ROMANIAN DIACRITICS RESTORATION USING RECURRENT NEURAL NETWORKS

STEFAN RUSETI[1], TEODOR-MIHAI COTET[1], MIHAI DASCALU[1,2]

[1] *University Politechnica of Bucharest, Computer Science Department, 313 Splaiul Independentei, 060042, Bucharest, Romania*
[2] *Academy of Romanian Scientists, 54 Splaiul Independenței, 050094, Bucharest, Romania*
stefan.ruseti@cs.pub.ro, teodor_mihai.cotet@stud.acs.upb.ro, mihai.dascalu@cs.pub.ro

Abstract

Recurrent neural networks have been recently used for most tasks that require the encoding of a text sequence, which makes them suitable for diacritics restoration. We propose a novel model for Romanian language that predicts if a character needs to be replaced by one with diacritics using the surrounding context. Our network models the context with two different encoders, one with character embeddings and the other with pre-trained word embeddings. The model is trained on a corpus of transcriptions from the Romanian Parliament, which contains over 50 million words. Only using character embeddings, the model achieves over 99% accuracy for predicting the correct symbol for the potential diacritics candidates. We show that adding word-level information from the current context further improves accuracy to 99.37% at the character level, and at 98.57% on words.

Key words — diacritics restoration, Recurrent Neural Networks, character and word embeddings.

1. Introduction

Writing texts with Romanian diacritics is in general a cumbersome endeavor as most computer keyboards do not have special keys for specific Romanian letters with diacritics. Hence, people sometimes write digital text in Romanian language with the simplified version of letters with diacritics, namely the ASCII[1] version, thus removing all diacritics. Therefore, diacritics restoration is a mandatory step for adequately processing Romanian texts.

Even a very simple baseline, which substitutes a possible misspelled word with the most frequent candidate, can reach 95.59% accuracy on letters and 93.72% on words. The accuracy on letters is computed by considering all letters which may accept diacritics in the ASCII original version of the text that does not contain diacritics (e.g., letter "t" may accept diacritics, thus becoming "ț"). Similarly, the accuracy on words is computed taking into consideration all words containing at least one letter that may accept diacritics, as described above. Although the accuracy of diacritics restauration systems is quite high, the problem cannot be considered solved. For instance, a page in a novel contains around 400 words. If we consider that 26% of those words contain at least a letter with diacritics (as can be noticed in Table 1 for the Parliament – PAR º corpus described in the third section) and have an accuracy of 99.00% on words, this will still result in one erroneous word per page. Moreover, although systems can correctly determine the diacritics of most words, there are a few words that tend to be more ambiguous, requiring a deeper understanding of the context.

[1] http://www.asciitable.com/

Diacritics change the morphology of a word (‘cană’ and ‘cana’ meaning ‘cup’ and ‘the cup’, respectively) or even the complete meaning of the word (‘fată’ and ‘față’ meaning ‘girl’ and ‘face’, respectively); thus, this is a particularly important task for various NLP processes. Without it you do not know the word in case, thus you cannot perform dictionary checks, represent the word in a semantic space, check for stop-words, etc.

The paper continues by describing existing solutions that tackle this task. Next, we present the corpora used for training, the proposed solution in detail, together with performance results of different configurations of our model. The paper concludes with discussions, analyzes the most frequent errors, and considers alternatives for improving the model.

2. Related work

Most previous methods which were experimented for Romanian restoration of diacritics do not use neural networks. Among those that do, there are no solutions specifically optimized for this particular language (i.e., they were generally designed to work on many different languages).

Using a knowledge-based approach using *part-of-speech* tagging for choosing the correct word in ambiguous cases is described by Tufiş & Ceauşu (2008), reporting an 97.75% accuracy on words and 99.40% on characters. Another approach by Ungurean, Burileanu, Popescu, Negrescu, & Dervis (2008) uses sequential filtering with *n-grams* on words and suffixes (last 2-4 characters of each word) for speech synthesis reported an F-measure of 99.34% on characters. As reported in the paper, the use of suffixes was motivated by the fact that they generally capture the morphology of words and incorrect diacritics restorations of ambiguous words are mostly caused by characters at the end of the word.

