Deep Learning for Automatic Diacritics Restoration in Romanian

Maria Nuţu[a,b], Beáta Lőrincz[a,b], Adriana Stan[a]
[a]Communications Department, Technical University of Cluj-Napoca, Romania
[b]Department of Computer Science, "Babeş-Bolyai" University, Cluj-Napoca, Romania
{maria.nutu, beata.lorincz, adriana.stan}@com.utcluj.ro

***Abstract*—In this paper we address the issue of automatic diacritics restoration (ADR) for Romanian using deep learning strategies.**

We compare 6 separate architectures with various mixtures of recurrent and convolutional layers. The input consists in sequences of consecutive words stripped of their diacritic symbols. The network's task is to learn to restore the diacritics for the entire sequence. No additional linguistic or semantic information is used as input to the networks.

The best results were obtained with a CNN-based architecture and achieved an accuracy of 97% at word level. At diacritic-level the accuracy of the same architecture is 89%.

***Index Terms*—automatic diacritics restoration, deep neural networks, LSTM, CNN, sequence-to-sequence, Romanian**

I. Introduction

Automatic Diacritics Restoration (ADR) is the process of restoring the diacritic symbols in orthographic texts. The applications of this process are numerous and include: spelling checkers, lexical disambiguation, part-of-speech tagging, natural language understanding, etc. The lack of diacritics is predominant in electronic texts where the user does not use adequate text editing software, or is not technologically proficient so as to use the diacritic symbols specific to his or her native/acquired language.

Most of the European languages contain different sets of diacritic symbols in their alphabets, with the most numerous being in French and Slovak. The set of diacritics used in European languages based on the Latin alphabet are illustrated in Table I.

Romanian uses 5 diacritic letters: *ă, â, î, ş* and *ţ*. Although not all words have alternative spellings with and without diacritics, in some cases, a missing diacritic could completely change a word's meaning (e.g *peste* = over vs. *peşte* = fish), while in other cases, the absence of the appropriate diacritic in the word's ending letter makes it impossible to discern between the definite or indefinite form of a noun (*mamă* = a mother vs. *mama* = the mother).

Tufiş et al. [1] reports that between 25% and 45% of the Romanian words contain diacritics, while in a random French text, only 15% of the words contain diacritic symbols [2]. The diacritic percentage across the European languages is reported in [3].

Motivated by the the relevance of diacritic restoration across various text-based applications, in this work we address the Romanian ADR problem using sequence-to-sequence deep learning architectures based on convolutional and recurrent neural networks.

TABLE I
Diacritics in European Languages with Latin based alphabets

Language	Diacritics	Language	Diacritics
Albanian	ç ë	Italian	á è é ì í ï ò ó ù ú
Basque	ñ ü	Lower Sorbian	ċ č ě ł ń ŕ ś š ź ž
Breton	â ê ñ ú ö	Maltese	ċ ġ ż
Catalan	à ç è é í ï ò ó ú ü	Norwegian	å æ ø
Czech	á č é í ñ ó ř š ý ž	Polish	ą ę ć ł ń ó 's 'z ż
Danish	å æ ø	Portuguese	â ã ç ê ó ô ũ ü
Dutch	ë	Romanian	ă â î ş ţ
English	none	Sami	á ï č d- ń ŋ š t- ž
Estonian	ä č ō ö ž	Serbo-Croatian	ć č d- š ž
Faroese	á æ d- ó øú ý	Slovak	á ä č d' é Í ñ ó ô ŕ š t' ú ý ž
Finnish	ä å ö š ž	Slovene	č š ž
French	á â æ ç é è ë ê î 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 ö ş ü
Hungarian	á é í 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	ă ě ĭ ŏ ŭ ŵ ŷ

The paper is structured as follows: Section II is a brief overview of the state-of-the-art methods used in ADR. Section III outlines the sequence-to-sequence architectures, while Section IV presents the dataset and the tested architectures. The final results are illustrated in Section V. The conclusions and future perspectives are summarized in Section VI.

II. Related work

With the increase in the use of electronic devices across different social and cultural categories, the need for high-quality ADR applications is more prevalent, and so is the number of published scientific studies. Simard [2] employs Hidden Markov Models trained at word level on French texts. For the Vietnamese language, Nguyen et al. [4] combine Adaboost and C4.5 decision tree classifiers with a letter-based feature set in five different strategies: learning from letters, learning from semi-syllables, learning from syllables, learning from words, and learning from bi-grams.

