U.P.B. Sci. Bull., Series C, Vol. 70, Iss. 4, 2008 ISSN 1454-234x

AUTOMATIC DIACRITIC RESTORATION FOR A TTS-BASED E-MAIL READER APPLICATION

Cătălin UNGUREAN[1], Dragoş BURILEANU[2], Vladimir POPESCU[3], Cristian NEGRESCU[4], Aurelian DERVIŞ[5]

Sinteza automată a vorbirii reprezintă o tehnologie importantă pentru aplicațiile care au ca suport rețelele de date. Astfel de aplicații, cum ar fi cele de citire a mesajelor de tip e-mail sau SMS, trebuie să rezolve numeroase probleme, una dintre acestea fiind necesitatea refacerii diacriticelor în textul original. Lucrarea propune un algoritm eficient de poziționare automată a diacriticelor pentru un sistem de sinteză pornind de la text în limba română, utilizat într-o aplicație de citire a poştei electronice. Algoritmul utilizează metode statistice axate pe n-grame, se bazează pe cunoştințe lingvistice limitate şi necesită un corpus de antrenare de dimensiuni medii.

Speech synthesis technology is becoming more important for network-based applications. However, an e-mail or SMS reader application based on TTS (text-to-speech) technology, for example, needs to face many difficulties, one of them being the requirement for restoring missing diacritics to text, as a common problem for many languages that use the Latin alphabet. The paper proposes an efficient automatic diacritic restoration algorithm for a TTS system in Romanian used in an e-mail reader application. The algorithm is essentially based on a statistical strategy that uses n-gram similarity measures, relies on limited linguistic knowledge and needs a medium-sized training corpus.

Keywords: Text-to-speech synthesis, diacritic restoration, n-grams

1. Introduction

While telephone speech recognition is already the largest market for speech technology, it must be noticed that there is a broad range of services and

[1,] PhD student, Faculty of Electronics, Telecommunications and IT, University POLITEHNICA of Bucharest, Romania
[2] Prof., Faculty of Electronics, Telecommunications and IT, University POLITEHNICA of Bucharest, and Romanian Academy Center for Artificial Intelligence, Bucharest, Romania
[3] Assistant Prof., Faculty of Electronics, Telecommunications and IT, POLITEHNICA University of Bucharest, Romania, and Laboratoire d'Informatique de Grenoble, Grenoble Institute of Technology, France
[4] Prof., Faculty of Electronics, Telecommunications and IT, University POLITEHNICA of Bucharest, Romania
[5] PhD student, Faculty of Electronics, Telecommunications and IT, University POLITEHNICA of Bucharest, Romania

products that already integrate synthesized speech into their communication facilities. On the other hand, in many network-based applications one cannot predict the message that needs to be spoken, and the system must generate sentences from arbitrary text (database records, e-mail messages, etc.). This task can be accomplished only by text-to-speech synthesis systems, which must provide at least very good intelligibility for the resulting speech to be helpful and accepted by the user [1].

Besides this requirement (necessary to deal with the reduced telephone bandwidth and other channel degradation factors), an e-mail or SMS reader application based on TTS technology needs to meet another important constraint, namely handling the missing diacritics problem.
Usually, the majority of users still disregard the diacritics (marks above, through, or below letters), even if the terminal (or the operating system) allows this operation, or the diacritics are simply stripped by a computational process that uses 7-bit forms [2], [3], [4]. This is the case for Romanian, but also for many languages that use the Latin alphabet (such as French, Spanish, German, Dutch, Hungarian, Polish, Swedish, etc.) and supplement the basic set by making use of letters with diacritics (or accent marks) to indicate additional sounds (or stress). On the other hand, synthesizing a text generated without diacritics usually leads to a poor intelligibility; sometimes, syntactic or semantic ambiguities make certain sentences entirely incomprehensible. Unfortunately, the automatic restoration of diacritics in textual media where they are missing is a very difficult problem, as there are not obvious linguistic rules to accomplish this task.

