# ROMANIAN AUTOMATIC DIACRITICS RESTORATION CHALLENGE

FLORIN IORDACHE, LUCIAN GEORGESCU, DAN ONEAȚĂ, HORIA CUCU

*Speech and Dialogue Research Laboratory,*
*University Politehnica of Bucharest,*
*florin.nicolae.iordache@gmail.com,*
*lucian.georgescu@speed.pub.ro,*
*dan.oneata@gmail.com,*
*horia.cucu@upb.ro*

## Abstract

Recent work from various research groups addresses the task of diacritics restoration for the Romanian language. However, since each method is trained and evaluated on a different dataset and using different metrics, these approaches remain somewhat isolated and incomparable. In this paper, we propose a solution by introducing the diacritics restoration challenge. Our contributions are three-fold. First, we release a unique dataset of talk-shows transcripts, consisting of more than 40M words and 2.5M sentences. Second, we provide an online framework through which teams can submit results, have them automatically evaluated and displayed in a competition leader-board. Third, we provide baseline results using methods that range from unigram models to character-level recurrent neural networks. Our best model is a two-layer BiLSTM, which achieves a word-level accuracy of 99.46%. All information regarding the challenge and the dataset is available at https://diacritics-challenge.speed.pub.ro.

*Key words* — challenge, diacritics restoration, neural networks, Romanian

## 1. *Introduction*

Many languages use scripts in which the usual letters can receive marks, named *diacritics*. Albeit small in size, diacritics are important as they alter the pronunciation and meaning of a word. The lack of diacritics can lead to ambiguous sentences with several meanings: *un roman la Roma* (*a Romanian in Rome* or *a Roman in Rome*). Electronic texts are usually written using English keyboards, which lack these characters, making it inconvenient to type correctly; this process results in large amounts of electronic texts without diacritics. In this paper we tackle the task of *automatic diacritics restoration* (ADR) – inserting the correct marks (comma, cedilla, umlaut, breve, acute, slash, etc.) and in the correct position (above, below, over, etc.) for the usual letters in an orthographic text.

ADR is an active field of study with on-going efforts in a wide variety of languages, including Czech, Polish, Romanian and Hungarian (Novák and Siklósi, 2015; Tufiș and Ceaușu, 2008; Mihalcea and Năstase, 2002), Turkish (Arslan, 2016), Arabic (Khorsheed, 2012; Belinkov and Glass, 2015; Nelken and Shieber, 2005; Schlippe *et al.*, 2008), Maori (Cocks and Keegan, 2011), Uyghur (Tursun and Cakici, 2017), Urdu (Raza and Hussain, 2010), Vietnamese (Pham *et al.*, 2017; Luu and Yamamoto, 2012;

Do *et al.*, 2013), as well as West African languages like Igbo (Ezeani *et al.*, 2016) and Yoruba (Adegbola and Odilinye, 2012; Scannell, 2011; De Pauw *et al.*, 2007).

We study ADR for the Romanian language. Romanian uses three diacritic marks (breve, circumflex, comma below) that can be used for four letters (*a*, *i*, *s*, *t*), as follows: *ă* (*a* breve), *â* (*a* circumflex), *î* (*i* circumflex), *ș* (*s* comma) and *ț* (*t* comma). Although there are only a few characters that can support diacritics, 30-40% of the words in a Romanian running text contain at least one character with diacritics (Tufiș and Chițu, 1999). Moreover, many of these words can take multiple valid forms depending on the choice of diacritics; for example, *pana*, *pană*, *până* are all valid words. A baseline that restores diacritics according to the most common occurrence (a unigram model) gives an error rate of 29.71% – making Romanian the most challenging European language in this regard (Náplava *et al.*, 2018).

Missing diacritics lead to poor intelligibility: the text needs to be read several times to eliminate ambiguities or, worse, the sentence context might not be enough to disambiguate the text meaning. More importantly, texts without diacritics cannot be correctly processed by natural language processing modules such as part-of-speech (POS) taggers, named entity recognizers and cannot be used as input for text-to-speech systems. Hence, an ADR system is of critical importance in all the above situations.

