomplish the entire letter-to-phone conversion task. Moreover, we want to mention that our L2P module will also pass (to the NLP stage output) the lexical stress markers placed in front of the correspondent phonetically converted syllables. This step is essential for the next signal processing stage.

The proposed L2P algorithm is presented in the following:

1. For each test word do the following steps:
    a. Preprocessing: transform **y** to **i**, **ke** to **che**, **ki** to **chi**, **k** to **c**, **x** to **cs**, and **w** to **v**.
    b. Obtain the syllabified form of the word from the syllabification module.
    c. Obtain the stress information for the word, from the lexical stress assignment module and mark the stressed vowel on the correctly syllabified word.
2. For each syllable in the word identify the type of the inner diphthongs and triphthongs, and based on this vowel / semivowel information, perform the letter-to-phone conversion according to the set of 45 rules.
3. Replace the graphemes **ce/ci**, **che/chi**, **ge/gi**, and **ghe/ghi** with the corresponding phones or phone pairs, according to the vowel / semivowel information previously acquired.
4. For the syllables which carry the diphthong **iu**:
    a. If the lexical stress is positioned on the **i** letter, **i** becomes a vowel and **u** is a semivowel.
    b. If the lexical stress was predicted on the letter **u**, or the lexical stress is missing, **u** becomes a vowel and **i** is a semivowel.
5. For the word ending syllable which finishes in **i** letter, preceded by a consonant:
    a. If the lexical stress is predicted on the **i** letter, **i** becomes a vowel.
    b. If **i** carry no lexical stress, **i** is phonetically replaced by the **final-i** phone.

4. Experiments and results

4. 1. Experiments for syllabification

We started our experiments with an initial T corpus composed of 56,284 correctly syllabified words, from which we extracted a first 20% test set.

The syllabification algorithm was tested for different *n*-gram histories, and the best results were found for a length of 5, as can be seen in Table 1.

Table 1

Syllabification results for various *n*-gram lengths

***n*-gram length (history)**	**WER (%)**
2	28.62
3	15.54
4	6.30
5	**2.86**
6	3.92

It can be noticed a fast algorithm performance improvement until a history of 5, but after this level the performance slightly decreases. We consider this degradation to be normal; it indicates both the saturation of the training data and the limits of our method, which does not use smoothing techniques under the level of trigrams, for matters of speed.

Three different training / testing experiments have also been performed on these data, by splitting the initial T corpus in three different T1 and (T–T1) training and respectively testing corpora, but always keeping the same 80% versus 20% ratio. In this manner, training and testing words were always different in these experiments.

The results of the three tests are shown in Table 2.

Table 2

Syllabification results for three different tests

	Test words	**Errors**	**WER (%)**
Test 1	11,256	347	3.08
Test 2	11,257	331	2.94
Test 3	11,257	322	2.86

The overall performance of the syllabification algorithm, calculated as the media of the three tests accomplished, is a **2.96 % WER**.

4.2. Experiments for the L2P conversion

We tested the proposed L2P conversion algorithm on a manually-corrected test corpus which has 11,819 different inputs. The result (in terms of the WER) is presented in Table 3, and the error distribution, for different phonetic ambiguities, is given in Fig. 4.