Our team has a long experience in concatenative TTS synthesis. A desktop (PC-based) TTS system in Romanian language and also an embedded version are available for several years [1], [5]. The newest version of this system was used to develop an e-mail reader platform, involving common telephony interface, POP3 (Post Office Protocol) standard communication and off-the-shelf hardware. The application permits to listen to messages from an e-mail account by accessing the platform from a legacy POTS (Plain Old Telephony System) terminal – voice only capable; this platform is at present fully functional [3].

The main purpose of this paper is to describe one of the natural language processing modules of the TTS system used in this application. More specifically, we propose an efficient and accurate automatic diacritic restoration algorithm for the Romanian language. The algorithm is essentially based on a statistical strategy that uses *n*-gram similarity measures, relies on limited linguistic knowledge and needs a medium-sized training corpus. The paper is structured as follows. Section 2 briefly discusses the automatic diacritic reconstruction problem as well as the available approaches, and then describes the philosophy of the proposed algorithm. Experiments and results are presented in Section 3. Section 4 concludes the paper with final remarks.

2. Automatic diacritic restoration

2.1. Preliminaries

The automatic restoration of diacritic signs is of premium importance in many applications that need to process electronically stored texts, from Web information retrieval and indexing to TTS synthesis of SMS or e-mail messages. Hence, several methods and algorithms have been proposed in this respect; some of them are pure statistical methods and do not rely on any linguistic knowledge, others use language models and various levels of linguistic processing. For example, in [2] Németh et al. describe an algorithm for Hungarian that uses a vocabulary based on a large training corpus and simply chooses for a given word written without diacritics its most probable version with diacritics. Also from the first category, a diacritic restoration algorithm that uses a machine learning technique operating at letter level and applied to Czech, Hungarian, Polish, and Romanian (for which the performance obtained is about 98.30 %) is discussed in [6] by Mihalcea and Nastase; De Paw at al. present in [7] a similar approach, but with slightly poorer performance.

On the opposite side, Tufiş and Chițu propose in [4] a more complex algorithm for diacritic insertion in Romanian texts based on POS (part-of-speech) tagging, which yields at word level an overall accuracy of 97.4 %. A similar work based on Hidden Markov Models and learning at word level is described by Simard in [8] for French. Finally, we mention the contribution of Yarowsky, which discusses in [9] and [10] several methods for restoring accents in Spanish and French; best results were obtained by using hierarchies of decision lists, classifications being based on a large set of morphological contextual features.

The main objective for our team was to develop an accurate automatic diacritic restoration algorithm, thus well suited for a high-quality TTS synthesis task. Accordingly, the first and simplest approach consisted in randomly inserting diacritics in words that *might* contain such signs, using a dictionary. Thus, an accuracy around 88 % was obtained. The second method, which yielded better results, relied on unigrams for replacing the ambiguous words in test corpora. This method, although context-independent, resulted in an accuracy of only 95 %. These results confirmed that, in ambiguous cases, one word version is usually dominant. Clearly, further increasing the accuracy asks for more refined approaches. Such a method, proposed in this paper, aims at choosing the most likely version of ambiguous words, also inspecting their context of occurrence.

The first novel aspect introduced by our method consists of a sequential filtering process that relies on the likelihood of word tokens *and* on their context of occurrence. Each filtering criterion brings a further gain, keeping at the same time the ambiguous words, according to a certain filtering level considered. In the experiments reported in this paper, three such filtering stages were used (at word unigram level, at word bigram level, and at suffix trigram level), but the number of these filters can be augmented. The main advantage of such an approach resides clearly in that each filtering stage can only improve or keep unaltered the results for the preceding filter.

