Diacritics Restoration in Romanian Texts

Dan Tufiş, Alexandru Ceauşu
Institute for Artificial Intelligence, Romanian Academy.
"13 Septembrie", 13,050711, Bucharest 5
{tufis, aceausu}@racai.ro

Abstract

There are several languages that use diacritical characters outside the ASCII charset. For some of the languages, most diacritical characters can be deterministically recovered but in general, this is not the prevailing case. However, the difficulty of the task differs from language to language depending on the functional role of the diacritical characters. For Romanian, automatic restoration of the diacritics is a real challenge, both because of their frequency and due to their significant contribution to the morpho-lexical and semantic disambiguation of the words.

In Romanian, every third word might contain at least one diacritical character and for large texts that lack diacritics, to insert them manually is highly time-consuming, boring and error-prone.

We present a professional implementation, embedded into MS Office environment, which builds on our tiered tagging technologies.

Keywords

diacritics restoration; part-of-speech tagging, tiered tagging

1. Introduction

Spelling verification and correction is one of the oldest natural language processing applications that were used on large scale. Most of the spell checkers are focused on correcting the typographical errors and cognitive errors. Typographical errors are the most common variety of spelling errors. In typed text, a majority of the misspellings result from errors such as transposing characters, pressing the wrong key, omitting a character, inserting an unnecessary character, or omitting a space between words. Cognitive errors occur when the user does not know the standard spelling of a word. In this case, the user typically enters a phonetic spelling. But there is another important aspect of spelling verification and correction that does not occur in English and several other languages: diacritics insertion.

To find a way to automating the diacritics insertion is worthy not only for old valuable texts stored in electronic form, but also for contemporary electronic texts as they continue to be produced in non-diacritical form. The reasons for this could be many, including the lack of localized and standardized keyboards. Ergonomic factors can also be mentioned (if someone is supposed to press more than two keys to get a diacritical character, then, mainly in informal communication (e.g. e-mail), he/she will probably take the easiest one-stroke solution).

We present a diacritics restoration program for Romanian language, based on recent advances in probabilistic tagging technology. Similar approaches have been proposed in [8] for Romanian, [7] for French, [2] (cf. [7]) also for French. Mihalcea [5] addresses the restoration of diacritical characters in Romanian using an n-gram model. Yarowsky [14] addresses this problem for Spanish (mainly) and French but instead of POS tagging, he uses a decision-list framework which offers very satisfactory performance (speed & accuracy) in spite of a language model that "was admittedly quite weak: in the absence of a hand-tagged training corpus, he based his model on an *ad hoc* set of tags" [7]. As compared to French, Romanian makes more intensive use of diacritical signs and their absence creates much more difficulties.

2. Diacritics in Romanian

Romanian language has 5 diacritical characters: *ă,â,î,ş* and *ţ* (plus their uppercase variants). A text missing the diacritics will usually have these characters substituted by *a* (for both *ă* and *â*), *i*, *s* and *t* respectively. For a significant part of the words with the diacritics stripped-off their recovering is deterministic, because the non-diacritical variants of those words are not legal lexemes of Romanian. But in most of the cases, the absence of diacritics creates genuine ambiguity, hard to resolve sometimes even for a human (when given only a limited context).

Here are some examples of strings that if missing diacritics are not legal words of Romanian (the real word and its translation are specified between parentheses):

A) padure (*pădure*- forest), tufis (*tufiş*- bush), autorizatie (*autorizaţie*- autorisation), cantar (*cântar*- balance), carare (*cărare*- pathway), macar (*măcar* – at least), fara (*fără* – without), cati (câți -how many) etc.

We call such strings unambiguous stripped words, or *U-words.*

To exemplify the ambiguity caused by the lack of diacritics, let us consider the string *fata*. In a text where the diacritics were removed, this string could stand for any of the following words:

B) *fata* – the girl, *fată* – a girl; or (about animals) gives birth , *fâţa* – the quick-swimming little fish/the coquette, *fâţă* – a quick-swimming little fish/a coquette, *faţa* – the face, *faţă* – a face, *făta* – (about animals) to give birth; gave birth, *fătă* – (about animals) just gave birth.

All the strings of the *fata* type above (i.e which could stand for more than one diacritical or non-diacritical word) are referred to in the following as ambiguous stripped words, or *A-words.* The strings that are neither U-words nor A-words are simply referred to as words.