Table 3

Overall L2P conversion result

Words	**Errors**	**WER (%)**
11,819	356	**3.01**

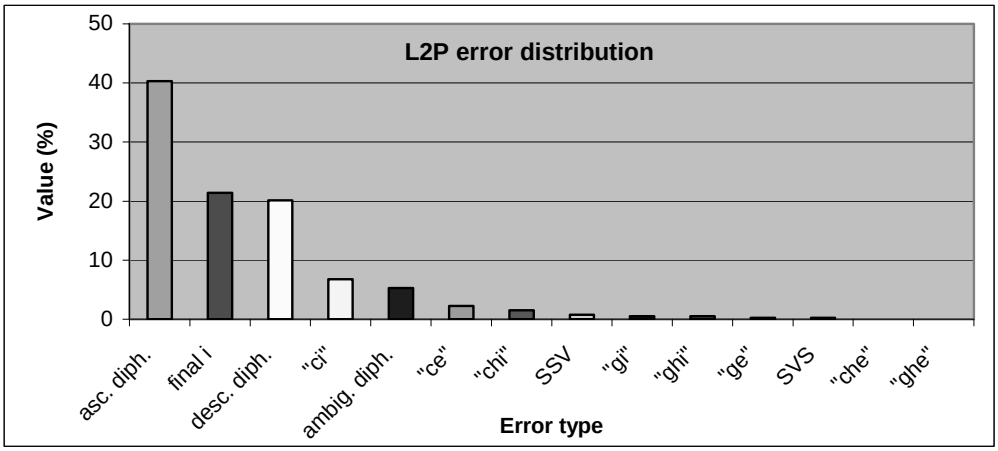

Fig. 4. Error distribution for the L2P conversion algorithm

5. Conclusions

For a preliminary version of the syllabification algorithm, we applied only a statistical approach and ignored any syllabification rules. We obtained a minimum of 5% WER, which represented a result fairly similar to the one reported in Section 4. However, a notable drawback of this method was an increased processing time for longer words (i.e., with more than 10 characters), with a negative impact on the complete TTS system processing time.

For this reason, we proposed a hybrid rule-based / data-driven approach, by implementing first a minimal set of deterministic syllabification rules. This idea has proven to be superior to the previous attempt, as both an improved syllabification performance and an increased speed processing have been obtained. An overall result of 2.96 % WER was obtained during tests.

Regarding the phonetic transcription, a detailed analysis of the L2P conversion results, made for every incorrectly-returned phone, as well as several complex listening tests that have been conducted for this purpose, proved that for the overall TTS system, an approximate number of 130 wrong phonetic conversions (out of 356) can still be considered as acceptable; the perceived quality of the synthesized speech is almost not affected by these transcription errors. This observation led to a final L2P conversion result of about 2% WER.

Acknowledgement

The research reported in this paper was funded by the Romanian Government, under the National Research Authority CNCSIS grant IDEI no. 782/2007.

R E F E R E N C E S

[1] *D. Burileanu*, "Basic Research and Implementation Decisions for a Text-to-Speech Synthesis System in Romanian", in International Journal of Speech Technology, **vol. 5**, no. 3, Kluwer, Dordrecht, Sept. 2002, pp. 211-225

[2] *D. Burileanu, C. Negrescu*, "Prosody Modeling for an Embedded TTS System Implementation", Proceedings of the 14th European Signal Processing Conference (EUSIPCO), Florence, 2006, pp. 715–718

[3] *D. Burileanu, C. Negrescu and M. Surmei*, "Recent Advances in Romanian Language Text-to-Speech Synthesis", in Proceedings of the Romanian Academy, Series A – Mathematics, Physics, Technical Sciences, Information Science, **vol. 11**, no. 1, Publishing House of the Romanian Academy, Bucharest, 2010, pp. 92-99

[4] *C. Ungurean, D. Burileanu, V. Popescu, C. Negrescu and A. Dervis*, "Automatic Diacritic Restoration for a TTS-based E-mail Reader Application", in UPB Scientific Bulletin, Series C, **vol. 70**, no. 4, Politehnica Press, Bucharest, 2008, pp. 3-12

[5] *C. Ungurean, D Burileanu, C. Negrescu and A. Dervis*, "A Statistical Approach to Lexical Stress Assignment for TTS Synthesis", in International Journal of Speech Technology, Springer Netherlands, **vol. 12**, no. 2-3, 2009, pp. 63-73

[6] *S.A. Toma, E. Oancea and D.P. Munteanu*, "Automatic Rule-based Syllabification for Romanian", in From Speech Processing to Spoken Language Technology, Publishing House of the Romanian Academy, Bucharest, 2009, pp. 87-94

[7] *O. Buza, G. Toderean*, "Syllable Detection for Romanian Text-to-Speech Synthesis", Proceedings of the 6th International Conference on Communications, Bucharest, June 2006, pp. 135-138

[8] *M.A. Ordean, A. Saupe, M. Ordean, M. Duma and Gh.C. Silaghi*, "Enhanced Rule-Based Phonetic Transcription for the Romanian Language", Proceedings of the 11th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC), Timişoara, Sept. 2009, pp. 401-406

[9] *Romanian Academy, Iorgu Iordan – Al. Rosetti Lingvistic Institute*, Dicţionarul ortografic, ortoepic şi morfologic al limbii române, Ed. a 2-a, Editura Univers Enciclopedic, Bucureşti, 2005

[10] *F. Şuteu, E. Şoşa*, Dicţionar ortografic al limbii române, Editura ATOS, Bucureşti, 1993.

[11] *S.M. Katz*, "Estimation of Probabilities from Sparse Data for the Language Model Component of a Speech Recognizer", in IEEE Transactions on Acoustics, Speech and Signal Processing, **vol. 35**, no. 3, 1987, pp. 400-401.