One of the best results was obtained using an *n-gram* (Brown, Desouza, Mercer, Pietra, & Lai, 1992) language model, precisely solving ambiguity at word level by replacing each word with a version of its which has the highest probability given the *n* previous, already corrected, words. Creating this model requires computing the probabilities for the n-grams throughout the training process. The results reported by Petrică, Cucu, Buzo, & Burileanu (2014) are the highest on this task, namely errors of 0.52% for words and 0.116% on letters. Previous methods using n-grams have been tried; nonetheless their accuracy on both words and letters are lower than those previously mentioned. Their method removed unreliable online texts (e.g., news articles) in terms of diacritics orthography, filtering those parts which did not contain a high enough percentage of characters with diacritics.

One problem is that all these experiments were done using a different corpus, which makes difficult the comparison of results. However, studying these methods is important, as they provide relevant insights for improving a model.

3. Method

3.1 Corpus

Two corpora have been considered, both described in Table 1. The first corpus (PAR) contains transcriptions of the parliamentary debates in the Romanian Parliament[2], from 1996 to 2017. The corpus is very diverse in terms of subjects, as it contains the debates of the parliamentary committees in which economic, social, political and judicial issues are discussed exhaustively. Virtually, all texts in this corpus are valid, in the sense that all sequences of characters are meant to be a transcription of speech, thus grammatically valid sentences. In terms of diacritics some mistakes are expected in this corpus.

We have also considered to use the ROWIKI corpus of all Romanian texts from Wikipedia, but, as can be noticed from Table 1, the proportion of letters and words with diacritics are far lower (4.09% compared to 6.39% for letters in the PAR corpus). Moreover, unlike the previous corpus, this one contains some invalid sequences of characters which are not meant to be correctly formed sentences, but some representation of data (e.g. link references, tables with statistics, etc.). Because of these reasons we have decided not to use this corpus, although it was much larger and diverse than the PAR corpus. In Table 1 letters that accept/contain diacritics refer to all letters for which their ASCII version may accept diacritics (e.g., “a”, “ă”, “î”, “s”, “ș”, etc.).

Table 1: Corpora descriptions

		PAR		ROWIKI	
Letters	Letters with diacritics	15M	6.39%	38M	4.09%
	Letters that accept/contain diacritics	84M	35.28%	296M	31.78%
	All letters	239M	100.00%	933M	100.00%
Words	Words with diacritics	13M	26.37%	33M	16.40%
	Words that accept/contain diacritics	35M	70.41%	118M	57.98%
	All words	50M	100.00%	204M	100.00%
Sentences		2.6M		22.0M	
Unique words		0.21M		2.62M	

Different spelling rules existed during different time periods and the most important replacement consisted of the exchange of character „î” with „â” when it occurs in the middle of a word. This does not influence our model because the two cases produce different inputs for the network (different base letters); thus, no contradictory examples appear.

3.2 Network Architecture

The architecture of the proposed model is presented in Figure 1. The neural network is composed on three different paths: characters, current word, and the current sentence. The

[2] https://www.senat.ro/, http://www.cdep.ro/

main idea behind this architecture is to combine lexical with semantic information, therefore capturing more complex contexts.

The proposed neural network predicts the correct diacritic for each letter in the input text, only considering letters that can potentially be substituted with the diacritic version. Romanian language contains five different diacritics: “ă”, “â”, “î”, “ș”, and “ț”. We decided to group them into 4 different classes, based on the sign that is added to the letter: a) No sign, b) “ă”, c) “î” and “â”, and d) “ș” and “ț”.

