Diacritics Restoration: Learning from Letters versus Learning from Words

Rada F. Mihalcea

Southern Methodist University
Computer Science and Engineering Department
Dallas, TX, 75275-0122
rada@seas.smu.edu

Abstract. This paper presents a method for diacritics restoration based on learning mechanisms that act at letter level. This technique is new to our knowledge, and we compare it with the well known techniques for diacritics restoration that learn from words. Our method is particularly useful for languages that lack large electronic dictionaries and where means for generalization beyond words are required. Accuracies of over 99% at letter level are reported.

1 Introduction

Diacritics restoration is the problem of inserting diacritics into a text where they are missing. With the continuously increasing amount of texts available on the Web, tools for automatic insertion of diacritics become an essential component in many important applications such as Information Retrieval, Machine Translation, Corpora Acquisition, construction of Machine Readable Dictionaries, and others. Spelling correction has a direct impact on the processing quality in many of these applications. For instance, in the absence of a tool for diacritics recovery, a search for the Romanian word *peşte*(*fish*) retrieves *peste*(*over*) as well, *paturi* can be wrongly translated as *beds*, where the intended meaning was *blankets* (the translation of *pǎturi*), and so forth.

The problem as such is not very difficult, and previous work has demonstrated that a good dictionary can lead to over 90% accuracy in accent restoration for French and Spanish [9], [11], [5]. The method described by Michael Simard in [9] is an improvement over a similar method proposed by El-Bèze [4]. It relies on Hidden Markov Models and learns from surrounding words for an overall reported accuracy of 99%. Tufiş and Chiţu [10] propose a similar approach for diacritics insertion in Romanian texts. Yarowsky gives in [11] a comprehensive overview of accent restoration techniques. Most of the algorithms he presents rely on dictionaries and surrounding words in deciding whether to select a form or another for a given ambiguous word. He mentions, in addition to the baseline constituted by the dictionary based approach, N-gram taggers, Bayesian classifiers and decision lists, all of them relying on contexts, and eventually on additional morphological and syntactic information. A different approach is proposed by Nagy et. al in [7], where strings extracted from texts are used to derive statistics, with high precision reported on French texts. Their work is similar with the approach proposed in [1], where trigram similarity measures are employed for automatic spelling correction.

The majority of studies performed so far in this field have addressed well known and widely spread languages such as French or Spanish, and very few studies have emphasized less popular languages like Czech, Slovene, Turkish or other languages that employ diacritics in their spelling. Table 1 [1] lists the diacritics encountered in European languages. As seen in the table, a large number of languages face the problem of diacritics restoration. From the entire set of 36 languages listed in the table, English seems to be the most "lucky" one from this point of view, as it is the only one with no diacritics. However, because of this distinction its semantic ambiguity is higher than the average language[2].

Language	Diacritics	Language	Diacritics
Albanian	ç ë	Italian	à é è í ì ï ó ò ú ù
Basque	ñ ü	Lower Sorbian	ć č ě ł ń ŕ ś š ź ž
Breton	â ê ñ ù ü	Maltese	ċ ġ ħ ż
Catalan	à ç è é í ï l· ò ó ú ü	Norwegian	å æ ø
Czech	á č d' é í ň ó ř š t' ý ž	Polish	ą ć ę ł ń ó ś ź ż
Danish	å æ ø	Portuguese	â ã ç ê ó ô õ ü
Dutch	á à â ä é è ê ë i í ì î ï ó ò ô ö ú ù û ü	Romanian	â ă î ş ţ
English	none	Sami	á ï č d- ń ņ š t- ž
Estonian	ä č õ ö š ü ž	Serbo-Croatian	ć č d- š ž
Faroese	á æ d- í ó øú ý	Slovak	á ä č d' é í Ĺ ñ ó ô ŕ š t' ú ý ž
Finnish	ä å ö š ž	Slovene	č š ž
French	à â æ ç è é ê ë î ï ô œ ù û ÿ	Spanish	á é í ó ú ü ñ
Gaelic	á é í ó ú	Swedish	ä å ö
German	ä ö ü ß	Turkish	ç ğ i ı ö ş ü
Hungarian	á ó ö ő ú ü ű	Upper Sorbian	ć č ě ł ń ó ř š ž
Icelandic	á æ ∂ é í ó ö ú ý ♭	Welsh	â ê î ô û ŵ ŷ

Table 1. Diacritics in European languages with Latin based alphabets.