A second novelty is represented by the fact that a *minimal* filtering is performed, i.e., the main decision criterion is the existence of word-level or *suffix*-level *n*-grams, in a training corpus. As for the performance criterion that drives the decision process, we chose to follow the reduction of diacritic insertion errors. This is particularly motivated by the needs of speech synthesis of text without diacritics. Yet, another feature of our diacritic restoration algorithm consists of using word suffix-level *n*-grams; this is particularly motivated for morphologically rich languages (Romanian is such a language), as [9] points out. Moreover, in the Romanian language the greatest number of ambiguous words (concerning the presence of diacritics) is yielded precisely by characters contained in word suffixes.

2.2. Algorithm description

The Romanian language makes use of three diacritic marks, i.e. a breve, a circumflex accent, and a cedilla, leading to five letters with diacritics: *ă*, *â* / *î* (used as the same sound, but in different circumstances), *ş* and *ţ*, that lead to four ambiguity classes: *a* / *ă* / *â*, *i* / *î*, *s* / *ş* and *t* / *ţ*. Some diacritics indicate only a different noun form (e.g., *cas**ă*** – a house, and its pair *casa* – the house); in other situations their presence leads to a completely distinct meaning (e.g., *fata* – the girl, but *fa**ţ**a* – the face). We must note that the percentage of words written with diacritics in a Romanian text is substantial: between 25 % and 40 % of the total number of words [4]. In French, for example, only about 15 % of the words from an arbitrary text carry accents [8]. However, other languages, such as Dutch, Czech, and Slovak, make use of a much larger number of diacritics compared to Romanian [6].

After studying large corpora texts from various domains, we established that Romanian words can be classified into the following five categories:

① Words without diacritics;
② Words always written with diacritics (*c**â**teva* – some, ***ş**tiin**ţ**ific* – scientific);
③ Ambiguous words with a single possible diacritic (*dou**ă*** – two and its pair *(a) doua* – the second);
④ Ambiguous words where a number of diacritics are always present (*cămaşă* / *cămaşa* – a shirt / the shirt);
⑤ Ambiguous words where any diacritic can be present or not – multiple diacritic patterns (*pan**ă*** / *pana* / *p**â**n**ă*** – a feather / the feather / until).

This word categorization guided our diacritic restoration algorithm, which includes a training stage and a test stage. The training side of the algorithm encompasses the following steps:

1. Manually building a dictionary of the most used Romanian words, containing also as many inflected forms as possible; this dictionary is denoted by D_1.
2. From $D1$ build a structure $D2$ that contains, for each word in $D1$, a mapping rule connecting the word form with all its diacritics removed, to all the possible versions (with diacritics) for that word.
3. Using text corpora containing correctly spelled words with diacritics, build a set U of word unigrams, a set B of word bigrams, and a set T of trigrams, the latter at a word suffix level; this step is performed automatically.

It is important to note that we do not take the word “suffix” in its strict morphological sense; here, “suffix” means rather a number of terminal characters for each word. This number has been chosen between 2 and 4, based on empirical insights provided in the test stage (so that the character-level diacritic insertion error rate is minimized).

Concerning the test side of the algorithm, it includes mainly three cascaded filtering processes, where the output of each stage represents the input of the subsequent stage (the input to the overall algorithm consists of texts where the diacritics have been stripped off):

1. Using the word dictionary D_1, correct the unambiguous words that always contain diacritics (that is, words of type ② above); then, using the structure D_2, correct the words that should always occur without diacritics.
2. For each pair of consecutive words in the test text, that contain at least one ambiguous word, form the set of alternative versions that contain these words; then, if at least one member of this set is found in the list of bigrams B, discard the alternative pairs in the set, that are not in B; else, leave unaltered the set concerned.
3. For the input text that contains the alternatives output at step 2, split it into word triples and extract the suffixes; we thus obtain sets of alternative word and suffix triples; if at least one member of the suffix triples set is in the list of trigrams T, then discard all the members in that set, that are not in T (and, by consequence, discard the corresponding word triples as well); else, leave the suffix (and word) triples unchanged.
4. Using the set of word unigrams (U), keep only the words that occur with the highest probability in the training set, out of all the possible ambiguous words in the text output at step 3.