We found that the morpho-syntactical information disambiguates most A-words. Yet, there exist subsets of A-words for which morpho-syntactic descriptions are identical and diacritics restoration distinction could be made only based on meaning:

C) *fata* (Ncfsry) – the girl, *fâţa* (Ncfsry) – the quick-swimming little fish/the coquette, *faţa* (Ncfsry) – the face.

These words, which we call *C-words*, require sense disambiguation. The C-words are a subset of A-words.

The table in Figure 1 displays data we extracted from our reference corpora. The journalism corpus consists of articles from the weekly magazine "Agenda" from Timişoara (years 2003-2006). The juridical corpus is a collection of around 6000 Romanian documents extracted from the Jrc-Acquis corpus [8]. The part-of-speech annotation was made with our tiered tagger in order to reduce as much as possible the number of tagging errors. The total number of words shown in Table 1 (line 1) does not include numbers or tokens containing one or more digits, proper names, foreign words (tagged by the X tag), abbreviations (tagged by the Y tag), dates (tagged by the DATE tag) and punctuation. From the total number of tokens in the mentioned texts, the discarded tokens account for 35.09% and 26.04% respectively. The big difference between the number of discarded items in the two corpora is due to the fact that journalism corpus contains lots of numbers (in the "Sport" sections one can find scores, minutes, timings, distances etc) dates, foreign words and abbreviations. These categories are not significant for the diacritics restoration problem because, in the vast majority of cases, they do not contain diacritical signs. However, the proper names in Romanian are words that might contain diacritics, thus being relevant for the diacritics restoration task. Yet, in the juridical corpus, although the names are quite frequent, none of them contained diacritics. Thus, in order to make a meaningful comparison among the two register data, we excluded the proper names from our analysis.

There are two different figures for C-words, depending on what tagset was used in POS tagging: a reduced tagset (Ctag-set in line 5) and the lexicon morpho-syntactic descriptors tagset (MSDtag-set in line 6). The two figures demonstrate that the diacritics restoration is more accurately done when the system has access to more linguistic contextual information. On the other hand, in general, using a reduced tagset as compared to a large one, increases the tagging accuracy, which is vital for our approach in diacritics restoration. The Ctag-set and MSDtag-set and the way we solved the tension between tagset cardinality and tagging accuracy are briefly discussed in section 3.2.

Table 1. The distribution of the words with diacritics in texts of different registers

Corpus	Journalism	Juridical
1. Words	6, 680,448	3,511,093
2. Words with diacritics (out of 1.)	2,004,763 (30,01%)	1,026,385 (29,23%)
3. U-words (out of 2.)	238,132 (11,88%)	175,822 (17,13%)
4. A-words (out of 2.)	1,766,631 (88,12%)	850,563 (82,87%)
5. C-words (Ctag-set, out of 4)	58,420 (3,31%)	38,323 (4,51%)
6. C-words (MSDtag-set, out of 4)	24,916 (1,41%)	16,463 (1,94%)

As one can notice in the table above, in regular Romanian texts, almost one third of the words contain at least one diacritical character (30.01% of the words in the journalism data contain on average 1,17 diacritical signs, while 29.23% of the words in the juridical texts contain on average 1,16 diacritical signs). Out of the diacritical words only a small percentage are U-words (11,88% in the journalism data and 17,13% in the juridical texts). That is, in an ideal setting, with a fully coverage dictionary available and a text with no typographical error other than the missing diacritics, about 25% (#A-words/#Words) of the total number of words in a running Romanian text would remain ambiguous. In a more realistic setting of a diacritics restoration process, this figure is significantly higher because no dictionary fully covers any possible text and most texts contain typing errors (other than missing diacritics). In our supposedly error free data we identified 72,722 (1,09%) typing errors in the journalism texts and 29,387 (0.84%) typing errors in the juridical texts. Below we list the main categories of errors:

a) even if a word contains diacritics, it might not contain all of the necessary ones (e.g. "invăţământ"[1] vs. "învăţământ", "lacătuş" vs. "lăcătuş" etc.);

b) even if a word contains diacritics, one or more of them might be wrong (e.g. "sărmă" vs. "sârmă", "câtre" vs. "către", "neâncăpător" vs. "neîncăpător" etc.)

c) even if a word contains diacritics, they might not be in accordance with the current orthography of Romanian (e.g. "considerînd" vs. "considerând", "curînd" vs. "curând" etc.)