We have started to think about this problem when faced with diacritics restoration in an electronic Romanian dictionary. No context is available in this case, and we deal with the dictionary itself and therefore methods relying on information encoded in a dictionary are not useful for this task. The role of a

[1] The table lists only lower case letters. There is a corresponding upper case diacritic letter for each lower case letter. The information in this table was compiled from lists of diacritics in European languages available at http://www.tiro.com/di_intro.html

[2] Studies performed on bilingual parallel corpora have shown that the vocabulary built from an English text is about half the size of the vocabulary build for the same text written in a different language. Senseval competition [6] has also reported significantly lower precision for English with respect to other languages, in a word sense disambiguation task.

dictionary could be played by an *ad-hoc* vocabulary built from online corpora. Nonetheless, for some languages the availability of online data is quite limited, especially when we place the constraint that the texts should contain diacritics.

It turns out that the applicability of previous methods is limited when:
(1) Electronic dictionaries are not available, or only limited size dictionaries are made public. Moreover, when the dictionary itself lacks diacritics, methods relying on diacritics restoration from dictionaries become useless.
(2) Tools for morphological and/or syntactic analysis, which are considered to be helpful for the problem of diacritics restoration, are inexistent or are not publicly available.
(3) Size of usable corpora containing diacritics is limited. The size of the corpora available on the Web or in other public forms influences the size of the vocabulary that can be built *ad-hoc* out of these texts. Moreover, Web publishers choose in many cases to avoid diacritics, for reasons of simplicity, uniformity or just the lack of means for diacritics encoding.

We propose in this paper a technique for diacritics restoration based on learning performed at letter level, rather than word level. The strongest advantage of this method is that it provides the means for generalization beyond words. The method is particularly useful for languages that lack large electronic dictionaries with diacritics. Well studied and widespread languages such as French and Spanish can benefit as well from this methodology in dealing with unknown words.

We have experimented this algorithm on diacritics restoration in Romanian texts, and a precision of over 99% at letter level was observed. Moreover, this method does not require any preprocessing steps, only a small size corpus of raw text with diacritics. Due to the simplicity of the algorithm, the processing speed is very high, about 20 pages of text per second, measured on a Pentium III running at 500MHz, with 250MB memory.

Specifically, instead of learning rules that apply at word level, such as *"anuncio should change to anunció when it is a verb"*, we are interested in learning rules at letter level, like *"s followed by i and preceded by white space should change to ş"*. This latest type of rules are more general and they have higher applicability when only small dictionaries are available, when many unknown words are encountered in the input text, or when there are no usable tools for morphological or syntactic analysis.

It is obvious that letters constitute the smallest possible level of granularity in language analysis, and therefore have the highest potential for generalization. Instead of having about 150,000 units that are potential candidates for the algorithm (the approximate size of the vocabulary of a language), we have more or less 26 characters that will constitute the entry to the disambiguation mechanism.[3]

[3] The actual numbers depend on the language considered. It was shown, for instance, that about 85% of the French words do not have any spelling that includes diacritics, and hence only about 20,000 words are potentially ambiguous. On the other hand, only 7 letters are ambiguous in French.

2 Experimental setup

The purpose of the experiments reported in this paper is to see whether learning at letter level is possible to the end of solving the problem of diacritics re-insertion. Besides providing an additional method for diacritics restoration, the purpose of doing learning at such a low level is to supply a methodology for languages that have only few lexical and semantic resources and for which diacritics restoration via learning at word level is hard to perform.

2.1 Data

During our experiments, we have considered the Romanian language. First, Romanian is not a widely spread language, and consequently it does not have many publicly available tools for preprocessing, and only small electronic dictionaries are available. Secondly, we had to solve a specific problem that required diacritics restoration. We have an electronic dictionary that we plan to use in further development of tools for Romanian. The size of the dictionary is fairly large, about 75,000 entries, but it has the disadvantage of containing no diacritics. Instead of relying on smaller dictionaries with diacritics, we chose to further study the problem of diacritics restoration and make use of our large dictionary. Furthermore, for the tools we plan to develop we need Romanian corpora, which usually lack diacritics, and once again diacritics restoration is required. Moreover, we have the means of comparison with learning at word level performed on the same language, through the experiments and results reported in [10].

