ITC 4/46
Journal of Information Technology and Control
Vol. 46 / No. 4 / 2017
pp. 508-520
DOI 10.5755/j01.itc.46.4.18066

Character-Based Machine Learning vs. Language Modeling for Diacritics Restoration

Received 2017/04/27 | Accepted after revision 2017/10/05

http://dx.doi.org/10.5755/j01.itc.46.4.18066

Character-Based Machine Learning vs. Language Modeling for Diacritics Restoration

Jurgita Kapočiūtė-Dzikienė, Andrius Davidsonas, Aušra Vidugirienė

Faculty of Informatics; Vytautas Magnus University; Vileikos Str. 8, LT-44404, Kaunas, Lithuania

Corresponding author: jurgita.kapociute-dzikiene@vdu.lt

In this research we compare two approaches (in particular, character-based machine learning and language modeling) and according to their results offer the best solution for the diacritization problem solving. Parameters of tested approaches (i.e., a huge variety of feature types for the character-based method and a value n for the *n*-gram language modeling method) were tuned to achieve the highest accuracy. Despite the main focus is on the Lithuanian language, we posit that obtained findings can also be applied to other, similar (Latvian or Slavic) languages.

During experiments we measured the performance of used approaches on 10 domains (including normative texts and non-normative Internet comments). The best results reaching ~99.5% and ~98.4% of the accuracy on characters and words, respectively, were achieved with the tri-gram language modeling method. It outperformed the character-based machine learning approach with the tuned feature set by ~1.4% and ~3.8% of the accuracy on characters and words, respectively.

KEYWORDS: Diacritics restoration, character-based machine learning vs. language modeling, the Lithuanian language.

1. Introduction

The increasingly rapid pace of life causes people to save every minute, but does not suppress their will to share news and ideas on social networks, to express opinions on forums, to react to news by posting Internet comments, to tweet or to chat. People want to be quick and responsive, they often write from their tablets and smart-phones. However, all these habits do not pass without a trace in an everyday written

language. The Internet slang phenomenon refers to a presence of invalid words (typing errors, abbreviations, simplified versions of existing words and neologisms), influent and incomplete sentences. In addition, due to ergonomic reasons or writing habits people tend to ignore diacritic marks, by replacing characters with diacritics with their ASCII equivalents. Missing diacritics reduce the readability of texts and cause various ambiguities. For instance, from a short Lithuanian phrase with missing diacritics *sunelis susirgo* (where *sunelis* is a grammatically incorrect word) is not clear if *sūnelis susirgo* (a sonny got sick) or *šunelis susirgo* (a doggy got sick). However, words which maintain grammatical correctness with missing diacritics cause even more difficulties: e.g., *rastas* (found) vs. *rąstas* (a balk) or *raštas* (a script); *karstas* (a coffin) vs. *karštas* (hot), etc. Most of similar ambiguities are easily resolved by humans (who are able to retain contextual and meta-information in mind and/or to link it with the general knowledge) but it is not the case for the computational processing. Due to this reason, the diacritics restoration research area is still important and active.

The contemporary Natural Language Processing research is more focused on the real-world applications, mostly involving the non-normative language texts with missing diacritics. However, the effective solutions for, e.g., Corpora Acquisition, Information Retrieval, Machine Translation and other applications often rely on the external resources (such as ontologies or databases) and tools (morphological analyzers or parsers), which are mostly adjusted for the normative texts. Therefore, a low quality of text degrades the performance of these tools and as a consequence reduces the overall accuracy of all previously-mentioned applications. Hence, an insertion of missing diacritics becomes an important pre-processing component.

Of all the 36 European languages, only English does not have the diacritization problem. Even though some loanwords (e.g., *café*, *pâté*, etc.) exist in English, all of them do not have undiacritized equivalents. The focus of this paper is on the Lithuanian language which orthography has nine letters with diacritics *ą, č, ę, ė, į, š, ų, ū, ž* often replaced with *a, c, e, e, i, s, u, u, z,* respectively. Before offering an effective solution for solving the diacritization problem for the Lithuanian language, we will review the existing approaches.

2. Related Work

Despite the non-usage of diacritics is based on a deliberate writer's decision, this problem is often considered as a special case of spelling correction. In general, the diacritization problem does not seem very complicated: the previous research works done on various languages report over ~90% of the accuracy. Even though the main goal always remains to achieve as high accuracy as possible, the solutions tackling this problem have evolved over time.

The pioneering attempts were based on the naïve dictionary lookup methods: if the matching word could not be found in a dictionary, a system provided a list of possible word-alternatives. More sophisticated knowledge-based approaches, which already check the context of the analyzed word, require syntax analysis and word-sense disambiguation. The seminal work in this field was done by Yarowsky [31], who formulated the diacritization problem as the disambiguation problem and solved it for the Spanish and French languages. The offered decision approach analyzed a context around the target word relying on n-gram part-of-speech tags, morphological analysis and word classes for Spanish, but not on linguistic or lexical resources for French (where a classifier was trained on raw word associations solely). Even for the most difficult ambiguities, the accuracy exceeded ~90%. A very similar method based on the automatic construction of the lemmatizer with lemmatization patterns from the full form lemma dictionary was applied on the Czech language [14]. The achieved diacritics restoration results were similar to the lemmatization results on the Prague Dependency Treebank.

Later, the main interest of the diacritization problem solving shifted towards statistical approaches, which can be grouped into two main categories: i.e., *character-based* and *word-based*.