[1] All the examples provided trhoughout this paper are extracted from real texts used during the evaluation.

d) the words (with or without diacritics) might be misspelled (e.g. "înopta" vs. "înnopta", "indenmizație" vs. "indemnizație", "compensdiu" vs. "compendiu" etc.) or miss-tokenized (e.g. "5%pentru" vs. "5% pentru", "20o" vs. "200" etc.)

e) a lexical token might be distorted by a combination of the above cases, making it very difficult to be recovered.

One could argue that a traditional spell-checker could fix these error-cases, but at least for Romanian, this is not entirely so, because of what we called A-words (words which remain legal words of the language even after the diacritics removal, e.g. "peste" (*over*) versus "peşte" (*fish*), "scoală" (*wake up*) versus "şcoală" (*school*), "barca" (*the boat*) versus "barcă" (*a boat*) etc. A standard spell-checker (such as A-spell or the one included into MS Office) would not even detect the A-words as possibly problematic.

In a traditional spell-checker solution, the "out of the dictionary" words are highlighted and the user is expected to select one correction from a list of possible choices (which might not include the proper correction). In our approach, most of the corrections (all but the C-words) are automatically performed without user going through each individual token. While the automatic procedure is practically done in no time, the manual procedure is error prone and even when assisted by a spell-checker might require hours, days or even months for very large textual data. For the C-words occurring in a text, the system behaves like a spell-checker, i.e. it requires the user to make a choice, out of a list of contextually plausible corrections. A contextual plausible correction should comply with the linguistic restrictions specified by the morpho-syntactic description associated to the respective C-word. For instance, if the C-word "fata" was tagged as a feminine noun, in a direct case, definite singular form, the plausible solutions are "fata" (the girl), "faţa" (the face), "fâţa" (the quick-swimming little fish/the coquette) all characterized by the same morpho-lexical attributes as the original C-word. All the other variants (fată, faţă, fâţă, fãta, fătă) should be ignored due to different morpho-syntactic descriptors (indefinite nouns or verbs)..

The last two lines in Table 1 show that when we use a tag-set of finer granularity (614 tags in the MSDtagset versus 92 tags in the Ctagset) the number of C-words (the tokens for which the diacritical forms cannot be deterministically recovered) is more than twice less than before. We noticed that the majority of the C-words in this case are genuine spelling errors, very few of them requiring sense disambiguation.

In the next sections we will briefly describe the underlying technologies used by our diacritics restoration system DIAC$^+$, provide an evaluation and few details on its implementation and conclude with a comparison between DIAC$^+$ and its ancestor described in [9].

3. Text pre-processing

Since the DIAC$^+$ was designed to work with MS formatted documents, the system extracts the textual data from the input file and stores it in an internal format adequate for our pre-processing tools, using as database a full-text search engine – Lucene. The textual data extracted from the input file is tokenized and tiered-tagged, thus creating a linguistic knowledge space for the current text within which the proper restoration of diacritics takes place.

3.1 Tokenization

The tokenizer is a program that identifies within the input text the elementary processing units called lexical tokens. A lexical token usually corresponds to the generally accepted idea of a word, namely a sequence of characters delimited by white spaces. However, several words may form a natural single unit (such as "*pentru că*" – because) or on the contrary, a sequence of characters delimited by white spaces may be split into distinct lexical units (such as "*dă-mi-le*" – you_(singular)_give to_me them = give them to me). The tokenizer also recognizes dates expressed in a large variety of formats (1 ianuarie, 1999; 01/01/99; 01-ian-99, etc), abbreviations (*dl, dna, dra*, *dr.* etc.), various types of punctuation, etc. Initially we used the MULTEXT tokenizer - a configurable language independent tool. However, the price the MULTEXT tokenizer pays for its language independence and flexibility was considered too high. We have developed our own tokenizer which although not equally flexible, is still language independent, smaller and at least 1000 times faster.