The task of ADR for the Romanian language was approached by several research groups in the past two decades. A knowledge-based method using POS tagging to disambiguate the different diacritical words hypotheses, was introduced in (Tufiș and Chițu, 1999) and later refined in (Tufiș and Ceaușu, 2008). A statistical approach using a character n-gram model, a memory-based learning system (TiMBL) and a decision tree classifier (C4.5) is described in (Mihalcea and Năstase, 2002). Word-level n-grams with a history of up to five words were employed in (Cucu *et al.*, 2014; Petrică *et al.*, 2014), while in (Ungurean *et al.*, 2008) the diacritics restoration task is regarded as a sequential filtering process based on unigrams and bigrams of diacritical words and trigrams of diacritical word-suffixes. Finally, starting from 2018 several deep learning architectures (Cristescu, 2018; Rușeți *et al.*, 2018; Nuțu *et al.*, 2019) were proposed for this task.

The various works on ADR for the Romanian language were performed on different training and evaluation datasets and used different evaluation metrics. Table 1 summarizes the sizes of the datasets and the performance metrics used. Moreover, even if the evaluation metrics seem similar, there are still subtle differences in how they are computed. For example, (Cucu *et al.*, 2014) computes the word error rate as the number of incorrectly restored words relative to *the total number of words* in the evaluation dataset. This metric is, indeed, equivalent to the complement of word-level accuracy, used in (Nuțu et al., 2019), but it differs from the word-level accuracy of (Rușeți et al., 2018), as they consider only words that might contain diacritics (that is, words that contain at least one of the characters a, i, s, t).

Our work aims to facilitate the comparison across different methods and metrics by providing a common testbed for the task of automatic diacritics restoration. We frame our proposal as a challenge to the community by providing datasets (both for training

**Table 1**: Romanian diacritics restoration initiatives, datasets and evaluation metrics. The accuracies marked with * are computed relative to the words or characters that might contain diacritics, as explained above.

| **Work** | **Dataset** | **Evaluation metric** |
|---|---|---|
| (Tufiș and Chițu, 1999) | Fiction (118k tokens), Philosophy (135k tokens), Journalism (92 k tokens) | Accuracy (word-level) |
| (Mihalcea and Năstase, 2002) | 3M words | Char F-score |
| (Tufiș and Ceaușu, 2008) | Journalism (6.6M words), Juridical (3.5M words) | Accuracy* (word/ char -level) |
| (Ungurean *et al.*, 2008) | Online Romanian websites (10M words) | Accuracy (word-level), Char F-score |
| (Cucu *et al.*, 2014) | Europarl (5M words), misc (11M words) | WER, ChER, Char F-score |
| (Petrică *et al.*, 2014) | Europarl (5M words), misc (11M words), talkshows (44M words), Antena3 (27M words), Realitatea (68M words), Libertatea (72M words) | WER, ChER |
| (Cristescu, 2018) | unknown | Accuracy (word-level) |
| (Rușeți *et al.*, 2018) | PAR (50M words) | Accuracy* (word/ char -level), Char F-score |
| (Nuțu *et al.*, 2019) | Subset of CoRoLa (1M words) | Accuracy (word/ char/ 3-gram -level) |

and evaluation) and a web interface where participants can log-in to submit their results and have the predictions automatically evaluated.

The rest of the paper is organized as follows. Section 2 describes the diacritics restoration task and it presents the evaluation metrics. Section 3 provides a detailed description of the dataset, such as some statistics regarding it. Section 4 introduces the online leaderboard used to track all the approaches and their accuracy results. Our baseline systems are presented in Section 5. The last section is reserved for the conclusions.

## 2. *Task description and performance metrics*

The task of automatic diacritics restoration implies identifying the characters which require diacritics and replacing them with their correct diacritical form. This decision is made according to each word and the context in which it is found.

In the Romanian language, there are 4 characters that can have diacritics, each of them being able to have one or more diacritical forms, as follows: (i) *a – ă, â,* (ii) *i – î,* (iii) *s – ș*, (iv) *t – ț*.