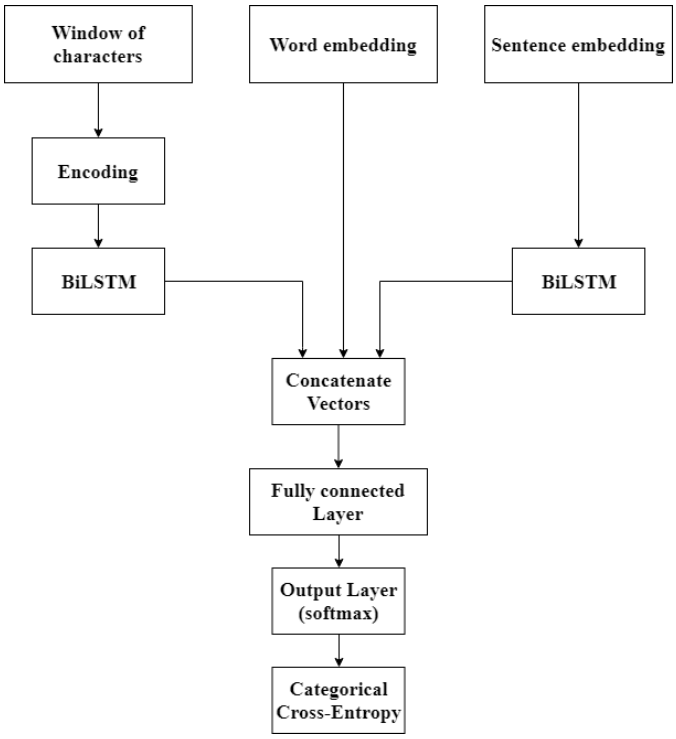

Figure 1: Network design

This grouping of diacritics can take advantage of the similarities between letters. For example, “î” and “â” actually represent the same sound. On the other hand, in the case of “ș” and “ț”, it is not obvious whether they occur in similar contexts. Because of this, experiments using 5 classes, with the two letters being separated, have also been conducted.

The first path in the network is represented by a Bidirectional Long Short-Term Memory (BiLSTM; Graves & Schmidhuber, 2005) encoder using character embeddings. A fixed size window is used to represent the context of the current character. The network can learn different similarities between letters by using character embeddings. Also, applying the encoder at the character level allows the network to generalize different forms of the same word, which will have very similar representations, and also generalize for unseen words, if they look similar to other known concepts. However, the model can benefit from semantic information, as well. In order to achieve this, another path was added to the architecture, represented by a BiLSTM encoder applied on the current sentence. The words in the sentence are represented by pre-trained FastText embeddings, which map words to points in a high-dimensional vector space based on the context in which they occur. The assumption was that

the two encoders are complementary one to another, and the model can learn how to combine the information from both.

The third path is represented by the embedding of the current word. This was added in order to help the network select which part of the encoded context is useful for the current entry. Experiments were also done with models without this word path, which proved to be less accurate.

3.3 Input Processing

The raw text used for input was first cleaned of unusual characters (those that were not in ASCII set were substituted with a fixed character) and all characters were transformed to their lowercase version. The input consists of 3 parts which were differently combined and fed to the neural network.

- The first part is the window of characters, containing a fixed window centered at the character for which the prediction is wanted. The embeddings for characters are learnt, as can be noticed in Figure 1.
- The second part is the embedding of the word which contains the character in question. The word embeddings are pretrained.
- The third part is the embedding of the entire sentence in cause. Each word is embedded independently, thus the sentence is represented as a list of words embeddings. Those word embeddings are identical with the ones from the second part (i.e., each word has the same embedding in both second and third input part).

For the word embeddings, we have used the pre-trained versions from FastText[3] which were trained by using the skip-gram model (Bojanowski, Grave, Joulin, & Mikolov, 2016). One advantage of FastText is that it can generate embeddings for words which are not in vocabulary by representing each word as a bag of characters n-grams and associating a vector representation for each n-gram, summing them up thus giving a vector for the word. This feature is very useful for rare words.