3.2 Tiered tagging

In highly inflectional languages, encoding the morpho-lexical properties of the wordforms requires a large set of description codes. The Multext European project in co-operation with EAGLES Lexical Specification Group developed a set of recommendations [6] for the languages in Western Europe. Starting with these specifications, the Multext-East Copernicus project further developed them so that to account for the specificity of six other languages from Central and Eastern Europe – Bulgarian, Czech, Estonian, Hungarian, Romanian and Slovene – [4] and developed large compliant lexical resources [13]. The set of morpho-syntactic descriptors (MSDs) specific to Romanian contains 615 codes.

It is well known that the larger the tag-set, the larger the training corpora needed [1] and unfortunately this is not a linear dependency. To avoid severe data sparseness and accuracy degradation, a huge amount of manual work would be necessary for building appropriately large training corpora.

Tiered tagging [10, 11] is a two-stage technique addressing the issue of data-sparseness. In general terms, tiered tagging uses a hidden tagset (we call it Ctag-set) of a smaller size (in our case 92 tags) on the basis of which a language model (LM) is built. This LM serves for a first level of tagging. Then, a second phase replaces the tags from the small tagset with contextually the most probable tags from the large tagset (we call it MSDtag-set) which contains 615 tags (MSDs). The fundamental idea in using the tiered tagging approach is that the attribute values in a MSD and the wordform are not independent. That is to say, having a MSD-based wordform lexicon, from a wordform and a subset of attribute-value pairs one could, in the vast majority of cases, deduce all the rest of the feature-values pairs characterizing the current wordform. In [10] we call this property the MSD-*recoverability*. The subset of the features in the MSDtag-set having the recoverability property represents the set of attributes in terms of which the Ctag-set is defined. In [10] we provided an algorithm to construct the Ctag-set from a MSD-based lexicon. In [11] we demonstrated that the algorithm is language independent and that the tiered-tagging approach is working very well for a completely different language than Romanian. In [12] we presented a further enhanced version of the Ctag-set automatic design and demonstrated its effectiveness on six languages (Czech, English, Estonian, Hungarian, Romanian and Slovene).

The lexicon, underlying the induction of the Ctag-set and backing-up the tiered tagging approach, contains the words annotated with the MSD tags, an entry having the form: <*word*> <*lemma*> <*msd*>. For Romanian, this lexicon, referred to in the following as LEX, contains more than 800,000 entries.

For a small number of the C-tags, the recovering process can face some ambiguities which have to be solved by using additional knowledge resource. In [10] this new resource is a set of hand-written contextual disambiguation rules. The applicability of both the deterministic and the rule-based recovering is limited only to the words recorded in the MSD tag-set lexicon. We replaced the second phase of the tiered tagging process with a maximum entropy-based MSD recovery [3]. In this approach, the rules for Ctag to MSD conversion are automatically learnt from the corpus and their application does not require looking-up the MSD tag-set lexicon. Therefore, even the Ctags assigned to unknown words can be converted into MSD tags. If an MSD-lexicon is available, replacing the Ctags for the known words by the appropriate MSD tags is almost 100% accurate.

4. Diacritics insertion

The overall architecture of DIAC$^+$ is shown in Figure 1. From the LEX lexicon, mentioned in the previous section, the system derives a diacritical words only lexicon (D_0), and a diacritics stripped-off lexicon (D_1) which are used to generate the hypotheses search space for the current text. Additionally, the system builds on the fly a list of words in the current text which are not in the previous dictionary but which could be considered typographical errors (D_2):

- D_0 dictionary is the subset of LEX containing all the diacritical words;
- D_1 dictionary is the diacritics stripped-off version of LEX; one should bear in mind that the entries from D_0 corresponding to A-words will differ among each other only by POS information
- D_2 dictionary contains words in the current text which are neither in D_0 nor in D_1 and which are suspected of being typing errors; they are associated with words in $D_0 \cup D_1$ differing by plus or minus one character or by switching two consecutive characters (additionally, the switched characters should be neighbors on the keyboard).