There are multiple ways of evaluating an automatic diacritics restoration system. The most natural metric is accuracy. Accuracy can be computed at both word or character level, as the number of correct characters (or words) in the output text, relative to the number of characters (or words) in the groundtruth text. These metrics are presented in equations (2) and (4) and further called CA2 and WA2.

Provided that only a few characters accept diacritics (*a, i, s, t*), a more appropriate manner to compute the character accuracy would be to take into consideration only these characters, as in equation (1). When it comes to words, the accuracy can be computed as the number of correct words accepting diacritics in the output text relative to the number of words which could accept diacritics in the groundtruth text. The term "word accepting diacritics" refers to any word that contains at least one character that might accept diacritics: *a, i, s, t.*

$$CA1 = \frac{\#\ correct\ characters\ accepting\ diacritics\ in\ output\ text}{\#\ characters\ accepting\ diacritics\ in\ \ groundtruth\ text} \qquad (1)$$

$$CA2 = \frac{\#\ correct\ characters\ in\ output\ text}{\#\ characters\ in\ groundtruth\ text} \qquad (2)$$

$$WA1 = \frac{\#\ correct\ words\ accepting\ diacritics\ in\ output\ text}{\#\ words\ accepting\ diacritics\ in\ groundtruth\ text} \qquad (3)$$

$$WA2 = \frac{\#\ correct\ words\ output\ text}{\#\ words\ in\ \ groundtruth\ text} \qquad (4)$$

The primary metric that we decided to use for the proposed challenge is WA1. For a more detailed analysis of the results, the application also offers information about the following metrics: WA2, CA1, CA2, F-score per character and total F-score. F-score represents the harmonic mean of two other well-known metrics: precision and recall. The closer these values are to 1, the higher the performance. On the other hand, a low value for the precision means that the correction of the diacritics has been done, but in a wrong way, while a low value for the recall means that some corrections of the diacritics have been omitted.

## 3. *Dataset description*

The Diacritics restoration text corpus was created by the Speech and Dialogue Research Laboratory (SpeeD). With about 42 million words, this is one of the largest Romanian text corpus containing diacritics freely available for research. The texts represent transcriptions of several Romanian talk-shows and online news. The raw text was preprocessed and normalized using *TextCorpusCleaner*, a tool created by SpeeD to uniformize diacritics and hyphens, replace URLs, emails and abbreviations with their spoken form, replace numbers (different formats) with text, handle special characters, lowercase text, and remove text in other languages. The corpus contains a training set of 39.7 million words with correct diacritics and an evaluation set of 2 million words without diacritics. Table 2 describes the training corpus in terms of the number of words, letters and sentences, considering the presence of diacritics.

A word without diacritics there may take one or more forms with diacritics, each having a different meaning. Here are some such examples from the dataset:

- suta →șuta, suta, șuță, șută, sută
- pana → pană, pana, până
- fata → fâța, fâță față, fătă, fata, fată, fața
- pasa → pașa, pasă, pașă, păsa

**Table 2**: Training corpus description. A “word with diacritics” is a word which contains at least one letter with diacritics (*ă, â, î, ş or ţ*), while a “word accepting diacritics” is a word which contains at least one letter that might accept diacritics (*a, i, s, t*)

| **Statistics** | **Number of occurrences** |
|---|---|
| Words with diacritics | 13M |
| Words accepting diacritics | 29M |
| Total number of words | 39M |
| Letters with diacritics | 14M |
| Letters accepting diacritics | 56M |
| Total number of letters | 224M |
| Number of sentences | 2.6M |
| Number of unique words | 0.16M |

Figure 1 and Figure 2 present some statistics regarding the training set. The first figure is a heatmap that illustrates the relationship between the words length and their number of diacritical forms: the darker the colour, the greater the number of words (the colour is on logarithmic scale).We noticed that most words have lengths of 6-11 characters and it is obvious that most words have a single diacritical form. Another interesting fact we observed is that shorter words have more diacritical forms. The second diagram is a histogram that shows the distribution of sentences according to their length as the number of words. Most sentences contain between 4-7 words.