A deep learning approach for diacritics restoration is proposed by Náplava et. al. in [5] and uses Bidirectional Neural Networks combined with a language model. The model was tested for 23 languages, including among others Czech, Slovak and Romanian.

For the Romanian language, in particular, the works of Mihalcea et al. [3], [6] explore instance based learning at letter

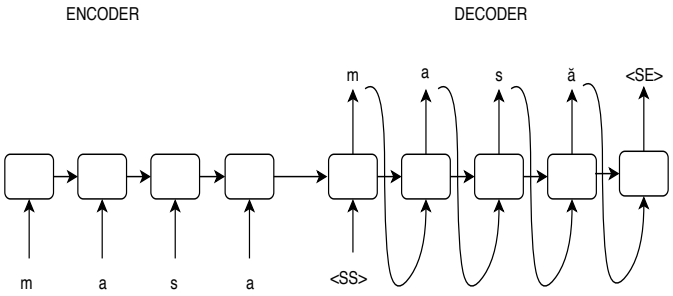

Fig. 1. Sequence-to-sequence flow

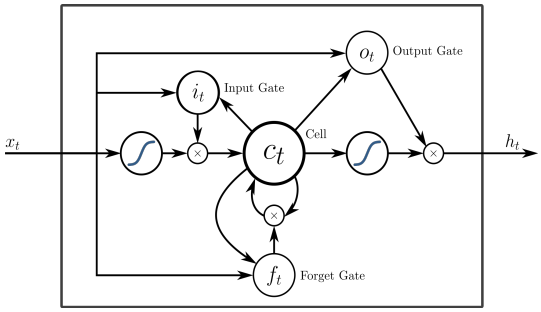

Fig. 2. LSTM memory cell [11]

level, using the Tilburg memory and the C4.5 decision tree classifier, scoring an overall F-measure of 98.30 %.

Tufiş et al. [1] propose a Part-Of-Speech tagger and the use of two lexicons to solve the ambiguity problem in Romanian ADR. An overall accuracy of 97.4 % is achieved at word level.

Ungureanu et. al [7] propose a word classification schema, based on the occurrence of diacritics in each word (words always written with diacritics, words with no diacritics at all and words with different diacritical written pattern - words which change their meaning as diacritics are missing, as shown in Section I). Then these categories are distilled into two dictionaries. During training and testing, the two lexicons are used to improve the ADR results, obtaining an overall F-measure of 99.34%.

In [8] Petrică presents a diacritics restoration system trained on unreliable raw data sets. First, the correctly spelled sections are identified and used as training data for the ADR. Second, the trained ADR is applied to the remaining parts of the initial text.

The previously described approaches use language models and linguistic information extracted from the texts at different levels. In this work, we propose a deep learning approach to solve the ADR problem for Romanian using only character sequences and without any expert linguistic knowledge.

III. Sequence to sequence learning

The sequence-to-sequence (seq2seq) [9] architecture is designed to handle input and output sequences with different lengths. The most common applications for this architecture include automatic machine translation, video captioning, speech recognition and speech synthesis.

Broadly speaking, the seq2seq architecture is formed of two parts: an *encoder* and a *decoder*, each of them being a separate neural network. The encoder is responsible for understanding the input and representing it in a lower dimensional space. The output of the encoder will then be used to condition the decoding network's prediction. Figure 1 presents a seq2seq model for the word "masa" as input and "masă" as output. The tags <SS> and <SE> mark the start and the end of the sequence. The most prevalent architectures behind the encoders/decoders are the recurrent and convolutional neural networks.

A. Recurrent Neural Networks

Recurrent Neural Networks (RNNs) are a type of neural network in which the output of the current time step is conditioned on the output of the previous time step. As a result, RNNs are commonly used to model temporal sequences. However, a major problem with vanilla RNNs is that they cannot model sequences in which the temporal dependencies are stretched across multiple time steps.

The solution for this problem is to use more advanced network nodes, in which an internal state of the node can memorize or forget data snippets which are of interest to the current prediction. One such specialized node is the Long Short Term Memory (**LSTM**) cell [10].

A LSTM cell (graphically depicted in Figure 2) contains the following elements:

- forget gate f_t - a neural network (NN) with sigmoid activation
- input gate i_t - a NN with sigmoid activation
- output gate o_t - a NN with sigmoid activation
- hidden state h_t - a vector
- memory state c_t - a vector

The input gate selects what new information to be stored in the current cell at a time step *t*. The forget-gate expresses the amount of information which will be discarded, while the output-gate will provide the activation to the final output of the LSTM block. The hidden state is calculated from the cell state passed through an activation function and element-wise multiplied with the output vector at the time step t.