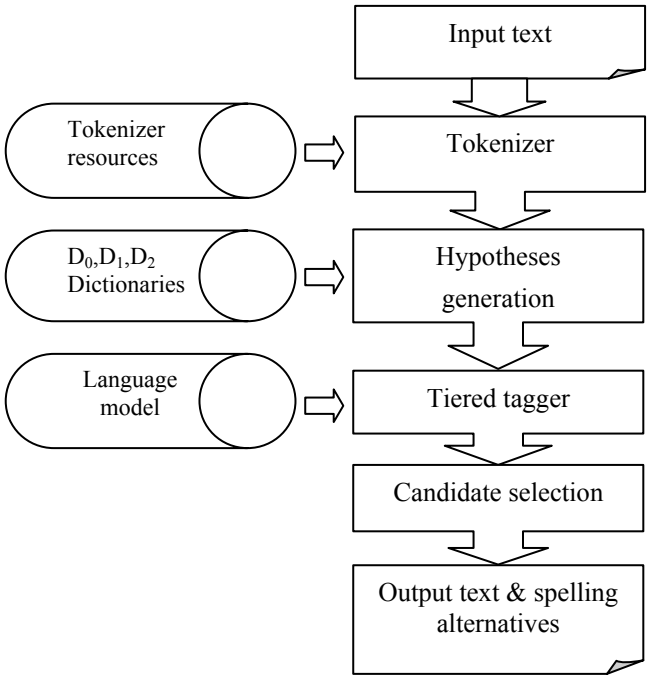

Figure 1. General architecture of DIAC$^+$

The procedure for automatic insertion of diacritics in Romanian texts has four stages:

(i) TOKENIZATION. The input text is segmented into lexical tokens according to the rules specified as external resources.

(ii) HYPOTHESES GENERATION. In the hypotheses generation step, a word is first searched in the union of D_0 and D_1 dictionary because in a text without diacritics or with partial diacritics one cannot be sure if a word is in its regular form or not unless contextual information is available.

If the word cannot be found in the union of D_0 with D_1 it is searched in the D_2 dictionary. A word which is not found in any of the system's lexicons is considered unrecoverable and left untouched.

In this step, a word *W*, occurring in the current text, may be associated with several entries in the LEX wordform lexicon and as such it will be associated with a set of pairs <*surface-form*$_k$ *MSD*$_k$> so that the diacritics stripped-off versions of the *surface-form*$_k$ and of *W* are identical. The information provided by the next tagging step will be used to filter this set and eventually to select the single contextually correct <*surface-form*$_i$>.

(iii) TIERED TAGGING. The text is tiered-tagged (tagged with the reduced tag-set, then each C-tag is mapped to a MSD-tag by the ME-tagger [3]; for this stage, only the MSDs from the hypothesis generation step are taken into consideration). In the case of unknown words the tagger chooses the best alternative resulted from the maximum entropy model. For tagging texts with partial or missing diacritics we used a special HMM language model in which the transition probabilities were computed from the regular training corpora (i.e. with diacritics) and the emission probabilities were computed from the diacritics stripped-off training corpora. This way the ambiguity classes for the words in the probabilistic lexicon and their respective POS lexical probabilities were modified, but the transition probabilities remained unchanged. For instance, the two unambiguous words *peste*/Spsa (eng. over) and *peşte*/Ncms-n (eng. fish) in the diacritics stripped-off training corpora will be represented by the same token type (*peste*) which in this case will become POS ambiguous (Spsa or Ncms-n). It is obvious that the spurious ambiguities created by the lack of diacritics degrade the tagging accuracy, but as discussed in [9] not all tagging errors are harmful for the diacritics restoration process.

(iv) CANDIDATE SELECTION. The U-words are replaced with their diacritical counterpart. The A-words which are not C-words are replaced by the *surface-form* identified by the MSD assigned by the tagger to the respective A-word. For the C-words, depending on the DIAC$^+$ variant (see further) either the user is presented with a list of contextually meaningful choices or the replacement is automatically done based on lexical probabilities or some probabilistic preferences.

5. Evaluation

For the evaluation purposes we used a reference corpus R, containing about 100,000 words. The reference corpus was hand tagged and lemmatized. We removed all the diacritics, from R but preserved the original tagging. This version of R is what we call the idealized DIAC$^+$ tagged text (TT): it has no tokenization or tagging errors, and no diacritical character is present in the text. Running DIAC$^+$ on TT provided us with an evaluation of the upper-limit of the system's accuracy (perfect tagging).

For a more realistic setting we further removed from TT the associated tags getting a raw tokenized text (RT) on which we applied the processing chain (tagging with the reduced tag-set, mapping the C-tags to MSD and DIAC$^+$). In both of these experiments DIAC$^+$ was used without any user interaction (that is with the C-words automatically dealt with).