## 4. *Evaluation details*

Our laboratory introduces a diacritics restoration challenge for the Romanian language, available at http://diacritics-challenge.speed.pub.ro/. The purpose is to keep track of performance using the current state-of-the-art technology, while systems training and testing takes place on the standard dataset, proposed in Section 3. More information about the dataset can be found on the main page of the challenge, under the *Data* section. The usage of other datasets than those specified is not permitted.

First of all, the users must register using a username and password in the Register section of the menu, as well as providing some personal info: first name, last name, email, affiliation, personal webpage URL. After registration, the user will receive an email to confirm his identity. The Submit section allows users to upload their output file obtained after restoring diacritics on the evaluation subset. The submission file should have the same number of words as the evaluation file, but with the characters a, i, s, t being replaced with their correct version. The leaderboard menu helps users to track the performance of the submitted results. There is a public leaderboard, which reports the best performances of each user, along with information about the total number of submissions, as well as the date of the last submission. The menu also includes a private leaderboard, showing all the user’s submissions and their performance.

This challenge intends to be of interest to all researchers working on diacritics restoration task on Romanian language.

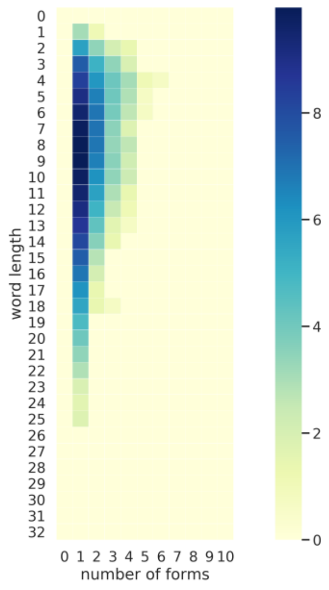

**Figure 1**: Word length vs. number of forms

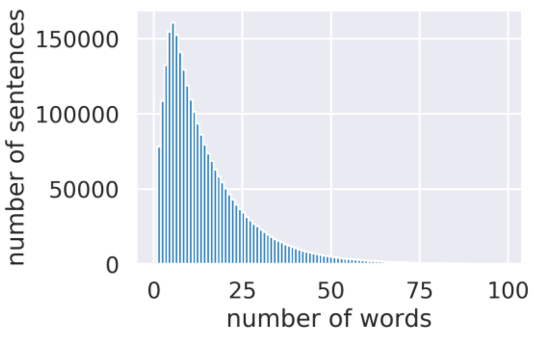

**Figure 2**: Sentences length histogram

## 5. *Challenge baselines*

### 5.1. *Simple baselines*

Two simple baselines were created in order to set a reference for the range of the possible performance values. The first and most straightforward approach has involved comparing the evaluation set without any diacritics against the groundtruth. We obtained a WA2 of 68.11%, which means that this is the proportion of words that have no diacritics. The second approach involved the use of a 1-gram probabilistic model, to replace each word that might have diacritics with its most commonly encountered form containing diacritics. In this case, we obtained a WA2 of 97.57%. The conclusion that emerges from these two baselines is that any simple algorithm should not obtain accuracies under 68%, while a more elaborate algorithm should definitely be more accurate than 97% (WA2 obtained with the 1-gram approach).

### 5.2. *Recurrent neural network approach*

Because the task is of the type sequence-to-sequence, a natural option is the use of a recurrent neural network (RNN). Its role is to extract useful information about the sequence of characters making up the words and then the sentences. We work at the character level to prevent the situations when some words are not found in the vocabulary. All the diacritics were removed from the training set in order to be used at the network input. At each iteration, the loss function, categorical cross-entropy, is computed as the difference between the text obtained at the network output, which is supposed to have the correct diacritics, and the training set containing diacritics. The network was trained using character level examples. A total of 44 symbols were used: non-diacritical letters, hyphens, space or end of line symbols. Each one was assigned to an integer number. The next step involves moving to a vector representation, where each trainable example is associated with a vector, which corresponds to a recurrent cell in the network. On top of the RNN is a fully connected layer, characterized by the softmax activation function. Two approaches regarding the network output were used:

- each symbol is represented at the output as its index from the list of possible output symbols. Because a total of 44 symbols are used, the network could perform the classification task into 44 classes. But this method is not optimal, because the large number of output classes leads to a large number of parameters, implicitly to a longer duration of training.
- to prevent the previous issue, the network output consists into 3 possible classes as follows: 0 - no diacritics required, 1 - one of the *î*, *ş*, *ţ*, *â* (the final decision depends on the input character) and 2 -*ă*.