The main advantage of the character-based approaches is that they are rather fast, simple, and easy to implement. Moreover, they are considered as language-independent, because they do not require any additional language resources apart from the pure text. The ambiguous characters (letters with diacritics and their ASCII equivalents) are treated as the training instances: the original form with diacritics (the so-called base form) becomes a class and the context around the target character (preceding or suc-

ceeding characters, words or their parts) plays a role of features. Afterwards, all these instances are used to train a classifier. The method of this type was applied and evaluated on the newspaper texts in four languages: Czech, Hungarian, Romanian, and Polish [20-21]. The instance-based approach of the TiMBL (Tilburg Memory-Based Learner) implementation was used as a classifier. The best accuracy was achieved with a window size equal to ten (having five preceding and five succeeding characters around the target one). A similar approach is offered for the Maori language [7]. It employs the naïve Bayes classifier and a rich set of features, containing character and word n-grams (window before/after the target character) both simple and compound. The experiments carried out on the corpus of short stories, Bible verses, dictionary definitions and conversational texts resulted in over ~99% of accuracy. The paper in [9] describes experiments based on (1) the machine learning (ML) and (2) the technique combining the ML with the lexicon lookup for (1) unrecognized and (2) recognized words, respectively. This method was able to restore diacritics on the basis of the local character context (using a sliding window of eleven characters, including the target one). The authors trained the memory-based TiMBL classifier and applied their method on 14 languages, including five Indo-European ones. Another language-independent approach is based on a sequence classification and uses a recurrent Neural Network with memory layers to restore diacritics for Arabic [5]. During training, each text sequence is fed into the network to create a prediction for each character. The input layer stacks preceding and succeeding character vectors, thus enabling the model to also learn contextual information. The experiments performed on the Arabic Treebank prove that the offered technique is competitive to the other state-of-the-art methods that have access to external resources.

Unlike character-based, the word-based systems are typically language-dependent and knowledge-intensive, i.e., they rely on dictionaries and probabilistic language models, which indeed are an integral part of the language. The ambiguities in the text are usually resolved by using the n-gram language model which assigns the probability to some word (or phrase) depending on the previous words, corrected with the language modeling. However, the creation of such exhaustive and robust models containing probability distributions over sequences of words and reflecting the language it represents requires huge amounts of grammatically correct text and still remains a risk that some word may not be encountered. The method of this type was successfully applied on the Romanian language [23]. The authors describe the methodology which allows identifying correct and reliable text sections (with regard of diacritics use) in the corpus. If some text produces a ratio below the determined threshold, the text is considered unreliable (i.e., having a lack of diacritics). This method was developed for the automatic speech recognition task and demonstrated the ~20% reduction of the out-of-vocabulary words. Another method applied on the Romanian language and Internet-extracted text corpora is based on a Viterbi algorithm, which allows selecting an optimal state sequence from a variable number of possible options for each word in a sentence [10]. One more statistical language modeling method offered for the Romanian language builds two high-level structures, an n-gram language model and a probabilistic map linking non-diacritical words to their diacritical forms [8]. The ambiguities in the mapping are resolved by finding the sequence with the highest probability. The authors also tested n of n-grams from the interval [2, 5] and achieved the best results with n=3. A solution for the Spanish language combines bigrams of the target word with preceding or succeeding words, backed-off with unigrams of the target words [4]. Another method described in [28] was applied on the French texts. It uses a stochastic Hidden Markov Model (HMM) as the language model (in which non-diacritized words are considered as observations and their possible diacritization alternatives are hidden states that produce these observations) which assigns the scores to any sequence of words. Afterwards, each possible combination is examined to find the most probable one. The dictionary with the bi-gram language model (using Witten-Bell smoothing) method (out of two more tested methods, precisely, (1) dictionary-based and (2) dictionary-based with unsmoothed bi-gram language model) was the most accurate on the Croatian newspaper articles and forum posts [25]. The dynamic programming method aims to assign diacritics to the Arabic texts, collected from the Islamic religious heritage books [13]. The possible sequences with diacritics are assigned with scores using the n-gram language model and afterwards the dynamic algorithm searches the most likely sequence. The researchers claim that the higher order n-grams (in particular, trigrams, tetra-grams) can lead to a higher word accuracy rate.

There are some methods which do not completely fit into the frame of previously described word-based approaches. A good example of such approach is a system proposed for the Algerian Arabic dialectal texts [12]. The diacritization process is performed via the statistical machine translation from the undiacritized source text to the diacritized target text. The system is phrase-based, uses GIZA++ toolkit (developed for the Statistical Machine Translation) for the word alignment, and is trained on books. Even though the experiments were performed on a small parallel corpus, the authors claim that the system achieved acceptable results. Another similar approach, also based on the statistical machine translation but implemented for the Hungarian texts, is described in [22]. It incorporates a morphological analyzer and is able to achieve the accuracy equal to ~99%.

There also are examples of hybrid approaches, combining the advantages of several techniques. The proposed hybrid method restoring diacritics in Turkish social media texts uses discriminative sequence classifier (i.e., Conditional Random Fields – CRFs) in the first stage and a language validator to select the possible options in the second [2]. Another statistical approach integrating multiple knowledge sources (stemming, part-of-speech tagging, pronunciation lexicons, and word bigrams) revealed that character-level trigrams achieve the highest accuracy by applying all these language sources on the Urdu corpus [3]. The system applied on the Romanian journalism and juridical texts (described in [29]) uses lexicon lookup and HMM language model for known words and character-based method (similar to described by [20]) for the out-of-vocabulary words.

The diacritization problem solved by using only one method under very different experimental conditions for the various languages does not give the answer which method is actually the top-notch. The comparative research on Wikipedia, general web and Twitter texts using the lexicon approach (applying the most frequent token-level translation) and the corpus-based approach (combining information about the probability of a translation and the probability of observing a translation in the given context via a simple log-linear model) proves the superiority of the corpus-based approach for all three tested languages, in particular, Croatian, Serbian, and Slovene [17]. The research described in [27] compares two diacritization methods on the Arabic Treebank and AppTek Data texts. The first method is based on the machine translation which post-edits rule-based diacritization system. The second one explores the traditional view of sequence labeling problem using the CRFs classifier and a rich set of features (lexical, morphological and character). The experiments claim that machine translation approach performs better compared to the sequence labeling. The authors in [1] compare four techniques on the Hungarian Web corpus and Facebook comments, and demonstrate that the character-based technique is robust enough to outperform the dictionary-lookup, the dictionary-based, and the dictionary with character bigram based methods. Moreover, it requires only a few characters of context, thus can be applied to very short text segments, as, e.g., tweets. The comprehensive comparative research described in [26] compares seven algorithms based on two lexicon lookup, four character-level statistical models and the combination of both techniques (i.e., lexicon lookup and the best of character-level statistical models for words that do or do not appear in the lexicon, respectively). These algorithms were applied on the texts harvested from the Internet of ~100 African and other languages, including Lithuanian. The offered tool uses the naïve Bayes classifier for the word-level and the character-level modeling; all models are trained on the lowercased letters and use smoothing.