We needed therefore a corpus of Romanian texts with diacritics. To this end, we downloaded articles from "România Literară" [4], which is a Romanian newspaper published weekly, with publications related mostly to literature. The newspaper started to have a version including diacritics beginning with year 2000. The entire collection available online at the date of the download (August 2001) adds up to 2,780 articles.

Next, we converted the HTML files into text files. We have paid particular attention only to characters specific to the Romanian language. Other characters such as ê, ç, etc., have been transformed into their equivalents, since we are interested in Romanian characters, rather than French or other languages. After this step, we were left with a corpus of about 3 million words.

Upper case letters have been converted into lower case. It is worth mentioning the case of the â and î letters in the Romanian language. Practically, the two letters have the same pronunciation but their spelling depends on their position within the word. At the beginning of a word, î should be used, whereas â spelling is employed inside the word. The spelling of this letter has been controversial over the years. A law from the sixties changed the spelling from â to î, with the only exception being the words with the root *Român*. In early nineties the old spelling was reintroduced, and so we ended up having inconsistent texts. It so happens that one can encounter different spellings for the same word. For

[4] Available at http://www.romlit.ro

instance, *cîntec* and *cântec*, both meaning *song*, can be sometimes found in the same source text. The "România Literară" newspaper is still applying the î spelling, with few exceptions (i.e. articles written by invited writers who chose to use â instead of î).

2.2 Learning algorithms

We decided to use an instance based learning algorithm for our diacritics restoration task. The reasons for this decision are twofold. First, it was demonstrated that forgetting exceptions is harmful in Natural Language applications, and instance based learning algorithms are known for their property of taking into consideration every single training example when making a classification decision [2]. Secondly, this type of algorithms are efficient in terms of training and testing time. We have used the Timbl [3] implementation to run our learning experiments.

Additionally, we have performed similar experiments with a decision tree classifier, namely C4.5 [8]. The results obtained were similar with the cases when instance based learning is employed, but C4.5 has the capability of generating expressive rules, which are useful for practical implementations.

As we work at the low level of letters, the target attribute to be learned is constituted by the ambiguous letters. It can be therefore any of the *ambiguous* characters listed in Table 1. For Romanian, for instance, we have four pairs of ambiguous letters: *s* - *ş*, *t* - *ţ*, *a* - *ă* and *i* - *î*. Upper case diacritics are not considered as they have been previously converted to lower case. Due to the fact that the source data we are using applies the î spelling, as mentioned in Section 2.1, we do not have an *a* - *â* ambiguity, instead we have an *i* - *î* ambiguity. This fact does not imply any loss in generality. The conversion between the two spelling modes is very simple, using merely the position of the letter within the word, and thus different spellings do not affect in any way the final outcome of the algorithm.

2.3 Features

The features used in any learning algorithm have tremendous influence over the final accuracy. As mentioned in the introduction, we do not have the possibility of using part of speech taggers or any other morphological or syntactic analyzers. Furthermore, we do not want to rely on surrounding words, because the data we have is limited, and we would therefore encounter many cases of unknown words.

Hence, we decided for very simple features, for the extraction of whom no particular processing is required. We are using surrounding letters, with a special notation assigned to white spaces, commas, dots and colons (these characters may affect the learning process, as they are considered special characters by C4.5 and/or Timbl). This set of features performs surprisingly well in terms of accuracy, as shown later in the paper.

3 Results

For Romanian, there are four pairs of ambiguous letters. As mentioned earlier, we did not want to rely on any tags obtained with pre-processing tools, but simply on the information that can be extracted from raw text. Also, we are interested to find means for generalization, such that limited size corpora can be used to derive rules for diacritics re-insertion. Rather than learning from words, as it was the case with previous approaches, we want to learn rules from letters, as they constitute the smallest language units and enable learning from very small corpora.