The results of these evaluations are synthesized in Table 2. Unlike the statistics in Table 1, here, no tokens were removed from the evaluation. The *correct surface forms* differences in the two experiments (1,28%) can be ascribed entirely to the tagging errors, but as mentioned before not all the tagging errors generate diacritics restoration errors. A significant part of the incorrect surface forms were C-words (321), which should have received the user attention and choice. However, based on the log file they could be easily corrected. The rest of the incorrect surface forms resulted from tagging errors. Some of these tagging errors in Romanian are very difficult to solve in a limited context. Most of them refer to the tense value attribute (present and imperfect tenses) of verbs in the first class conjugation, the infinitive form of which ends in "a". Their resolution would require a post-tagging processing with an inspection of the neighboring clauses and an analysis of sequence-of-tenses (hoping that the neighboring verbs are not in the same conjugation class).

Table 2. DIAC$^+$ accuracy evaluation

Text	Tagged text (TT)	Raw text (RT)
Tokens	117909	117909
Correct surface forms	116812 (99.06%)	115300 (97.79%)
Incorrect surface forms	1097 (0,94%)	2609 (2,21)
C-words	361	361

6. Implementation

The DIAC$^+$ system is available in two implementations: web service, requiring a licensed access on our linguistic web-services platform for natural language processing and a stand-alone variant intended for local recovering of the diacritics in case of sensitive documents which the author might be reluctant to send via internet.

The web-service version takes the file and autonomously corrects the wordforms considered to lack the diacritics or to contain spelling errors. The C-words are corrected according to the statistical preferences, and a logfile is generated documenting each correction (initial wordform, possible replacements, the actual replacement). Optionally, the logfile can include for each replacement the sentence in which it was operated.

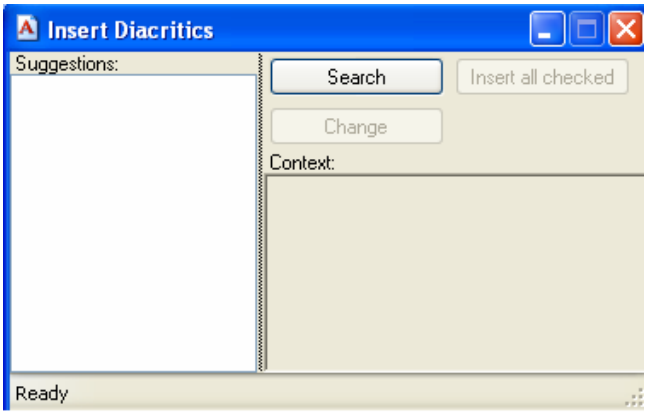

Figure 2. Launching DIAC$^+$

The stand-alone version of the application is embedded into the Microsoft Office suite and complements the MS spell checker. In Figure 2 is shown the snapshot of the DIAC$^+$ interface start-up.

Once an MS-Office document is opened, pressing the *Search* button of the DIAC$^+$ interface will launch the entire processing chain (text extraction, tokenization, tiered-tagging and multi-criteria indexing) discussed in the previous sections. As a result, all the words in the current document, potentially requiring diacritics restoration, will be listed in the left pane (*Suggestions)* of the DIAC$^+$ interface as shown in Figure 3.

Each wordform listed in the *Suggestion* window is preceded by a '+' unfolding button and a "check" box. If the "check" box is checked-out (☑) the system signals that for the respective word it found a unique correction. Selecting a wordform in the "*Suggestions*" window, will bring-up in the "*Context*" window (left window in the DIAC$^+$ interface) the sentence containing the respective occurrence and scroll the document window highlighting the selected wordform (see the wordform "marti" in Figure 3).

Pressing the "*Insert all checked*" will operate the respective corrections and in the "*Suggestions*" pane will remain only the words for which the system could not make an informed decision. These words are preceded by an unchecked box and pressing the '+' unfolding button will show the contextually possible diacritical forms. Each possible solution has a "check" button allowing the user to specify his option.

The system can correct a few typographical errors such as transposed characters, wrong typed characters, or omitted characters.

The MS spell-checker underlines all the unknown words, thus allowing the user to further inspect spelling errors which are out of reach for DIAC$^+$.