A few remarks are worth mentioning concerning this algorithm. First, in bigram and trigram-based filtering processes, the probabilities of those n-grams were not used; this is due to the fact that we did not impose any preference on word contexts (since this should be performed, in our view, at morphological or syntactic levels, driven by a grammar specific to the language in focus). However, in the unigram filtering process, occurrence probabilities were used, in order for this stage to be effective in further eliminating ambiguous words; thus, ambiguous word situations that could not be handled by contextual representations (word bigrams and word suffix trigrams) are ruled out at a word level, relying on occurrence frequencies. Bigram or trigram probabilities could be used as well, but these would request bigger training corpora, so that reliable statistics could be derived; yet, one of the advantages of our method is that it relies on a relatively small amount of training data (as shown in Section 3).

It should also be emphasized that the diacritic restoration algorithm described in this paper has been developed in an incremental manner, handling first the words always written with diacritics (or without diacritics), then gradually adding contextual information (such as word bigrams and word suffix trigrams), and finally filtering non-disambiguated words using their frequencies in a training corpus. Each processing level added a word error rate reduction; however, according to empirical studies of our team, the most successful series of filtering steps were those finished by unigram-based handling of ambiguous words. This fact is theoretically provable as well, since each filtering stage, except for the unigram-based one, relies on word (or word pairs or suffix triples) occurrences, not taking into account occurrence scores. Hence, for a given test text, each filtering stage (again, except for the unigram-based one) returns one or more possible words (or word pairs, or suffix triples), which include the correct words. Each filtering stage refines (i.e., reduces) this set, but it is only the unigram-based filtering process that returns only one element in each set of alternatives, namely the most frequent, discarding the rest. This is why in the experiments reported in Section 3 only results obtained using the unigram-based filtering as the last stage of the algorithm will be shown.

3. Experiments

3.1. Evaluation Context

The algorithm described in Section 2 has been trained on a corpus of literary Romanian language texts containing over 10 million word tokens[1]. These texts helped also to automatically add new words in D_1 dictionary (which was first manually built, as noticed in Section 2.2); this dictionary contains at present around 330,000 words (including approximately 6,000 proper nouns). Then,

[1] These data were taken from the following publically available on-line sources: http://www.liternet.ro, http://www.romanialibera.ro, http://www.romanialiterara.ro, and http://ro.wikisource.org.

word-level bigrams and word suffix-level trigrams were extracted from the training texts and stored as separate lists in text files.

The test corpus (manually corrected) contains journal and literature texts, along with Ph.D theses in several domains and other documents available in electronic form in the Romanian language, and has the following characteristics: (i) number of word tokens: 1,200,000 words, (ii) the percentage of words without diacritics: 57.41 %, (iii) the percentage of words always written with diacritics: 16.33 %, (iv) the percentage of ambiguous words (i.e., with multiple diacritic patterns): 26.26 %.

3.2. Performance measures

Using these data, *precision* (defined as the ratio between the number of correctly inserted diacritics and the total number of diacritics inserted by the algorithm), *recall* (defined as the ratio between the number of correctly inserted diacritics and the total number of diacritics in the manually-corrected test database) and *F-measure* (defined as the harmonic mean of precision and recall) were computed, at a character level. In Table 1 we show these measures, detailed for each of the four ambiguity classes; measurements are shown for three versions of the algorithm: (i) a baseline method that relies only on the mapping rules in D_2 and does not use any training data, (ii) a method that uses bigrams extracted from the training corpus, along with word unigram statistics trained on that corpus, (iii) the complete algorithm, that consists on a cascaded bigram and trigram-based filtering, along with the word unigram statistics. In this table we have also shown weighted average *F*-measures, for each ambiguity class; the weights are computed at a character level as well. From these results we can deduce that an overall *F*-measure of 99.34 % is achieved.