B. Convolutional Neural Networks

Convolutional Neural Networks (CNN), originally used in image processing, are another type of deep networks largely used for pattern recognition tasks.

A simple CNN architecture contains the following elements:

- a convolutional layer
- a non-linear activation layer
- a pooling (or sub sampling) layer
- a fully connected (softmax) output layer.

The convolutional layer defines a non-linear filter bank (or kernel), which is shifted over the input features using a fixed stride and generates a multi-dimensional feature map, which is processed by a non-linear activation function. The pooling layer reduces the representation of the convolutional layer's

output, as well as decreases the memory requirements. In general, the polling layer is placed between the convolutional layers. The features with the highest values (maxpool) are fed into a fully connected layer, whose activations are finally passed into a softmax layer. The output of the softmax function represents the estimated probability distribution over the output labels. In some cases, a normalization layer is stacked on the pooling layer to normalize the data, with mean 0 and variance 1. The normalization step ensures the networks stability.

The characteristics highlighted above make the seq2seq learning a good candidate for the Romanian ADR problem.

IV. Experimental setup

A. Training Data

For training and testing our models, we selected a subset of the CoRoLa text corpus [12]. The subset contains 51.043 sentences with 1 million tokens and 63.194 unique words. The style of the text is belletristic. The corpus is not purposely build for ADR tasks, but can be considered as a reliable source of correctly typed text (i.e. containing the correct diacritics) in Romanian as it was manually annotated at word-level with several linguistic information. We subsequently split the dataset into disjoint training (80%) and testing (20%) sets, each of them being individually shuffled.

A few pre-processing steps were performed and include the following operations:

- convert text to lowercase
- strip the digits and punctuation
- strip the diacritics
- parse the text in trigrams
- create pairs of input-target sequences
- append a start-character (”\t”) and an end-character (”\n”) to the target trigram

An example of a pre-processed sentence is shown in Table II. The obtained input-output pairs are illustrated in Table III.

TABLE II
PRE-PROCESSING EXAMPLE

Initial sentence	”Mă uitasem la ceas, era încă ora 22.00.”
Pre-processed sentence	”ma uitasem la ceas era inca ora”

TABLE III
INPUT-OUTPUT TRIGRAMS FOR A CHOSEN SENTENCE

Input sequence	Target sequence
ma uitasem la	\t mă uitasem la \n
uitasem la ceas	\t uitasem la ceas \n
la ceas era	\t la ceas era \n
ceas era inca	\t ceas era încă \n
era inca ora	\t era încă ora \n

When an unknown input sequence is decoded, we begin with the starting character and use the decoder to predict the next character until the ending character is generated. The trigrams were chosen to represent the context of the current sequence.

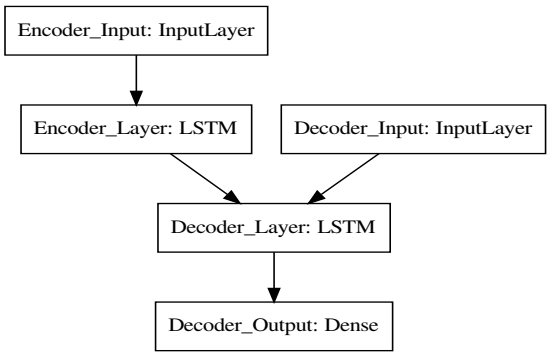

Fig. 3. **seq2seq-LSTM** model architecture

After the pre-processing steps, the train set ended-up containing 616.691 tokens, while the test set contained 162.791 tokens.

B. System architectures

For our initial tests we selected two ADR systems [5], [13] previously applied for Romanian. The systems were retrained using our dataset, but preserving the original parameter values.

Inspired by the architectures described in these two systems, we analyzed four other architectures with various combinations of recurrent and convolutional layers. For implementation, we relied on Keras[1] with TensorFlow[2] as backend. The networks' hyperparameters were tuned using a small development set.

All 6 architectures are described in the following subsections with the previously published works marked with an asterisk (*). All systems were trained over 50 epochs.