As demonstrated by Scannell [26], the best results on the Lithuanian texts are achieved with the lexicon lookup method. Scannell's paper is influential because it reports the only diacritics restoration results for the Lithuanian language. On the other hand, the tested methods are not specifically adjusted to the language characteristics; the testing set is small and unvaried. Consequently, the contribution of our work is to test and compare different techniques and according to the obtained results to offer the best possible diacritics restoration technique for the Lithuanian language, taking into account language-specific characteristics.

3. A Character-Based ML Method

A formal description of the task

The mathematical formulation of the diacritization problem that we are attempting to solve with the

character-based ML method is presented below.

Let d_i be any character in the lowercased text. Let *set* = {*a, ą, c, č, e, ę, ė, i, į, s, š, u, ų, ū, z, ž*} denote a set of ambiguous characters in the Lithuanian language.

The (undiacritized) target character d_i has its correct base form (so-called class c_i), where class $c_i \in set$ or $c_i \notin set$ (i.e., c_i = *). Thus, with $|set| + 1 = 16$ classes in total we get the multi-class classification task.

Let f be a fixed set of features, describing each d_i later used in the ML task.

Let $\gamma : \forall d_i \rightarrow c_i$ denote a function mapping each d_i to its class (correct character base form) c_i. The aim of this work is to offer a method, which would return as close approximation of γ as possible.

Features

Let d_0 and w_0 denote a target character and a target word (such that $d_0 \in w_0$), respectively. Thus, the training instance of that target character (within a sequence of other characters) is described with a set of the following features:

- d_n – a single character without diacritics. In our experiments we used n from the interval [-1,1]. Thus, d_0, d_{-1} and d_1 points to the target character, to the character preceding d_0, to the character succeeding d_0, respectively.
- w_m – a single word without diacritics. In our experiments we tested $m \in$ [-3,3].
- $d_{[n1,n2]}$ – a sequence of characters without diacritics, which precede ($n1 \in$ [-6,-1]) or succeed ($n2 \in$ [1,6]) the target character d_0 in the word w_0.
- $d(w_{m)[n1,\ n2]}$ – a sequence of characters without diacritics extracted from the end ($n1 \in$ [-3,-1] and $n2 = 0$) of w_{-1} or the beginning ($n1 = 0$ and $n2 \in$ [3,4]) of w_1.

Classification

During the classification stage, the labeled (with known classes) training instances are given to a classifier. The classifier generates a model, which can afterwards be used for the diacritics restoration.

In our experiments, we used the CRFs classifier, introduced by Lafferty et al. [16].[1] This discriminative-based approach was selected due to a number of reasons. Firstly, the created model outputs confidence measures indicating how certain it is about the predicted labels. Moreover, CRFs are not based on an independent assumption (stating that features do not depend, therefore do not affect each other) and allow providing neighboring characters as features into the system. Furthermore, this method has been successfully tested in similar diacritization tasks, e.g., on Turkish [2].

In our experiments, we tuned only the feature set (by adding new features and testing their influence on the overall accuracy) and for each instance selected the predicted label with the highest probability.

1 The implementation of the CRFs method was downloaded from https://taku910.github.io/crfpp/.

4. The Language Modeling-Based Method

The formal description of the task

The mathematical formulation of the diacritization problem that we are attempting to solve with the n-gram language modeling-based method is presented below.

Let w_i denote a word and m denote a length of a word sequence.

The language model assigns a probability P to any sequence of words appearing in the text $P(w_1, w_2, \ldots, w_m)$.

Having a unigram model with n=1, the probability $P(w_i)$ of some w_i depends on the word itself and is calculated as $P(w_i) = count(w_i) / count(all)$. Thus, the sum of all probabilities in the corpus must be equal to 1.

Having the higher order n-gram model with $n > 1$ the probability is calculated as:

$$P(w_1, w_2, \ldots, w_m) \approx \prod_{i=1}^{m} P(w_i \mid w_{i-(n-1)}, \ldots, w_{i-1}),$$

in which the conditional probability is calculated as:

$$P(w_i \mid w_{i-(n-1)}, \ldots, w_{i-1}) = \frac{count(w_{i-(n-1)}, \ldots, w_{i-1}, w_i)}{count(w_{i-(n-1)}, \ldots, w_{i-1})},$$

Language models

During the diacritics restoration process, a group of possible word-candidates with diacritics are gener-

ated for each undiacritized target word. For instance, the group of candidates for *kasti* would consist of two grammatically correct words: *kasti* (to dig), *kąsti* (to bite) and six grammatically incorrect words: *kašti, kąšti, kastį, kąstį, kaštį, kąštį*.

When multiple grammatically correct word-candidates are possible (e.g., *kasti* (to dig), *kąsti* (to bite)), the ambiguity is resolved considering their probabilities obtained from the training data. Under the unigram language model, the word-candidate with the higher probability is considered as the right choice and replaces the undiacritized target word. When using the higher order n-gram language model, the replacement of the target word depends on the previously replaced words. The overall probability of the whole n-gram has to be the highest [6].

The search is performed by using the back-off search strategy, i.e., if some n-gram language model fails, the method continues with the lower order of n-gram, i.e., with n-1, n-2, etc., and if necessary, n=1. In our experiments, we used our own language modeling method implementation.

5. The Data

In our experiments, we used two datasets: (1) *DevelopmentSet* (see in Table 1) for tuning the feature set of the character-based ML method; (2) *TestSet* (see in Table 2) for testing both character-based ML and language modeling methods. The captions *Numb. of words*, *Numb. (%) of amb. words*, *Numb. of characters*, *Numb. (%) of amb. char.* in both tables denote a number of words, a number (percentage) of words having ambiguous characters (*a*, which can remain *a* or be replaced to *ą*; *c* which can remain *c* or be replaced to *č*, etc.), a number of characters, a number of ambiguous characters (percentage), respectively.