From Table 1 it can also be observed that bigram filtering brings a substantial performance improvement at character level, for all ambiguous classes. Furthermore, word suffix trigrams increase the performance mainly for the *a* / *ă* / *â* class which raises most of the problems in Romanian; this improvement comes mainly from the tokens that include diacritics in their suffixes. At the same time, it is important to notice that word suffix trigrams do not alter the high results achieved for the other three ambiguous classes.

It is worth mentioning that we performed experiments in using word prefix-level trigrams as well, but this method did not bring any performance improvements; this can be explained by the nature of inflection patterns in Romanian, where prefixes are rather independent of the context where words occur.

Table 1

Diacritic restoration algorithm performance measures

Ambiguity class	**Precision (%)**	**Recall (%)**	***F*-measure (%)**	**Average *F*-measure (%)**
Baseline				
a / ă / â	78.03 / 92.69 / 99.50	99.65 / 35.27 / 61.86	87.53 / 51.10 / 76.29	77.11
i / î	98.63 / 99.67	99.96 / 89.54	99.30 / 94.34	98.71
s / ş	94.83 / 97.89	99.48 / 81.48	97.10 / 88.93	95.25
t / ţ	95.30 / 91.25	98.82 / 71.63	97.03 / 80.25	94.57
Bigrams and unigrams				
a / ă / â	97.82 / 91.58 / 99.24	96.49 / 94.66 / 99.57	97.15 / 93.10 / 99.40	96.19
i / î	99.97 / 99.57	99.94 / 99.82	99.95 / 99.69	99.93
s / ş	99.93 / 99.49	99.85 / 99.76	99.89 / 99.63	99.83
t / ţ	99.86 / 98.07	99.66 / 99.22	99.76 / 98.64	99.60
Bigrams, trigrams and unigrams				
a / ă / â	98.22 / 93.30 / 99.26	97.22 / 95.62 / 99.55	97.72 / 94.45 / 99.41	96.93
i / î	99.98 / 99.56	99.94 / 99.82	99.96 / 99.7	99.93
s / ş	99.93 / 99.51	99.85 / 99.77	99.89 / 99.64	99.84
t / ţ	99.87 / 98.2	98.69 / 99.27	99.78 / 98.74	99.63

To sum up, the weakest character and also word level performances were obtained for the ambiguous class *a* / *ă* / *â*, in agreement with results reported in [6] and [4]. Although several frequent error situations have been significantly reduced, there remains a set of ambiguous words that the algorithm cannot handle correctly. Results regarding these aspects are shown in Table 2 (in number of words).

We can see that the highest number of erroneous diacritic assignments is yielded by the ambiguous pair *ca* / *că* (as / that), which occurs with a high frequency in the test data. Other situations that are difficult to handle correctly consist in ambiguous pairs containing feminine nouns and some verbs.

Moreover, there are a number of limitations that cannot be, a priori, mitigated by the algorithm; these include:

- proper nouns or very infrequent words, not present in the D_1 dictionary; a practical solution to this would be to simply add, when needed, such words to D_1 and, consequently, to the mapping rules in D_2;
- discourse context-dependent word forms, where the sentence level context does not suffice to disambiguate the appropriate word; such an example in Romanian is *Am văzut o **fată** frumoasă* (I saw a beautiful **girl**), versus *Am văzut o **faţă** frumoasă* (I saw a beautiful **face**), where other utterances are needed in order to handle the ambiguity *t* / *ţ* in *fată* / *faţă* (girl / face).

Table 2

Word-level filtering performances examples

Word	**Baseline**	**Bigrams and unigrams**	**Bigrams, trigrams and unigrams**
ca / că (as / that)	19,718	4,890	2,478
fată / faţă (a girl / a face)	2,199	912	495
sau / său (or / his)	2,548	441	350
sa / să (her / to)	30,064	395	341
lua / luă (was taking /took)	547	330	272
uşa / uşă (the door / a door)	1009	362	267
banca / bancă (the bank /a bank)	507	248	243

Concerning word-level recognition scores, we indicate that an average accuracy of 97.87 % was obtained on the same test data.