For each ambiguous pair of letters, we scan the text and generate all possible examples encountered in the corpus. The attributes in an example are formed by N letters to the left and right of the ambiguous letter, and the target attribute is the ambiguous letter itself. We present below samples of feature vectors that were fed to the learning algorithm for the *s* - *ş* ambiguous pair. CO, DO and SP are the replacement codes we use to denote comma, dot or space.

l , i , n , SP , (, u , b , SP , i , n , s.
e , CO , SP , r , o , - , g , a , r , d , ş.
g , a , r , d , i , t , u , l , CO , SP , s.
e , SP , o , r , a , DO , SP , t , o , t , ş.

The number of examples extracted from the corpus depends on the pair of letters. From the entire set of three million words, we obtained 2,161,556 examples for the ambiguous pair *a* - *ă*, 2,055,147 for the pair *i* - *î*, 1,257,458 examples for *t* - *ţ*, and finally 866,964 examples for the *s* - *ş* pair.

The best accuracy was observed for an window size of ten surrounding letters (i.e. $N = 5$). We have therefore studied in more detail this case, including learning rates for the four pairs. Nevertheless, results are provided for various window sizes for comparative purposes.

Table 2 shows the results obtained for $N = 5$. The precision figures reported in this table are obtained using the instance based learning algorithm. We have performed tests with various sizes for the training set, ranging from 2,000,000 examples to as few as 10 examples, to the end of finding the learning rate and the minimum size of a corpus required for a satisfactory precision. All the experiments are performed with a test set size of 50,000 examples. A 10-fold cross validation scheme was used for more accurate results. The table also shows the baseline, defined here as the precision obtained when the most frequent letter is used out of the two letters found in a pair.

The results shown in Table 2 are plotted in Figure 1. It is interesting to observe that the most important part of the learning process is achieved with the first 10,000 examples. We have measured that about 100,000-250,000 running characters (approx. 25-60 pages of text) are needed to generate 10,000 examples with diacritics, which is a small corpus. From there on, a significant number of examples is required for every single percent of improvement in accuracy. We also show in bold the first precision figure that exceeds the baseline, as an indicative

	Ambiguous pair				
	a-*ă*	*a*-*ă*(2)	*i*-*î*	*s*-*ş*	*t*-*ţ*
Data set size	2,161,556	1,369,517	2,055,147	866,964	1,157,458
Baseline	74.70%	85.90%	88.20%	76.53%	85.81%
Training size	Precision obtained with a test set of 50,000 examples				
2,000,000	95.56%	-	99.69%	-	-
1,000,000	95.10%	99.14%	99.58%	-	98.75%
750,000	94.83%	98.97%	99.53%	99.07%	98.63%
500,000	94.57%	98.79%	99.46%	98.86%	98.40%
250,000	94.00%	98.37%	99.28%	98.87%	98.26%
100,000	93.03%	97.56%	98.96%	98.54%	97.81%
50,000	92.10%	96.86%	98.57%	98.13%	97.40%
25,000	90.99%	95.75%	98.11%	97.58%	96.92%
10,000	88.99%	93.75%	97.31%	96.53%	96.20%
5,000	87.56%	92.76%	96.65%	95.61%	95.10%
4,000	86.91%	91.86%	96.49%	94.99%	94.53%
3,000	86.39%	90.99%	96.19%	94.18%	94.30%
2,000	85.81%	89.93%	95.49%	93.47%	93.56%
1,000	83.49%	**88.36%**	93.78%	92.31%	91.85%
500	80.61%	85.66%	93.07%	90.75%	89.74%
250	77.89%	83.17%	92.75%	87.41%	**87.23%**
100	**74.80%**	84.04%	**91.41%**	82.13%	84.46%
50	72.79%	82.73%	88.05%	86.53%	77.54%
25	72.45%	81.34%	88.15%	**78.26%**	78.52%
10	73.38%	85.90%	88.20%	75.88%	85.81%

Table 2. Results obtained in solving diacritics ambiguity, using an instance based learning algorithm and an window size of ten surrounding letters

of the smallest size of training set where a form of learning is observed. Notice that as few as 1,000 examples are enough to perform *some* learning.

Using the entire set of examples extracted from the corpus, the disambiguation of the *i*-*î* pair is almost 100% correct. For this diacritic letter, we now have one instance wrong out of 300 instances, whereas the baseline implies one instance wrong for every eight instances, therefore a significant improvement.