Moreover, we have averaged across all words (i.e. words that are in FastText vocabulary) that match the initial word (matching means that stripping the diacritics you get the initial words) because we did not know exactly the word in case without diacritics and thus, neither its corresponding embedding. Consequently, for each letter of the diacritics-free text (i.e. “a”, “e”, “i”, “s”, “t”) which is fed into the network, we create these 3 parts.

All implementation was done in Python. For the machine learning part, we have used TensorFlow[4], an open source framework which facilitates execution on GPUs. Furthermore, because of the intense computations of the input we have used the Tensorflow Dataset API[5] which enables parallel usage of multiple CPUs for input preparation, while the GPUs are used for training the model on already prepared input. Keras[6] was used to implement the neural network we have used, while nltk[7] was used for text tokenization.

[3] https://github.com/facebookresearch/fastText/blob/master/pretrained-vectors.md
[4] https://www.tensorflow.org/
[5] https://www.tensorflow.org/guide/datasets
[6] https://keras.io/
[7] https://www.nltk.org/

4. Results

In order to train and evaluate the model, the PAR corpus was split into three distinct partitions: 60% for training, 20% for validation and 20% for test. The architecture and hyperparameters of the neural network were chosen using the validation partition. Some of the best performing models were then evaluated on the test partition and the results are presented in Table 2.

Some of the hyperparameters of the models were not included in Table 2because they were not modified during the experiments. The cell size of the word BiLSTM encoder was set to 300, the same as the word embedding size. The character window size was set to 13, while the maximum sentence length was 31. The training was done in batches of 256, using the cross-entropy loss function and the Adam optimizer (Kingma & Ba, 2015).

Table 2: Char/word accuracy

Model	Char Embedding	Char LSTM	Hidden	Epochs	Dev char acc (%)	Test char acc (%)	Test word acc (%)
Chars	16	32	32	5	98.865	98.864	97.413
Chars	20	64	256	5	99.012	99.017	97.750
Chars (5 classes)	16	32	32	5	99.048	99.068	97.867
Chars	24	64	64	4	99.064	99.057	97.856
Chars + sentence	20	64	256	3	99.068	99.065	97.881
Chars + word	20	64	256	4	99.309	99.329	98.453
Chars + word + sentence	20	64	256	5	99.365	**99.378**	**98.573**
Chars + word + sentence	20	64	256, 128	5	**99.380**	99.366	98.553

A detailed analysis of the best models, that includes precision and recall values for each letter, was also performed. In Table 3 we present these results for the All-256-128 model that uses characters, words, and sentences with two hidden layers of sizes 256 and 128.

Table 3: Detailed performance per letter

Model	Letter	Precision (%)	Recall (%)	F-Score (%)
All-256-128	“a”	99.16	98.86	99.01
	“ă”	96.29	97.31	96.80
	“â”	99.17	98.80	98.99
	“i”	99.97	99.96	99.97
	“î”	99.65	99.72	99.69
	“s”	99.84	99.84	99.84
	“ș”	99.44	99.43	99.43
	“t”	99.84	99.77	99.80
	“ț”	98.97	99.29	99.13

5. Discussion

The results presented before show that adding contextual information from the sentence encoded with a BiLSTM can improve the results of a char-based approach significantly. However, only adding the encoded sentence, without the current word is not enough, the accuracy of the char model, and char+sentence being very similar.

Our initial choice of grouping the diacritics into 4 classes turned out to make sense for "â" and "î", but not for "ș" and "ț", since a model using only characters and 5 classes achieved an improvement of about 0.2% accuracy compared with the same model with 4 classes. Further experiments should be conducted in order to find the best combination of hyperparameters.