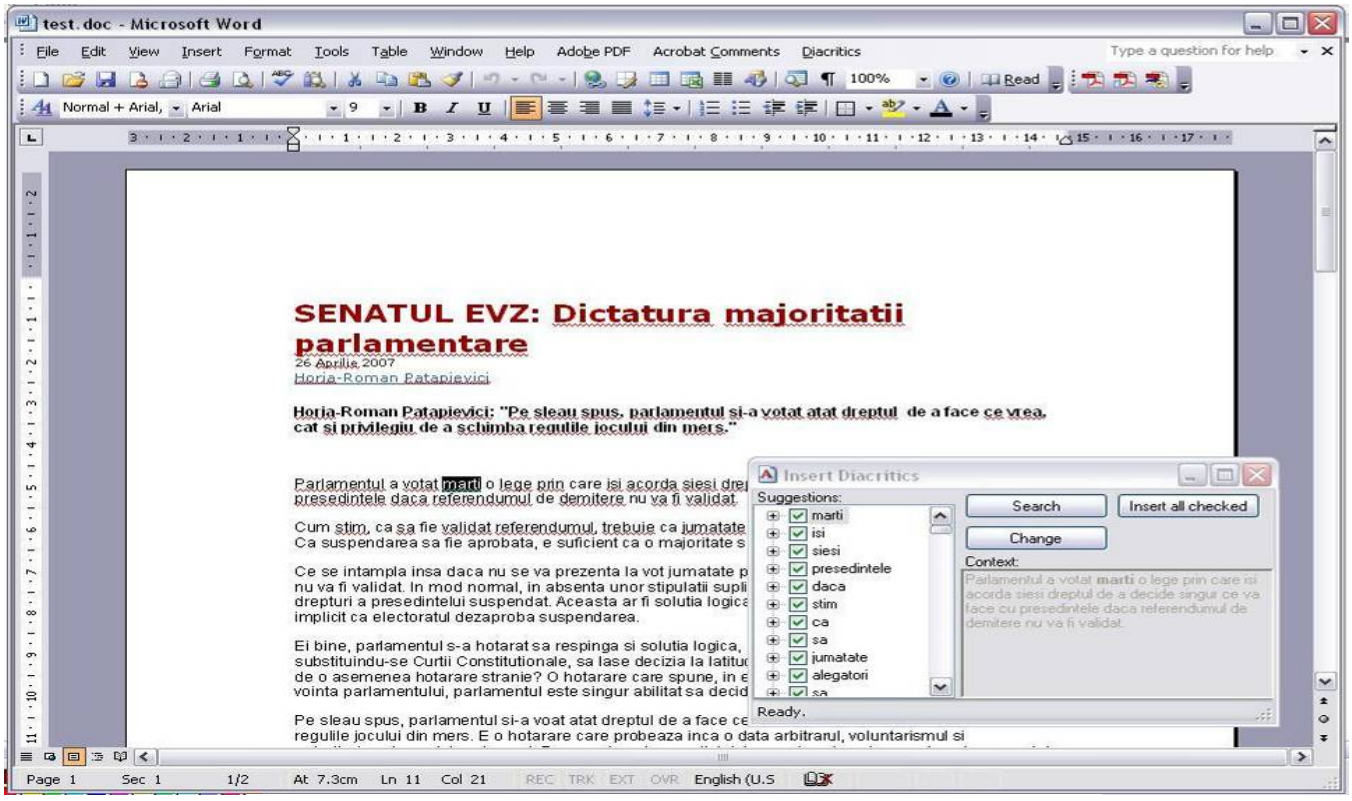

Figure 3. Diacritics insertion in Microsoft Word

7. Conclusions

As compared to our previous version [9], the present DIAC$^+$ implementation includes a spelling corrector, and it is more accurate due to the significant improvements in the underlying language model (the underlying lexicon is almost triple in size) as well as due to the increased accuracy of our tiered tagger. Also, in the previous version we used a combined language model (requiring the text to be re-tagged with each of the available language models and in the end combining the results (see [10] for details). DIAC$^+$ is much faster because it uses a single tagging step, thus avoiding the time overhead of combined language model tagging (at a price of a less than 0.3% decrease of accuracy[2]). Since the coverage of the DIAC$^+$ essentially depends on the statistical underlying dictionary and the language model used by the tiered tagger, the system checks, on a regular basis, our linguistic web-service platform for newer language models and lexicons and updates itself accordingly.