The worst precision is achieved in the case of *a*-*ă* pair. From a simple error analysis, it turns out that the main reason for this is the fact that many Romanian nouns have their base form ending in *ă*, whereas their articulated form ends in *a*. For instance, *masă* and *masa* are two forms, one articulated and one not, for the same noun *table*. Also, some verbs have two different tenses with the only difference standing in an *a* - *ă* ending letter. The learner is therefore tricked by many identical usages for these letters. A simple solution for this is to avoid in the learning process those examples that contain an *a* or *ă* letter at the end of a word. The results obtained under this simplifying assumption are reported in Table 2 under the heading *a-ă(2)*[5]. As shown in the table, more than four percents are gained in precision with this simple condition (this translates into 87% error reduction).

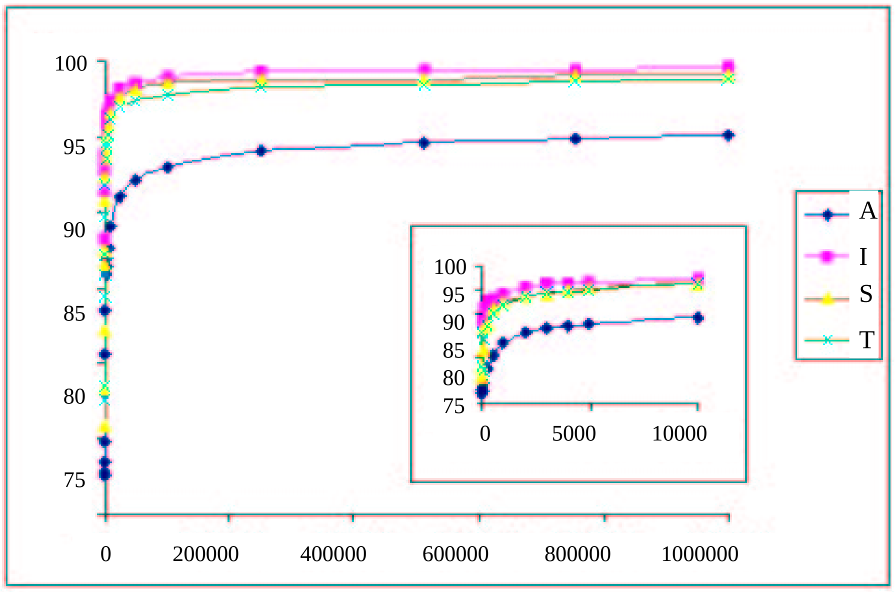

Fig. 1. Learning rates for the four ambiguous diacritics. The chart in the middle represents a zoom of the 0-10,000 range area.

We have also employed C4.5 on the same training data, but no improvements were observed with respect to the results from Table 2. The disadvantage of using C4.5 for this task is that the learning phase is slower than with the Timbl implementation. On the other hand, C4.5 has the capability of generating expressive rules. *"L_1=e and L_2=space then s"*(99.5%), *"L_1=t and L_2=space then s"* (98.7%), *"L_{-4} = p and L_{-1}=v and L_1=t L_2=e then ş"*(95.5%), are examples of such rules, where L_i denotes a surrounding letter at the relative position i with respect to the ambiguous letter. Notice that these rules do not say anything about whether or not the letters belong to one single word. The learning algorithm simply relies on letters, regardless of the word they belong to. Con-

[5] Generality is not affected in by our assumption that ending *a* or *ă* letters are not considered during the learning process. This case of ambiguity can be easily solved by finding words articulation, if any, which is a fairly simple task.

sequently, pseudo-homographs words (as in *peste* and *peşte* - see Section 1) are equally addressed by this method, as the algorithm has the capability of going across words.

3.1 Different window sizes

We have experimented various window sizes to determine the size of the context that would best model our problem. We considered window sizes of two, six, ten, fourteen and eighteen surrounding letters (i.e. $N = 1, 3, 7, 9$). Comparative results are reported in Table 3. These figures should be compared with the uppermost row in Table 2 (the N=5 column in the current table).