*1) One layer LSTMs (**ID: seq2seq_LSTM**):* In the RNN sequence-to-sequence architecture the encoder and decoder both included one LSTM layer. A latent dimension of 128 for both layers and a batch size of 512 were chosen. The input to the encoder and decoder was one-hot encoding at character level. The input of the decoder was also conditioned on the hidden state of the encoder. The output of the decoder LSTM layer is sent to a softmax dense layer with a dimension equal to the length of the one-hot encoded target character set. Figure 3 illustrates the architecture design of the RNN architecture.

*2) Stacked LSTMs (**seq2seq_stacked_*_LSTM**):* In order to improve the results, one additional LSTM layer was added to the encoder. The newly obtained encoder was tested in two different contexts. First, we used a 1 LSTM layer for the decoder (**ID: seq2seq_stacked_1_LSTM**). Then, another LSTM layer was stacked in the decoder (**ID: seq2seq_stacked_2_LSTM**).

[1]https://keras.io/
[2]https://www.tensorflow.org/

The model **seq2seq_stacked_1_LSTM** was trained with a 256 latent dimension and 128 batch size. For model **seq2seq_stacked_2_LSTM** a batch size of 512 and a latent dimension of 128 were used.

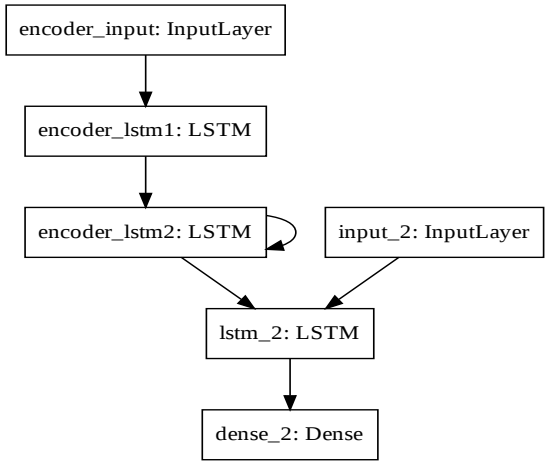

Fig. 4. **seq2seq_stacked_1_LSTM** model architecture

*3) Convolutional Sequence-to-Sequence (**ID: seq2seq_CNN**):* In our experiments, the CNN architecture contains 3 convolutional layers with 128 feature maps and a kernel of size 3, for both the encoder and the decoder networks. An attention architecture with a softmax activation follows the 3-layered convolutional decoder networks. The output is processed by another 2 convolutional layered architecture, with a softmax dense output. Figure 5 illustrates the model structure. The model is trained with a batch size of 1024 and a 128 latent dimension.

*4) *RNN and CNN hybrid model (**ID: seq2seq_hybrid**):* The RNN and CNN hybrid model [13] uses two paths - character level and word level. For the character path, an embedding layer feeds the input to 3 stacked CNN layers. The word path goes through embedding and a bidirectional LSTM (biLSTM). The two paths are merged by projecting words to characters based on a projection matrix which is received as an additional input. Hence, the character and word embeddings are jointly learned. These embeddings are fed to a stack of 3 convolutional layers. The output is predicted using a time distributed dense layer. We trained the network with a batch size of 32. The system architecture is illustrated in Figure 6.

*5) *RNN with language model:* In [5] a combination of character-level recurrent neural network based model and a language model are applied to automatic diacritics restoration.[3] The core model uses a bidirectional LSTM which deals with previous and next letter contexts in the sequence.

The bidirectional RNN contains 2 stacked layers with residual connections, composed of 300 LSTM units. A batch