DevelopmentSet (used for training and parameter tuning) was composed of the normative texts taken from Vytautas Magnus University corpus [18]. It consists of 50 thousand words from each of nine domains (republican newspapers, local newspapers, popular periodicals, specialized periodicals, fiction, non-fiction, state documents, philosophical literature translations, and memoirs).

Table 1
Statistics about the dataset *DevelopmentSet*

Numb. of words	Numb. (%) of amb. words	Numb. of characters	Numb. (%) of amb. char.
450,000	423,870 (94.2%)	2,747,451	1,427,260 (51.9%)

TestSet was also composed of the normative texts taken from Vytautas Magnus University corpus (none of the texts in *DevelopmentSet* and *TestSet* overlapped). However, in addition to the normative texts (which can be considered as artificial for the diacritization problem solving) we added one more domain, i.e., the non-normative Internet comments. These comments were harvested from under the articles of two subjects "In Lithuania" and "Abroad" in the news portal *www.delfi.lt*. These texts were manually corrected by a human expert to become suitable for the diacritics restoration testing. It is hard to distinguish problems related to some specific diacritization marks (either all or none of the diacritization marks were used in the text). It is important to note that the correction process involved only diacritization problems (not touching any other mistakes) (see *Internet comments* domain in Table 2). Moreover, diacritized characters and words found in the Internet comments were not used for testing and evaluation, therefore, they could not affect the overall diacritics restoration results.

In the language modeling experiments, we used four text corpora, containing only normative language texts:

- parliamentary transcripts. The corpus *STENOGRAMOS_INDV* of 23,908,302 running tokens.
- fiction texts. The corpus *GROŽINĖ_INDV* of 9,762,077 running tokens.[2]
- news articles. The corpus, described in [15] of 2,251,725 running tokens.
- texts from nine different domains. The corpus, presented in Table 2 of 450,000 running tokens.

All these corpora were used to create frequency word (or their n-gram) lists (ignoring punctuation and digits) (see Table 3) afterwards used in language modeling tasks.

2 Both corpora, i.e., *STENOGRAMOS_INDV* and *GROŽINĖ_INDV* were downloaded from http://dangus.vdu.lt/~jkd/eng/?page_id=16.

Table 2
Statistics about the dataset *TestSet*

Domain	Numb. of words	Numb. (%) of amb. words	Numb. of characters	Numb. (%) of amb. char.
republican newspapers	20,000	18,806 (94.03%)	122,886	63,108 (51.4%)
local newspapers	20,000	18,969 (94.8%)	124,001	64,928 (52.4%)
popular periodicals	20,000	17,966 (89%)	116,437	60,485 (51.9%)
specialized periodicals	3,355	3,225 (96.1%)	20,053	10,492 (52.3%)
fiction	20,000	19,365 (96.8%)	108,473	58,142 (53.6%)
non-fiction	20,000	19,404 (97.0%)	124,982	64,876 (51.9%)
state documents	16,307	14,198 (87.1%)	105,858	51,712 (48.9%)
philosophical lit. transl.	5,938	5,749 (96.8%)	33,067	17,371 (52.5%)
memoirs	20,000	18,691 (93.5%)	116,850	61,292 (52.5%)
Internet comments	20,000	19,120 (95.6%)	113,203	59,708 (52.7%)
Overall:	165,600	155,493 (93.9%)	985,810	512,114 (51.9%)

Table 3
The created frequency lists

Of n words	Numb. of elements	The highest frequency
n=1	705,185	1,285,817
n=2	12,173,949	60,864
n=3	25,255,584	18,059

6. Experiments and Results

For diacritization, we lowercased all the words in the texts and performed the following sets of experiments:

Experiment set No. 1: the feature set tuning for the character-based ML method (presented in the *Character-based ML method* section). The tuning of the feature set for the CRFs model was performed iteratively in the greedy manner, only the feature which gave the highest accuracy during particular iteration was added to the feature template for the later iterations. The development process was initiated with the empty feature template and continued until any added feature (described in the *Features* subsection of the *Character-based ML method* section) still allowed improving the accuracy. During each iteration, we performed an exhaustive search in the set of the remaining features (not yet added to the feature template) excluding those features which gave degradation in performance in any previous iteration. The experiments were carried out on *DevelopmentSet* (described in *The data* section) using stratified tenfold cross-validation.

Experiment set No. 2: the evaluation of n for the n-gram language modeling method (presented in the *Language modeling-based method* section). If the method with the back-off strategy failed even with n=1 (in case of, e.g., the out-of-vocabulary word), it returned the word without diacritics. The method was tested on each of domains in *TestSet* and on the entire dataset.

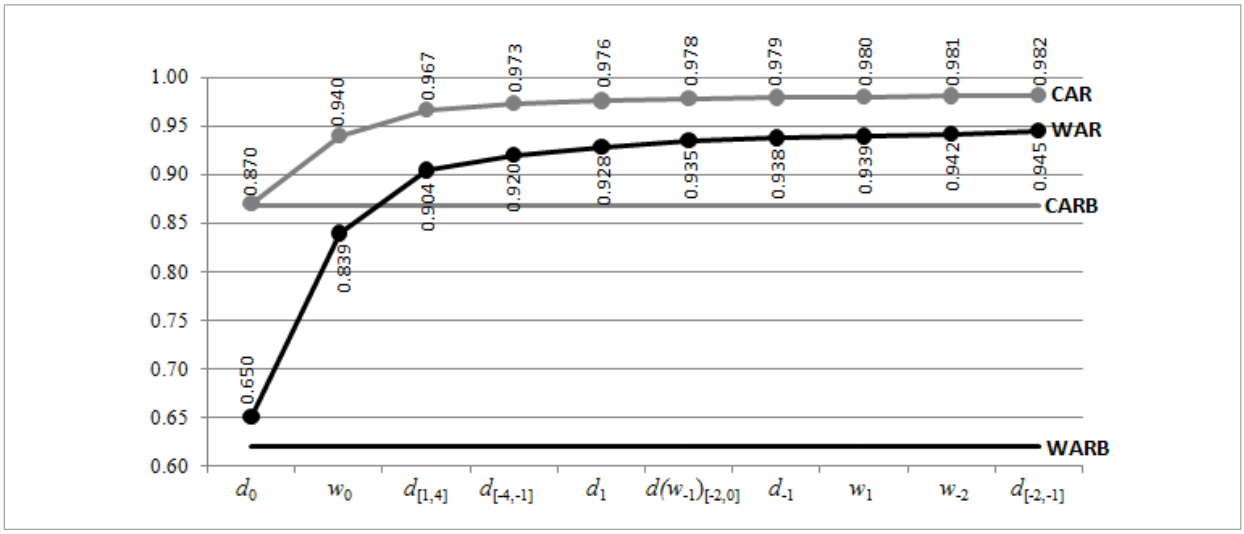
Experiment set No. 3: the evaluation of the character-based ML method. After the most accurate feature template was determined (during the experiment set No. 1), we trained CRFs classifier on the entire *DevelopmentSet* to generate the final model. This model was evaluated on the separate domains in *TestSet* and on the entire *TestSet*. The obtained results were compared with the language modeling method.