Ambiguous	Window size				
pair	N=1	N=3	N=5	N=7	N=9
a-ă(2)	88.63%	98.79%	99.14%	99.10%	99.10%
i-î	94.18%	99.13%	99.69%	99.68%	99.43%
s-ş	88.09%	99.06%	99.07%	99.02%	99.00%
t-ţ	89.45%	98.57%	98.75%	98.67%	98.25%

Table 3. Comparative results for various window sizes

When no context is available, window sizes of N=3 can be used without losing much in precision. Nevertheless, as stated earlier, the best accuracy is attained for a window of ten surrounding letters (N=5).

3.2 Comparison with related work

These results are best compared with the work reported by Tufiş and Chiţu [10], who employed the same language as in our experiments.

According to Tufiş and Chiţu, the task of diacritics recovery in Romanian is harder than with other languages, as Romanian makes more intensive use of diacritics. As reported in their experiments, only about 60% of the Romanian words are diacritics free, compared to the studies reported in [9] which show that about 85% of the French words are spelled with no accents.

The approach presented by Tufiş and Chiţu uses dictionaries, a tokenizer and part of speech tagger, and learning is performed at word level, for an overall performance of 97.4%. We cannot directly compare our results, as both methods and evaluations are fundamentally different. The average precision of 99% we have obtained is measured at letter level, whereas the accuracy they report is determined at word level.

Our methodology overcomes previous approaches in that very high precisions and processing speeds are obtained without any preprocessing tools or dictionaries being required, and therefore this algorithm is applicable to any language, with the only requirement being a relatively small corpus of texts with diacritics.

4 Conclusions

We have presented a method for diacritics restoration based on learning mechanisms that act at letter level. This technique is new to our knowledge, and its strongest advantage stands in its capability of generalization beyond words. No preprocessing steps are required, and no tools or dictionaries are employed. The only requirement is a relatively small corpus of texts with diacritics.

The method is particularly useful for languages that lack large electronic dictionaries and morphological or syntactic tools. Raw texts are fed to the learning mechanism, and accuracies of over 99% at letter level are reported. Moreover, due to its simplicity, processing speeds of about 20 pages of text per second can be attained.

References

1. Angell, R., Freund, G., and Willett, P. Automatic spelling correction using a trigram similarity measure. *Information Processing and Management 19*, 4 (1983), 255–261.
2. Daelemans, W., van den Bosch, A., and Zavrel, J. Forgetting exceptions is harmful in language learning. *Machine Learning 34*, 1-3 (1999), 11–34.
3. Daelemans, W., Zavrel, J., van der Sloot, K., and van den Bosch, A. Timbl: Tilburg memory based learner, version 4.0, reference guide. Tech. rep., University of Antwerp, 2001.
4. El-Bèze, M., Mérialdo, B., Rozeron, B., and Derouault, A. Accentuation automatique des textes par des méthodes probabilistes. *Techniques et sciences informatique 16*, 6 (1994), 797–815.
5. Galicia-Haro, S., Bolshakov, I., and Gelbukh, A. A simple Spanish part of speech tagger for detection and correction of accentuation error. In *Text, Speech and Dialogue - Second International Workshop, TSD'99, September 1999, Proceedings* (Plzen, Czech Republic, 1999), vol. 1692 of *Lecture Notes in Computer Science*, Springer, pp. 219–222.
6. Kilgarriff, A., Ed. *SENSEVAL-2* (to appear).
7. Nagy, G., N., N., and Sabourin, M. Signes diacritiques: perdus et retrouvés. In *Actes du 1er Collque International Francophone sur l'Écrit et le Document CIFED '98* (Queébec, Canada, 1998), pp. 404–412.
8. Quinlan, J. *C4.5: programs for machine learning.* Morgan Kaufman, 1993.
9. Simard, M. Automatic insertion of accents in French text. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing EMNLP-3* (Granada, Spain, 1998).
10. Tufiş, D., and Chiţu, A. Automatic diacritics insertion in Romanian texts. In *Proceedings of the International Conference on Computational Lexicography COMPLEX'99* (Pecs, Hungary, June 1999).
11. Yarowsky, D. *Corpus-based techniques for Restoring accents in Spanish and French Text.* In *Natural Language Processing Using Very Large Corpora.* Kluwer Academics Publisher, 1999, pp. 99–120.

This article was processed using the LaTeX macro package with LLNCS style