[3]https://github.com/arahusky/diacritics_restoration

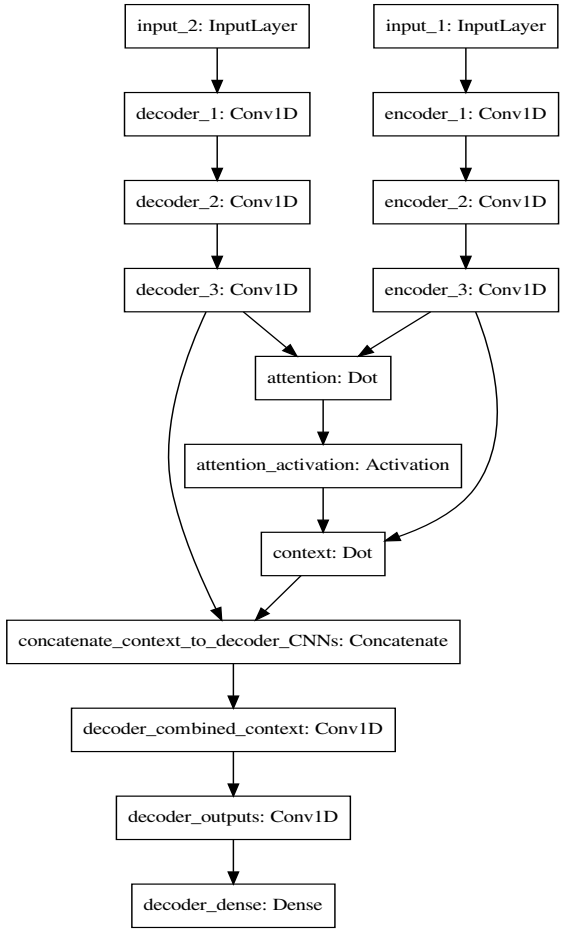

Fig. 5. **seq2seq-CNN** model architecture

size of 200 was chosen. The model language is based on left-to-right beam search. At each time step, the output of the biLSTM layers is reduced by a fully connected layer to v-dimensional vectors, where v is the size of the output vocabulary. A non-linear ReLU activation function is applied to the reduced vectors. The final output layer uses a softmax activation.

V. Evaluation

All 6 system architectures were evaluated using the *classification accuracy metric*, which is defined as the ratio between the correct predictions and the total number of samples.

We computed the accuracy at three different levels: trigram, word and character level. At trigram and word-level the accuracy reflects the number of correct predictions made by the system overall. At character-level, we computed the accuracy only for the characters which may be written with diacritic symbols (*a, i, s, t)*. Accuracy results for all the systems are presented in Table IV.

TABLE IV
NETWORK PARAMETERS AND ACCURACY RESULTS

Architecture ID	Latent dimension	Batch size	Accuracy		
			3-gram level	Word level	Character level
seq2seq_LSTM	128	512	75.50%	89.98%	71.61%
seq2seq_stacked_1_LSTM	256	128	79%	93 %	78%
seq2seq_stacked_2_LSTM	128	512	84%	94 %	82%
seq2seq_CNN	128	1024	**91%**	**97 %**	89%
seq2seq_hybrid [13]	N/A	32	77%	92%	84%
seq2seq_LSTM_Language_model [5]	300	200	84 %	96 %	**90%**

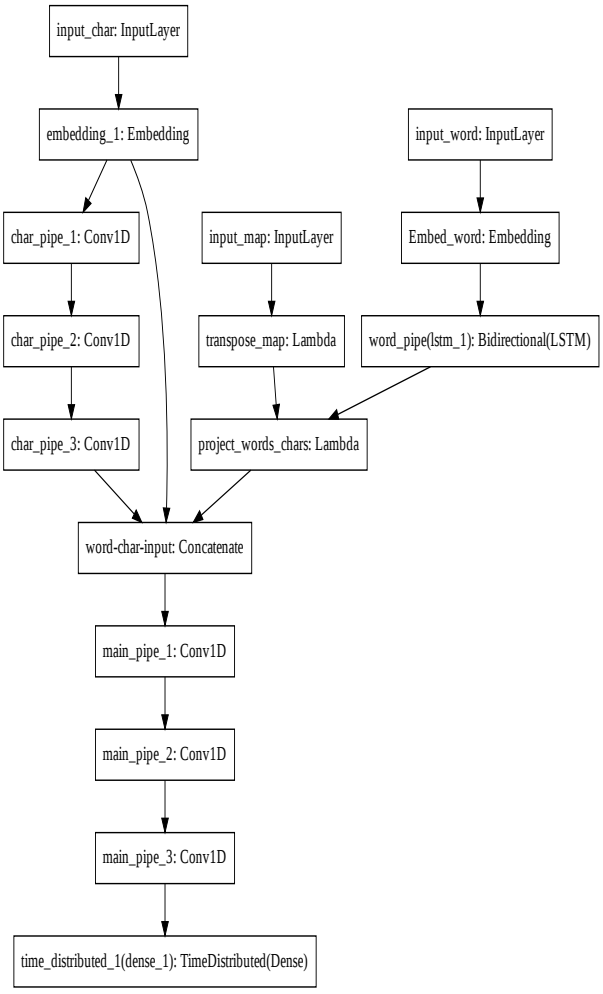

Fig. 6. **seq2seq-hybrid** model architecture

A separate set of results is shown in Table V, where the 4 ambiguous letter sets in Romanian (a-ă-â, i-î, s-ş, t-ţ) are analyzed individually.