The architecture depicted in Figure 3 represents the general concept of the network. In our experiments, it has undergone changes regarding the type of the recurrent cell (LSTM or GRU), the type of the recurrent layer (unidirectional or bidirectional) or the number of recurrent layers.

**Data Preprocessing.** The sentences in the training set are of different length in terms of the number of characters. Although RNNs allows variable length sequences to be trained, this results in a longer training time, since operations cannot be parallelized. Thus it is desired to train on sequences of equal lengths. Therefore, a couple of data preprocessing approaches can be taken into account. The simplest assumes that all sequences are brought to the length of the largest sequence, by filling them with the space character. However, this method creates a lot of redundancy and leads to a very long training time, so we did not use it. Instead, we used the following two methods:

1. inside cutting: all training examples were joined together into one single large string, followed by dividing it into equal pieces of characters. Empirically it was determined that the optimal value of this length is 256 characters. The method works, but there are situations when some words are cut inside or words from different sentences are joined.
2. smart grouping: an optimized method was applied that adaptively processes the sentences. The aim was to obtain sequences of 256 characters, but without cutting inside the words. Words shorter than a threshold are selected, and then grouped so that the resulting strings have a length of 256 characters or less. Those that have less than 256 characters are filled with the space character.

Our first two experiments consisted of using a recurrent network with LSTM cells, in which we applied the above listed data preprocessing methods. We obtained a value equal to 98.67% WA2 when we applied the inside cutting method, respectively 99.03% WA2 when we applied the smart grouping method. Therefore, there is a considerable relative improvement brought by the optimized data preprocessing method, being 27% better than the other one. We use this method in future experiments.

The experiments from Table 3 aimed at trying several modifications in terms of network architecture. Recurrent unidirectional and bidirectional networks were tested, both with LSTM cells and GRU cells, and the number of layers ranged from one to two layers. The performances were higher for the two-layer networks, compared to the single-layer ones. GRU cells obtained weaker results, but close to the results obtained with LSTM.

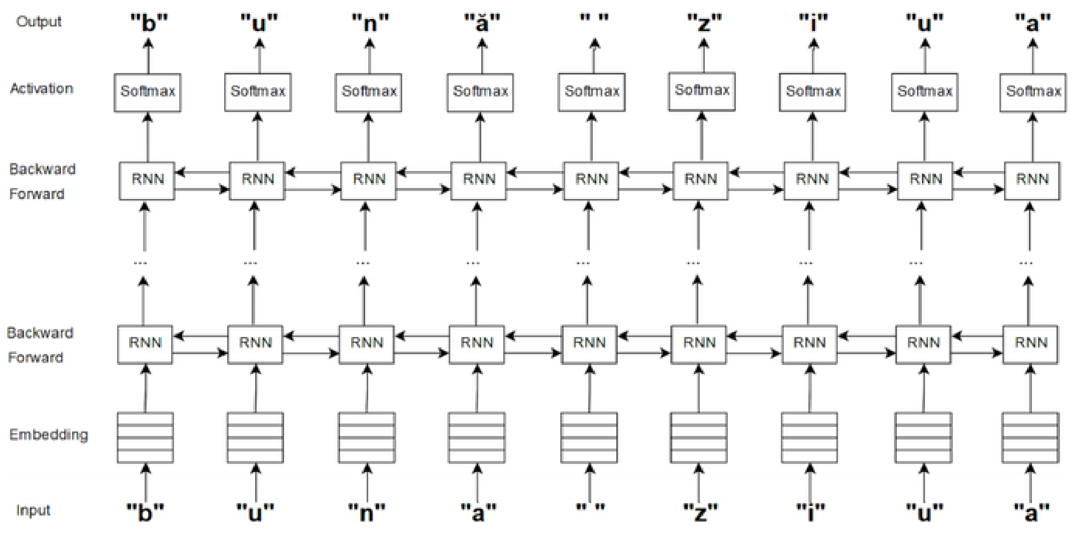
Output
"b" "u" "n" "ă" " " "z" "i" "u" "a"
Activation
Softmax
Backward
Forward
RNN
Embedding
Input
"b" "u" "n" "a" " " "z" "i" "u" "a"