During all the previously described experiments, we evaluated *Character Accuracy Rates* (*CAR*) Eq. (1) and *Word Accuracy Rates* (*WAR*) Eq. (2):

$$CAR = \frac{correctly\_diacritized\_characters}{all\_characters} \quad (1)$$

Figure 1

The accuracy (y axis) achieved with different features (x axis). The gray and black curves represent *CAR* and *WAR* values, respectively. The gray and black straight lines – *CARB* and *WARB* equal to 0.868 and 0.621, respectively



$$WAR = \frac{correctly\_diacritized\_words}{all\_words} \quad (2)$$

It is important to mention that while evaluating *CAR*, we encountered only ambiguous characters (*a, ą, c, č, e, ę, ė, i, į, s, š, u, ų, ū, z, ž*).

We also calculated the baselines to ensure that the achieved results are effective and reasonable (i.e., exceed the determined baselines). The baselines *CARB* and *WARB* denote *CAR* and *WAR* values obtained on the undiacritized *TestSet*, respectively.

The results, obtained during the experiment set No. 1 and No. 2 are presented in Figure 1 and Table 4, respectively. It is important to note that despite the features presented in Figure 1, four more (in particular, $d_{[1,3]}$, $d_{[-2,-1]}$, $d_{[-5,-1]}$, $d_{[5,1]}$) gave the marginal increase in the accuracy. As this increase was not statistically significant, we have not added these features in the figure, but still have preserved in the feature template for the experiment set No. 3. In addition, the statistical significance was evaluated by using the McNemar [19] test (with the significance level of 95%), meaning that differences were considered statistically significant if calculated probability density function p was lower than $\alpha = 0.05$.

The language modeling method achieved marginally the best results with n=3. Despite the increase of n=3 on n=2 being statistically insignificant on most of the domains, it demonstrated the statistically significant increase on the entire dataset *TestSet*. The aim of this research was to compare the effectiveness of character-based ML and language modeling methods (with their best parameters, determined feature set and n=3). Figures 2 and 3 report *CAR* and *WAR* values obtained with the character-based ML method (in the experiment set No. 2 and language modeling method (in the experiment set No. 3), respectively.[3]

7. Discussion

The Lithuanian texts consist of ~6.9% diacritized letters [11], moreover, ~39.1% of Lithuanian word forms contain at least one diacritical letter [30]. However, the problem is more complicated, diacritized letters have their ASCII equivalents, therefore, the total number of ambiguous characters and words in undiacritized texts jumps up to ~52% and ~94%, respectively (see Tables 1 and 2). Despite it, zooming into Table 4 and Figures 1-3 allows us to make the most

3 All real numbers (presented on each column in Figure 2 and Figure 3) represent exact *CAR* and *WAR* values (not their increase over *CARB* and *WARB*, respectively).

Table 4

The accuracy values using different *n* with the language modeling method. The upper/lower values in each cell represent *CAR*/*WAR* values, respectively. The underlined values indicate statistically insignificant increase on the results in the previous column

Domain	CAR & WAR with n=1	with *n*=2 (incr. on *n*=1)	with *n*=3 (incr. on *n*=2)
republican newspapers	0.9860 0.9565	0.0104 0.0321	0.0003 0.0010
local newspapers	0.9851 0.9523	0.0113 0.0359	0.0003 0.0009
popular periodicals	0.9860 0.9583	0.0111 0.0330	0.0002 0.0007
specialized periodicals	0.9857 0.9553	0.0099 0.0310	0.0010 0.0033
fiction	0.9858 0.9597	0.0113 0.0320	0.0024 0.0070
non-fiction	0.9839 0.9480	0.0123 0.0396	0.0002 0.0007
state documents	0.9899 0.9679	0.0060 0.0189	0.0007 0.0022
philosophical literature translations	0.9777 0.9360	0.0166 0.0477	0.0010 0.0030
memoirs	0.9903 0.9705	0.0054 0.0165	0.0015 0.0044
Internet comments	0.9723 0.9238	0.0026 0.0077	0.0001 0.0002
Overall:	0.9846 0.9537	0.0092 0.0280	0.0007 0.0022

important statement – the offered methods are robust enough, because all the obtained results exceed their baselines (character and word accuracy rates calculated on the undiacritized texts).

As can be seen from Figure 1, the best features (making the largest impact on the results) for the character-based ML method are the following: a character without diacritics, a word without diacritics, a context of four characters around the target character, etc. Moreover, all these options in the feature template are logically explained. A character without diacritics allows determining the particular set of alternatives (e.g., *c* might remain as *c* or be replaced with *č*, but never with *ą*). The word (with or without diacritics) itself helps the method to "memorize" incoming diacritics. The close context around the target character is more important compared to the information in the preceding or succeeding words around the target one.

Even in diacritized texts, the rate of ambiguous morphological forms reaches ~47% for the Lithuanian language [24], therefore, in the undiacritized texts this problem is even more evident. Due to this reason, the unigram language modeling method is not the best option, because it naïvely chooses the most common word-candidate not considering any context around the target word. Intuitively, it seems that the larger *n* should assure the higher accuracy (especially taking into account that the language modeling method uses the back-off strategy). In fact, it assures (see Table 4), but the increase of *n*=3 over *n*=2 on most of the domains is not statistically significant. We assume that the higher order n-grams become not very helpful due to the relatively free word order in Lithuanian sentences: the variety becomes huge (e.g., out of ~36.4 million running words there were generated more than ~25.2 million trigrams (see Table 3)), but some occurrences are very rare.