The highest accuracy in terms of trigrams and words, was achieved for the convolutional network **seq2seq_CNN**, while the single-layer LSTM system **seq2seq_LSTM** had the lowest accuracy. One explanation can be found in the recurrence of the LSTMs, which may require larger data context, as opposed to the CNN, which uses an attention layer and sliding windows (kernels) to simulate the recurrence.

TABLE V
ACCURACY RESULTS FOR INDIVIDUAL AMBIGUOUS PAIRS OF THE BEST PERFORMING SYSTEM

Architecture ID	Accuracy			
	a-ă-â	i-î	s-ş	t-ţ
seq2seq_CNN	93.51 %	99.44 %	98.39 %	97.94 %

However, at character-level, the system described in [5] outperforms all other systems. The justification for this result can be the use in [5] of a language model together with the RNN, while our systems restore the diacritics without any additional linguistic information.

VI. CONCLUSIONS AND FUTURE WORK

In this work we compared 6 neural networks architectures for the task of automatic diacritics restoration applied to Romanian. All the models are trained using only parallel input-output pairs of texts with and without diacritics. As input to the sequence-to-sequence architectures we used character-level one-hot encodings. However, it is common practice in NLP to encode the words or characters using multidimensional embeddings obtained from large amounts of text data. These embeddings would allow the network to have an initial estimate of the characters' function in a language. So as future work, we intend to substitute the one-hot encoding with letter or word embeddings, and also to include additional linguistic or semantic information.

In our experiments we split the data in trigrams, both for training and for testing. Each network receives a diacritic-stripped trigram and predicts the entire corresponding sequence with diacritics. We intend to experiment with other N-gram, allowing the network to capture more context. One other means of improving the results is to predict the diacritics only for the sequence-ending word, considering all previous words to be correctly typed.

In addition, we are planning to investigate other types of fully convolutional neural networks, based on dilated convolutions combined with attention mechanisms, architectures largely used in Machine Translation and Speech Synthesis fields, but unexplored in the ADR domain.

ACKNOWLEDGMENT

This work was supported by a grant of the Romanian Ministry of Research and Innovation, PCCDI – UEFISCDI, project number PN-III-P1-1.2-PCCDI-2017-0818/73, within PNCDI III.

REFERENCES

[1] D. Tufiş and A. Chiţu, "Automatic insertion of diacritics in romanian texts," in *Proceedings of the 5th International Workshop on Computational Lexicography COMPLEX*, 1999, pp. 185–194.

[2] M. Simard, "Automatic insertion of accents in french text," in *Proceedings of the Third Conference on Empirical Methods for Natural Language Processing*, 1998, pp. 27–35.

[3] R. Mihalcea and V. Nastase, "Letter level learning for language independent diacritics restoration," in *proceedings of the 6th conference on Natural language learning-Volume 20*. Association for Computational Linguistics, 2002, pp. 1–7.

[4] K.-H. Nguyen and C.-Y. Ock, "Diacritics restoration in vietnamese: letter based vs. syllable based model," in *Pacific Rim International Conference on Artificial Intelligence*. Springer, 2010, pp. 631–636.

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

[6] R. F. Mihalcea, "Diacritics restoration: Learning from letters versus learning from words," in *International Conference on Intelligent Text Processing and Computational Linguistics*. Springer, 2002, pp. 339–348.

[7] C. Ungurean, D. Burileanu, V. Popescu, C. Negrescu, and A. Dervis, "Automatic diacritic restoration for a tts-based e-mail reader application," *UPB Scientific Bulletin, Series C*, vol. 70, no. 4, pp. 3–12, 2008.

[8] L. Petrică, H. Cucu, A. Buzo, and C. Burileanu, "A robust diacritics restoration system using unreliable raw text data," in *Spoken Language Technologies for Under-Resourced Languages*, 2014.

[9] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in neural information processing systems*, 2014, pp. 3104–3112.

[10] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[11] H. Graves, Abdel-Rahman. Peephole long short-term memory, wikimedia commons. [Online]. Available: https://commons.wikimedia.org/wiki/

[12] V. B. Mititelu, E. Irimia, and D. Tufis, "Corola - the reference corpus of contemporary romanian language." in *LREC*, 2014, pp. 1235–1239.

[13] H. Cristescu. Romanian diacritic restoration with neural nets. [Online]. Available: https://github.com/horiacristescu/romanian-diacritic-restoration