**Figure 3**: Character-level RNN: it takes as input a character, it predicts its diacritical form

**Table 3**:Results varying the cell type and the number of layers

| Network type | Parameters [millions] | Epochs | Training time [h] | Evaluation (WA2 [%]) |
|---|---|---|---|---|
| BiLSTM-1 | 0.8 | 62 | 4.1 | 98.79 |
| **BiLSTM-2** | **2.4** | **55** | **9.2** | **99.42** |
| LSTM-1 | 0.4 | 71 | 2.9 | 88.47 |
| LSTM-2 | 0.9 | 88 | 7.0 | 88.88 |
| BiGRU-1 | 0.6 | 65 | 4.0 | 98.59 |
| BiGRU-2 | 1.8 | 47 | 6.1 | 99.29 |

Finally, the bidirectional two-layer LSTM architecture obtained the best results, but it is also the most expensive, both in terms of the number of parameters, implicitly of the memory, but also of the time required for training. The accuracy has been further enhanced by running hyperparameter tuning experiments, but also by applying optimization techniques, such as: early stopping, dropout on the recurrent layers or on the output layer, finding a suitable value for the learning rate or dimensioning the size of the batches used for training.

## *6. Conclusions*

In this paper we introduce a diacritics restoration challenge for the Romanian language. We provide a dataset containing both training and evaluation subsets, summing 42 million words, as well as an online leaderboard to keep track of the various submissions and their performances. We also offer baseline results based on a unigram model and recurrent neural networks. By carefully tuning the hyperparameters (number of layers, cell type, learning rate, dropout) and applying optimization methods such as early stopping, we obtained the best results using a two-layer BiLSTM network, gaining an accuracy value of 99.46% (WA2). We invite the community to further improve upon these results.

***Acknowledgements***

This work was supported by a grant of the Romanian Ministry of Research and Innovation, CCCDI – UEFISCDI, project number PN-III-P1-1.2-PCCDI-2017-0818, within PNCDI III.

## References

Adegbola, T., Odilinye, L. U. (2012). Quantifying the effect of corpus size on the quality of automatic diacritization of Yoruba texts. In *Spoken Language Technologies for Under-Resourced Languages*,2012, pp. 48–53.

Arslan, A. (2016). Deasciification approach to handle diacritics in Turkish information retrieval. In *Information Processing & Management*, vol. 52, pp. 326–339, 2016.

Belinkov, Y., Glass, J. R. (2015). Arabic diacritization with recurrent neural networks. In *EMNLP*, 2015, pp. 2281–2285.

Cocks, J., Keegan, T. T. (2011). A word-based approach for diacritic restoration in Maori. In *Proceedings of the Australasian Language Technology Association Workshop*, 2011, pp. 126–130.

Cristescu, H. (2018). Romanian-diacritics-restoration. Available: https://github.com/horiacristescu/romanian-diacritic-restoration. [Accessed at 26 June 2019].

Cucu, H., Buzo, A., Besacier, L., Burileanu, C. (2014). SMT-based ASR domain adaptation methods for under-resourced languages: Application to Romanian. In *Speech Communication Journal*, Vol. 56 – Special Issue on Processing Under-Resourced Languages, pp. 195-212, Jan 2014.

Do, T. N. D., Nguyen, D. B., Mac, D. K., Tran, D. D. (2013). Machine translation approach for Vietnamese diacritic restoration. In *Asian Language Processing (IALP), 2013 International Con-ference on. IEEE,* 2013, pp. 103–106.

Ezeani, I., Hepple, M., Onyenwe, I. (2016). Automatic restoration of diacritics for Igbo language. In *International Conference on Text,Speech, and Dialogue*. Springer, 2016, pp. 198–205.