As can be seen from Figures 2 and 3, the language modeling method outperforms character-based ML on all domains and on the entire testing set; moreover, the differences in their accuracies are statistically significant. It allows us to state that the language modeling method is more suitable for the Lithuanian language. The explanation of this phenomenon lies in the nature of each method. The language modeling method has less opportunity to make mistakes, because it predicts the whole word in its context. The character-based ML approach considers one character at a time in the analyzed word, which usually has more than one ambiguous character.

The character-based ML method with CRFs achieved the best results on the memoirs and state documents; the language modeling on fiction and memoirs. However, normative texts are artificial data, where diacritics were simply removed for the testing/training reasons. The weak spot of both methods is rather low accuracy on the real data (i.e., Internet comments), where diacritics were absent (i.e., not inserted by their authors). The detailed error analysis revealed that the character-based ML method fails on the foreign language insertions (which are the most often in the Internet comments, but rare/absent in the memoirs and state documents), because it still tries to "restore" diacritics. Since most of the foreign words (found in the Internet comments, but absent in frequency lists)

Figure 2

CAR values (y axis) achieved on the different domains (x axis). The *gray* and *white* columns (including the black part of *CARB* values) represent values for the *character-based ML* and *language modeling* methods, respectively

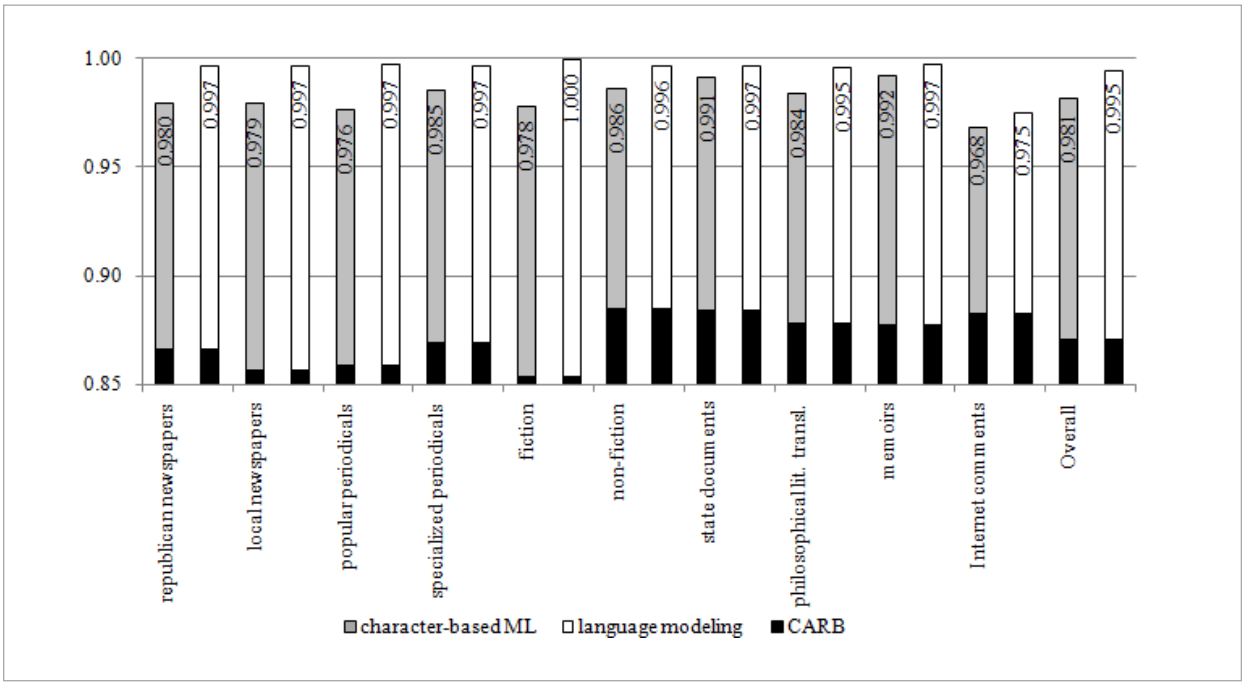
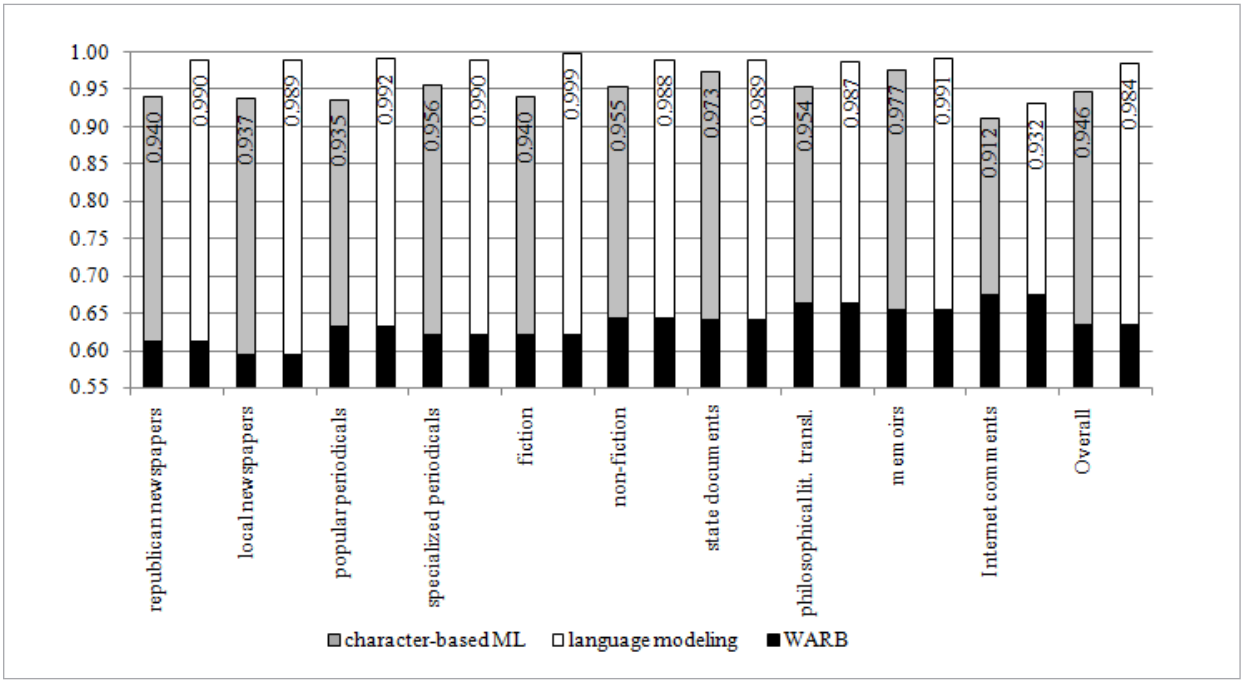