4. Conclusions

This paper described a complete and accurate algorithm for automatic restoration of missing diacritics from texts in Romanian. The performance obtained by the proposed algorithm can be summarized as follows: an overall *F*-measure of 99.34 % at character level, and an average accuracy of 97.87 % at word level. These results are better than those reported in literature so far (as it was discussed in Section 2.1), including those communicated for Romanian language, in [6], [7], and [4]. Compared, for example, with the contribution of Mihalcea and Nastase (they reported in [6] an overall *F*-measure of about 98.30 %), we can make two comments. First, their test corpus comprises only about 50,000 words, which means about 25 times less than our test database. Secondly, we consider that their approach based on looking at the surrounding letters is not appropriate for Romanian, where a large number of ambiguities (especially those of articulated versus non-articulated feminine nouns) could be solved only at word level.

The described algorithm is used at present in a TTS-based e-mail reader application. In fact, the algorithm is part of the pre-processing stage of the TTS system, and runs just after input text segmentation, punctuation marks detection, and substitution of upper case letters into lower case letters. We mention that punctuation marks, numerals, and abbreviations are ignored by the diacritic restoration algorithm and are processed (or normalized) by subsequent modules of the text analysis stage [5].

In the near future we plan to take into account further contextual effects on diacritic patterns, namely by using discourse-related information in order to disambiguate words that are still difficult to handle by the current version of the algorithm.

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*, Spoken Language Interfaces For Embedded Applications, in Human Factors and Voice Interactive Systems (D. Gardner-Bonneau and H. Blanchard – Eds.), 2nd Edition, Springer, Norwell, pp. 135-161, 2007.

[2] *G. Németh, Cs. Zainkó, L. Fekete, G. Olaszy, G. Endré-di, P. Olaszi, G. Kiss, P. Kiss*, The Design, Implementation and Operation of a Hungarian E-mail Reader, in International Journal of Speech Technology, Kluwer, Dordrecht, **vol. 3**, no. 3/4, pp. 217-236, 2000.

[3] *M. Surmei, D. Burileanu, C. Negrescu, R. Pîrvu, C. Ungurean, A. Derviş*, Text-to-Speech Engines as Telecom Service Enablers, in Advances in Spoken Language Technology, Publishing House of the Romanian Academy, Bucharest, pp. 89-98, 2007.

[4] *D. Tufiş, A. Chiţu*, Automatic Diacritics Insertion in Romanian Texts, in Proc. COMPLEX'99, Pecs, Hungary, pp. 185-194, June 16-19, 1999.

[5] *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, pp. 211-225, Sep. 2002.

[6] *R. Mihalcea* and *V. Nastase*, Letter Level Learning for Language Independent Diacritics Restoration, in Proc. CoNLL 2002, Taipei, Taiwan, pp. 105-111, 2002.

[7] *G. De Paw, P. W. Wagacha,* and *G. M. de Schryver*, Automatic Diacritic Restoration for Resource-Scarce Languages, in TSD 2007 – LNAI 4629 (V. Matousek and P. Mautner – Eds.), Springer, Berlin, pp. 170-179, 2007.

[8] *M. Simard*, Automatic Insertion of Accents in French texts, in Proc. EMNLP-3, Granada, Spain, pp.27-35, 1998.

[9] *D. Yarowsky*, A comparison of corpus-based techniques for restoring accents in Spanish and French texts, in Proc. 2nd Annual Workshop on Very Large Corpora, Kyoto, Japan, pp. 19-32, 1994.

[10] *D. Yarowsky*, Corpus-based Techniques for Restoring Accents in Spanish and French Text, in Natural Language Processing Using Very Large Corpora, Kluwer, Norwell, pp. 99-120, 1999.