Khorsheed, M. (2012). A HMM-based system to diacritize Arabic text. In *Journal of Software Engineering and Applications*, vol. 5, p. 124, 2012.

T. A. Luu, T. A., Yamamoto, K. (2012). A pointwise approach for Vietnamese diacritics restoration," In *Asian Language Processing (IALP), 2012 International Conference on. IEEE*, 2012, pp. 189–192.

Mihalcea, R., Năstase, V. (2002). Letter level learning for language independent diacritics restoration. In *Proceedings of the 6th conference on Natural language learning*-Volume 20 (pp. 1-7).

Náplava, J., Straka, M., Straňák, P., & Hajic, J. (2018). Diacritics restoration using neural networks. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC-2018).*

Nelken, R., Shieber, S.M. (2005). Arabic diacritization using weighted finite-state transducers. In *Proceedings of the ACL Workshop on Computational Approaches to Semitic Languages.* Assoc. for Computational Linguistics, 2005, pp. 79–86.

Novák, A., Siklósi, B. (2015). Automatic diacritics restoration for Hungarian. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing,* 2015, pp. 2286–2291.

Nutu, M., Lorincz, B., Stan, A. (2019). Deep Learning for Automatic Diacritics Restoration in Romanian. In *Proceedings of the IEEE 15th International Conference on Intelligent Computer Communication and Processing,* Cluj-Napoca, Romania.

Orife, I. (2018). Attentive Sequence-to-Sequence Learning for Diacritic Restoration of Yoruba Language Text. In *Proceedings of Interspeech.*

De Pauw, G., Wagacha, P. W., De Schryver, G.-M. (2007). Automatic diacritic restoration for resource-scarce languages. *In International Conference on Text, Speech and Dialogue.* Springer, 2007,pp. 170–179.

Petrică, L., Cucu, H., Buzo, A., Burileanu, C. (2014). A robust diacritics restoration system using unreliable raw text data. In *Proceedings of the 4th International Workshop on Spoken Language Technologies for Under-resourced Languages (SLTU),* St. Petersburg, Russia, pp. 215-221.

Pham, T.-H., Pham, X.-K., Le-Hong, P. (2017). On the use of machine translation-based approaches for Vietnamese diacritic restoration. In *arXiv preprint arXiv:1709.07104,* 2017.

Raza, A. Hussain, S. (2010). Automatic diacritization for Urdu. In *Proceedings of the Conference on Language and Technology,* 2010, pp. 105–111.

Rușeți, Ș., Cotet, T.-M., Dascălu, M. (2018). Romanian Diacritics Restoration Using Recurrent Neural Networks, In *Proceedings of the International Conference on Linguistic Resources and Tools for Natural Language Processing – ConsILR.*

Scannell, K. P. (2011). Statistical unicodification of African languages. In *Language resources and evaluation,* vol. 45, no. 3, p. 375, 2011.

Schlippe, T., Nguyen, T., Vogel, S. (2008). Diacritization as a machine translation problem and as a sequence labeling problem. In *AMTA-2008. MT at work: In Proceedings of the Eighth Conference of the Association for Machine Translation in the Americas, 2008,* pp. 270–278.

Tufiș, D., Chițu, A. (1999). Automatic diacritics insertion in Romanian texts. In *Proceedings of the International Conference on Computational Lexicography COMPLEX,* (Vol. 99, pp. 185-194).

Tufiș, D., Ceaușu, A. (2008). DIAC+: A professional diacritics recovering system. In *Proceedings of LREC 2008.*

Tursun, O., Cakici, R. (2017). Noisy Uyghur text normalization. In *Proceedings of the 3rd Workshop on Noisy User-generated Text,*2017, pp. 85–93.

Ungurean, C., Burileanu, D., Popescu, V., Negrescu, C., Derviș, A. (2008). Automatic diacritic restoration for a TTS-based e-mail reader application. In *University Politehnica of Bucharest Scientific Bulletin,* Series C, vol. 70, no. 4, pp. 3-12.