Figure 3

WAR values (y axis) achieved on the different domains (x axis). The *gray* and *white* columns (including the black part of *WARB* values) represent values for the *character-based ML* and *language modeling* methods, respectively



are in English, the language modeling method returns them in the untouched/undiacritized and at the same time corrects form. One more difficulty for both of the methods was unusual abbreviations occurring in the dataset of Internet comments. The character-based ML method was trying to restore diacritics, whereas the language modeling method treated them as out-of-vocabulary words leaving in the undiacritized form, which was not always correct.

Resuming, the drawback of the character-based ML method is the fact that it does not consider the whole words and their context at the same time. The language modeling method suffers from the out-of-vocabulary words problem. Despite the huge number of words or their n-grams in the frequency lists, something still might be missing. The Lithuanian language is fusional (meaning that morphemes in a combination with different affixes denote multiple grammatical, syntactic or semantic word forms), therefore, during the creation of frequency lists it is important not only to find a particular word, but to find it in all its possible forms, which already requires much more diverse training data.

We anticipate that our findings about the diacritization problem solving might be useful not only for the Lithuanian language, but for other languages (as, e.g., Latvian, Slavic languages) sharing similar characteristics as well.

8. Conclusions and Future Work

In this research, we are solving the important diacritization problem for the Lithuanian language, which has never been solved before considering the Lithuanian language characteristics.

In this paper we experimentally compared two approaches, in particular, character-based ML and language modeling, and proved the superiority of the language modeling method. The language modeling method outperformed character-based ML by ~1.4% and ~3.8%, achieving ~99.5% and ~98.4% accuracy on characters and words, respectively.

The worst results were obtained on the real data (i.e., Internet comments); therefore, in the future research we are planning to focus on similar types of non-normative texts. Probably the best solution could be the two-stage hybrid approach – the out-of-vocabulary words (not recognized with the language modeling) could be processed at the character level and corrected by using the machine learning approach.

References

1. Ács, J., Halmi, J. Hunaccent: Small Footprint Diacritic Restoration for Social Media. Normalisation and Analysis of Social Media Texts (NormSoMe) Workshop, 2016.
2. Adalı, K., Eryiğit, G. Vowel and Diacritic Restoration for Social Media Texts. The 5th Workshop on Language Analysis for Social Media (LASM) at EACL, 2014, 53–61. https://doi.org/10.3115/v1/W14-1307
3. Ali, A. R., Hussain, S. Automatic Diacritization for Urdu. Proceedings of the Conference on Language and Technology, 2010, 105–111.
4. Atserias, J., Fuentes, M., Nazar, R., Renau, I. Spell Checking in Spanish: the Case of Diacritic Accents. Proceedings of the 8th International Conference on Language Resources and Evaluation (LREC), 2012, 737–742.
5. Belinkov, Y., Glass, J. Arabic Diacritization with Recurrent Neural Networks. Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, 2015, 2281–2285. https://doi.org/10.1007/s10032-015-0242-2
6. Brown, P. F., deSouza, P. V., Mercer, R. L., Pietra, V. J. D., Lai, J. C. Class-Based n-gram Models of Natural Language. Computational Linguistics, 1992, 18, 467–479.
7. Cocks, J., Keegan, T. T. A Word-Based Approach for Diacritic Restoration in Maori. Proceedings of the Australasian Language Technology Association Workshop 2011, 2011, 126–130.
8. Cucu, H., Buzo, A., Besacier, L., Burileanu, C. SMT-Based ASR Domain Adaptation Methods for Under-Resourced Languages: Application to Romanian. Speech Communication, 2014, 56, 195–212. https://doi.org/10.1016/j.specom.2013.05.003
9. De Pauw, G., Wagacha, P. W., de Schryver, G. M. Automatic Diacritic Restoration for Resource-Scarce Languages. Proceedings of Text, Speech and Dialogue, the 10th International Conference, 2007, 4629, 170–179. https://doi.org/10.1007/978-3-540-74628-7_24
10. Dumitrescu, S. D., Boroş, T. A Unified Corpora-Based Approach to Diacritic Restoration and Word Casing. The 7th Language & Technology Conference: Human

Language Technologies as a Challenge for Computer Science and Linguistics, 2013, 299–303.