The stand-alone version of DIAC$^+$ is implemented as a DLL and incorporates all the required information and processing tools. Because of this, large MS-Office documents require powerful computers, with more than 1MB RAM memory, and the processing time deteriorates significantly. The web-service version does not have this problem, as the DIAC$^+$ code runs independently of MS-Office programs and thus, it is more appropriate for mass document processing than the DLL-based stand-alone version.

Acknowledgements

This research has been supported by the CEEX ROTEL project, granted by the National Authority for Scientific Research.

References

[1] Berger, A., L., Della Pietra, S., A., Della Pietra, V., J. (1996): A Maximum Entropy Approach to Natural Language Processing in *Computational Linguistics*, vol. 22, no. 1 (pp. 39-72), March 1996

[2] Bèze, M., Mérialdo, B., Rozeron, B., Serouault, A., M. (1994): Accentuation automatique de texte par des méthodes probabilistes. *Technique et sciances informatiques*, 13(6):797-815

[3] Ceauşu, Al. (2006): Maximum Entropy Tiered Tagging, Janneke Huitink & Sophia Katrenko (editors), *Proceedings of the Eleventh ESSLLI Student Session*, ESSLLI 2006, pp. 173-179

[4] Erjavec, T. and Monachini, M. (Eds.) (1997): Specifications and Notation for Lexicon Encoding. Deliverable D1.1 F. Multext-East Project COP-106. http://nl.ijs.si/ME/CD/docs/mte-d11f/

[5] Mihalcea, R., Năstase, V. A. (2001): An automatic method for diacritics insertion into Romanian texts (O metodă automată pentru inserarea diacriticelor în texte în limba română), Tufiş, D., Filip, Gh. (coord.) Limba română în Societatea Informaţională, Expert, Bucharest, pp. 191-205

[6] Monachini, M. & Calzolari, N. (Eds.) (1996): EAGLES Synopsis and Comparison of Morphosyntactic Phenomena Encoded in Lexicons and Corpora A Common Proposal and Applications to European Languages. EAG---CLWG---MORPHSYN/R August, 1996 (http://www.ilc.pi.cnr.it/EAGLES96/morphsyn/morphsyn.html)

[7] Simard, M. (1998): Automatic Insertion of Accents in French Texts. In Ide & Vuotilainen (eds) Proceedings of the Third Conference on Empirical Methods in Natural Language Processing , Granada, Spain, 27-35

[8] Steinberger, R., Pouliquen, B., Widiger, A., Ignat, C., Erjavec, T., Tufiş, D., Varga D. (2006). The JRC-Acquis: A multilingual aligned parallel corpus with 20+ languages. *Proceedings of the 5th International Conference on Language Resources and Evaluation* (LREC'2006). Genoa, Italy, 24-26 May 2006

[9] Tufiş, D., Chiţu, A. (1999): Automatic Insertion of Diacritics in Romanian Texts. In *Proceedings of the 5th International Workshop on Computational Lexicography COMPLEX*, Pecs, Ungaria, 1999, pp. 185-194

[10] Tufiş, Dan (1999). Tiered Tagging and Combined Classifiers. In F. Jelinek, E. Nth (eds) *Text, Speech and Dialogue, Lecture Notes in Artificial Intelligence*, Springer, pp. 28-33

[11] Tufiş, D. (2000). Using a Large Set of EAGLES-compliant Morpho-Syntactic Descriptors as a Tagset for Probabilistic Tagging, *International Conference on Language Resources and Evaluation LREC'2000*, Athens, 2000, pp. 1105-1112

[12] Dan Tufiş, Liviu Dragomirescu. Tiered Tagging Revisited. In Proceedings of the 4th LREC Conference, Lisabona, 2004, pp. 39-42

[13] Tufiş, D., Barbu, A.M. (2004), MULTEXT-East Morphosyntactic Specifications, Application to Romanian, Version 3, 2004

[14] Yarowsky, D. (1994): A Comparison of Corpus-based Techniques for Restoring Accents in Spanish and French Texts. In *Proceedings of the Second Annual Workshop on Very Large Corpora*, Kyoto, Japan

[2] Recall that not all the tagging errors generate diacritics restoration errors, and therefore the improvement by language models combination at the tagging level is not the same at the level of diacritics restoration level.