The detailed analysis shows that the model is biased towards choosing the class with no diacritics for each letter. Virtually all measures (precision, recall) are higher for letters with no diacritics ("a", "i", "s", "t") compared to the corresponding ones with diacritics. This can be explained by the higher number of letters with no diacritics compared to the ones with diacritics, but also by the missing diacritics in the corpus. For instance, the word "și" does not exist without diacritics (i.e., "si"), but the model reports an accuracy on this word of 99.82% and not 100% as expected. Indeed, in the corpus the word "și" is spelled wrongly as "si" a few times, thus confusing the neural network. Other similar words which exist in the Romanian language with just the diacritics version as "in" reported a similar accuracy; therefore, we suspect that missing some diacritics in the training corpus contributes significantly to the final error. Thus, having a perfect training corpus in terms of diacritics will probably increase the final accuracy.

Most of the hardest words to be restored are those in which the diacritics make the word (noun or adjective) indefinite as in "politică", "importanță" and "prezență", which mean the model still fails to distinguish between definite and indefinite words.

6. Conclusions

In this paper, we proposed a novel neural network architecture that combines lexical information at the character level, with a semantic representation of the surrounding context computed at the word level, using recurrent neural networks. Our experiments show a significant improvement of the accuracy when adding contextual information.

However, there is room for improvement. Due to time constraints and the low quality of the ROWIKI corpus, we only used the PAR dataset, which is smaller. Better quality data could be obtained using an approach similar to the one proposed by Petrică *et al*. (2014), where the texts were filtered based on diacritics distributions. This way, both datasets could be filtered and used for training, and the model could benefit from the higher number and quality of training examples.

The results show that contextual information helped, but the improvement obtained compared to a neural network that uses the character encoder and the current word is not extremely large. One future improvement could consist in adding an attention mechanism based on the current word, used to better select what is relevant from the context for the current letter. Another observed issue is the imbalance of diacritics in texts (e.g., "a" is much more frequent than "ă" or "â"), which could be solved by using a weighted loss that takes into account these distributions.

Acknowledgments

This work was supported by a grant of the Romanian National Authority for Scientific Research and Innovation, CNCS – UEFISCDI, project number PN-III 54PCCDI / 2018, INTELLIT – "Prezervarea și valorificarea patrimoniului literar românesc folosind soluții digitale inteligente pentru extragerea și sistematizarea de cunoștințe".

References

Bojanowski, P., Grave, E., Joulin, A., & Mikolov, T. (2016). Enriching Word Vectors with Subword Information. *EMNLP* (Vol. 91). https://doi.org/1511.09249v1

Brown, P. F., Desouza, P. V, Mercer, R. L., Pietra, V. J. Della, & Lai, J. C. (1992). Class-based n-gram models of natural language. *Computational Linguistics*, 18(4), 467–479.

Graves, A., & Schmidhuber, J. (2005). Framewise phoneme classification with bidirectional LSTM networks. In *Proceedings of the International Joint Conference on Neural Networks* (Vol. 4, pp. 2047–2052). IEEE. https://doi.org/10.1109/IJCNN.2005.1556215

Kingma, D., & Ba, J. (2015). Adam: a method for stochastic optimization. *3rd International Conference for Learning Representations*. https://doi.org/10.1109/ICCE.2017.7889386

Petrică, L., Cucu, H., Buzo, A., & Burileanu, C. (2014). A Robust Diacritics Restoration System Using Unreliable Raw Text Data. In *Spoken Language Technologies for Under-Resourced Languages*, (May), 14–16.

Tufiş, D., & Ceauşu, A. (2008). DIAC+: A Professional Diacritics Recovering System. *Proceedings of the Sixth International Language Resources and Evaluation (LREC'08),* (May). Retrieved from http://www.lrec-conf.org/proceedings/ lrec2008/

Ungurean, C., Burileanu, D., Popescu, V., Negrescu, C., & Dervis, A. (2008). Automatic Diacritic Restoration for a TTS-based E-mail Reader Application. *Bulletin*, Series C, 70, 3–12. Retrieved from http://www.scientificbulletin.upb.ro/rev_docs_arhiva/ full24826.pdf