11. Grigas, G., Juškevičienė, A. Raidžių dažnių lietuvių ir kitose kalbose, vartojančiose lotyniškus rašmenis, analizė [Letter Frequency Analysis of Lithuanian and Other Languages Using the Latin Alphabet]. Santalka. Filologija, Edukologija, 2015, 23, 81–91.
12. Harrat, S., Abbas, M., Meftouh, K., Smaïli, K. Diacritics Restoration for Arabic Dialect Texts. The 14th Annual Conference of the International Speech Communication Association (INTERSPEECH), 2013, 1429–1433. https://doi.org/10.21700/ijcis.2016.119
13. Hifny, Y. Higher Order n-gram Language Models for Arabic Diacritics Restoration. The 12th ESOLE Conference on Language Engineering (ESOLEC'12), 2012, 1–5.
14. Kanis, J., Müller, L. Using Lemmatization Technique for Automatic Diacritics Restoration. SPECOM 2005 proceedings, 2005, 255–258. https://doi.org/10.1007/978-3-540-30120-2_45
15. Kapočiūtė-Dzikienė, J. Krilavičius, T. Topic Classification Problem Solving for Morphologically Complex Languages. The 22nd international conference on Information and Software Technologies (ICIST), 2016, 511–524. https://doi.org/10.1007/978-3-319-46254-7_41
16. Lafferty, J. D., McCallum, A., Pereira, F. C. N. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. Proceedings of the 18th International Conference on Machine Learning (ICML'01), 2001, 282–289.
17. Ljubešić, N., Erjavec, T., Fišer, D. Corpus-Based Diacritic Restoration for South Slavic Languages. Proceedings of the 10th International Conference on Language Resources and Evaluation (LREC 2016), 2016, 3612–3616.
18. Marcinkevičienė, R. Tekstynų lingvistika (teorija ir paktika) [Corpus Linguistics (Theory and Practice)]. Darbai ir dienos, 2000, 24, 7–63.
19. McNemar, Q. M. Note on the Sampling Error of the Difference Between Correlated Proportions or Percentages. Psychometrika, 1947, 12(2), 153–157. https://doi.org/10.1007/BF02295996
20. Mihalcea, R. F. Diacritics Restoration: Learning from Letters versus Learning from Words. Computational Linguistics and Intelligent Text Processing, the 3rd International Conference (CICLing 2002), 2002, 339–348. https://doi.org/10.1007/3-540-45715-1_35
21. Mihalcea, R. F., Nastase, V. A. Letter Level Learning for Language Independent Diacritics Restoration. Proceedings of CoNLL-2002, 2002, 105–111. https://doi.org/10.3115/1118853.1118874
22. Novák, A., Siklósi, B. Automatic Diacritics Restoration for Hungarian. Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2015, 2286–2291. http://www.aclweb.org/anthology/D15-1275
23. Petrica, L., Cucu, H., Buzo, A., Burileanu, C. A Robust Diacritics Restoration System using Unreliable Raw Text Data. The 4th Workshop on Spoken Language Technologies for Under-resourced Languages (SLTU), 2014, 215–220.
24. Rimkutė, E. Morfologinio daugiareikšmiškumo ribojimas kompiuteriniame tekstyne [The Limitation of the Morphological Disambiguation in the Digitalized Corpus]. Ph.D. thesis, Vytautas Magnus University, 2006.
25. Šantić, N., Šnajder, J., Bašić, B. D. Automatic Diacritics Restoration in Croatian Texts. The Future of Information Sciences, Digital Resources and Knowledge Sharing, 2009, 309–318.
26. Scannell, K. P. Statistical Unicodification of African Languages. Language Resources and Evaluation, 2011, 45 (3), 375–386. https://doi.org/10.1007/s10579-011-9150-3
27. Schlippe, T., Nguyen, T., Vogel, S. Diacritization as a Translation Problem and as a Sequence Labeling Problem. The 8th Conference of the Association for Machine Translation in the Americas, 2008.
28. Simard, M. Automatic Restoration of Accents in French Text. Technical Report Co28-1/129-1996E, Industry Canada, Centre for Information Technology Innovation (CITI), 1996.
29. Tufiş, D. Ceauşu, A. DIAC+: a Professional Diacritics Recovering System. Proceedings of the 6th International Conference on Language Resources and Evaluation (LREC'08), 2008, 167–174. http://www.lrec-conf.org/proceedings/lrec2008/pdf/54_paper.pdf
30. Utka, A. Dažninis rašytinės lietuvių kalbos žodynas. [Frequency Dictionary of the Lithuanian Language], 2009. http://donelaitis.vdu.lt/publikacijos/Dazninis_zodynas.pdf
31. Yarowsky, D. Decision Lists for Lexical Ambiguity Resolution: Application to Accent Restoration in Spanish and French. Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics (ACL), 1994, 88–95. https://doi.org/10.3115/981732.981745

Summary / Santrauka

In this research we compare two approaches (in particular, character-based machine learning and language modeling) and according to their results offer the best solution for the diacritization problem solving. Parameters of tested approaches (i.e., a huge variety of feature types for the character-based method and a value *n* for the n-gram language modeling method) were tuned to achieve the highest accuracy. Despite the main focus is on the Lithuanian language, we posit that obtained findings can also be applied to other, similar (Latvian or Slavic) languages.

During experiments we measured the performance of used approaches on 10 domains (including normative texts and non-normative Internet comments). The best results reaching ~99.5% and ~98.4% of the accuracy on characters and words, respectively, were achieved with the tri-gram language modeling method. It outperformed the character-based machine learning approach with the tuned feature set by ~1.4% and ~3.8% of the accuracy on characters and words, respectively.

Lietuviškus nenorminius tekstus (interneto komentarus, elektroninius laiškus, forumų tekstus, socialinių tinklų žinutes) įprasta rašyti nenaudojant diakritinių ženklų. Deja, tokius tekstus sudėtinga analizuoti automatiškai: morfologiniai ar sintaksiniai analizatoriai, automatinio vertimo įrankiai, paieškos sistemos ir kt. yra pritaikyti veikti su norminės kalbos tekstais. Įvairūs klaidų taisymo įrankiai (lietuvių kalbai) dažniausiai tik siūlo galimas taisymo alternatyvas, tačiau neanalizuoja konteksto ir nėra pritaikyti automatiškai atstatyti diakritinius ženklus visame tekste.

Iš didelės apimties tekstynų (daugiau nei 36 mln. žodžių) konstruojame kalbos modelius, įvertiname įvairių parametrų, požymių modifikacijų poveikį rezultatams, bei siūlome efektyviausią sprendimą lietuvių kalbai. Metodus įvertiname su 10 skirtingų sričių tekstais (interneto komentarais, populiariąja periodika, memuarais ir kt.). Geriausi rezultatai buvo pasiekti su trigraminiu kalbos modeliu: ~99,5% tikslumas atstatytiems simboliams, ~98,5% – atstatytiems žodžiams. Suformuluotos išvados gali būti naudingos ir kitoms panašioms kalboms (latvių, slavų).