Lightweight diacritics restoration for V4 languages

Bálint Csanády, András Lukács

Eötvös Loránd University, Institute of Mathematics
Department of Computer Science, AI Research Group

csbalint@protonmail.ch, lukacs@cs.elte.hu

Abstract. Diacritics restoration became a ubiquitous task in the Latin-alphabet-based English-dominated Internet language environment. In this article, we describe a small footprint 1D convolution-based approach, which works on character-level. The model even runs locally in a web browser, and surpasses the performance of similarly sized models. We evaluate our model on the languages of the Visegrád Group, with emphasis on Hungarian.

Keywords: diacritics restoration, 1D convolutional neural network, A-TCN, small footprint, V4, Hungarian

1 Introduction

Many languages have alphabets in which some characters are derived from other characters using *diacritical marks*. For example, most European languages have alphabets which are derived from the Latin alphabet in this way. The goal of *diacritics restoration* is to restore diacritical marks, given an input text which does not contain (or only partially contains) the proper diacritical marks. Diacritics restoration is a practical task on the Internet, where it is still apparent that computers were built initially with the base Latin alphabet in mind.

Diacritics restoration is a useful preprocessing tool for many NLP tasks, e.g. question answering (Abdelnasser et al., 2014). On the other hand, diacritic restoration is an important tool for language revitalization (Galla, 2009), thus contributing to linguistic diversity, the literacy of endangered languages, and the maintenance of their digital presence (Kornai, 2013). This can be effectively supported by language-independent diacritic restoration tools. Nevertheless, we consider only living languages where large corpora based on the Latin alphabet are available (omitting such exciting cases as Celtic languages or poetry marking). Diacritical marks appear in certain Slavic languages (Czech, Slovak, Polish), some Finno-Ugric languages (Finnish, Hungarian, Latvian), Romanian, Turkish, and, most intensively, in Vietnamese.

Approaches to diacritics restoration have evolved from rule-based and statistical solutions to the application of machine learning models (Yarowsky, 1999). The latter approach can be broken down into solutions using fixed or learned

representations. All solutions with learned representations seem to be based on neural networks connected to the models used in NLP, lately recurrent neural networks models (Hucko and Lacko, 2018) being replaced by transformers (Laki and Yang, 2020; Náplava et al., 2021). In such cases, models used for machine translation are often used to correct diacritical marks (Novák and Siklósi, 2015). Another approach is to consider diacritics restoration as a sequence labeling problem where convolutional neural networks and recurrent neural networks such as BiLSTM-s (Náplava et al., 2018) can be applied. We apply a fast language-independent method with small footprint for automatic diacritic restoration using a neural architecture based on 1D convolutions, the so called Acausal Temporal Convolutional Networks (A-TCN). Models based on A-TCN have a comparable performance to BiLSTM-s (Alqahtani et al., 2019).

We focus on the languages of the Visegrád Group (V4), with emphasis on Hungarian. In Hungarian the characters which can receive diacritical marks are exactly the vowels (e.g. u $\mapsto$ {u,ú,ü,ű}). For Hungarian, the current state of the art is reported by Laki and Yang (2020) and is achieved by neural machine translation. Our main contribution is a lightweight model, which can even be run locally in web browsers, allowing client-side inference. We compared our model with Hunaccent (Ács and Halmi, 2016); both models have a similar size of around 10MB. Our approach outperformed Hunaccent by a large margin.

2 Methods

We approached the diacritics restoration problem as a character-sequence labeling task. We chose the output labels as the set of characters in each alphabet. An alternative way to model the restoration task could have been to produce the possible diacritical marks (including the empty mark) on the output side. Our choice is motivated by the expectation that the model's scope could be expanded, and it might be able to correct other local errors in the text, not only missing diacritical marks.

The neural network architecture we considered for sequence labeling are Temporal Convolutional Networks (TCNs). TCNs are a generic family of models, with notable examples including WaveNet (Oord et al., 2016). TCNs are 1D fully convolutional networks, where the convolutions are causal, and at time t output is produced in each layer by the convolution of input elements from time $t-1$ and earlier (Bai et al., 2018). To increase the effective size of the convolutional windows, dilated convolutions can be used (Yu and Koltun, 2015). The network is built with dilation factors which increase exponentially by the depth of the network (Fig. 1). This ensures that the window on the input sequence, which the network can utilize for the inference of a given label, also increases exponentially.

TCNs also contain residual connections (He et al., 2016). A residual block involves a series of transformations, the result of which are then added to the input. The transformation consists of a dilated convolution followed by a normal-

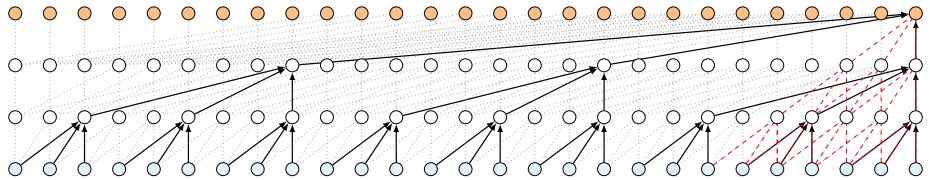

Fig. 1: TCN architecture (kernel size: 3, dilation factors: 1,3,9). Red dashed: without dilation.

ization layer, activation function, and dropout. This is repeated b times (typically $b = 2$).

TCNs work well for applications where information flow from the future is not permitted. For diacritics restoration it is essential to incorporate future context as well as past context. To achieve this, we have to slightly modify the base TCN architecture as seen in Fig. 2. The modified TCN architecture is called acausal TCN, or A-TCN for short (Alqahtani et al., 2019).

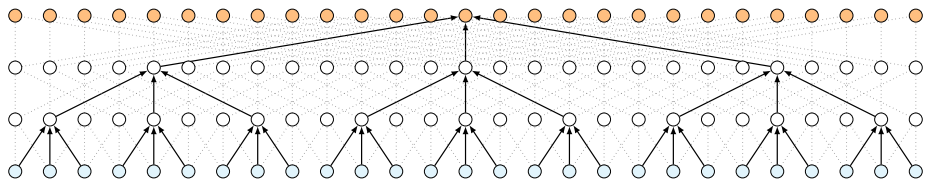

Fig. 2: A-TCN architecture (kernel size: 3, dilation factors: 1,3,9).

3 ONNX.js compatibility

Our model can be converted to ONNX (Bai et al., 2017), a cross-platform neural network format. ONNX.js (Wang et al., 2018) is a JavaScript library, which can run models in ONNX format, which in turn makes it possible to run our model in the browser. Inference happens on the clients device, making use of the clients graphical processor with the help of WebGL.

Converting a model to work with ONNX.js requires some care. For example LSTMs are not supported yet, and even 1D convolutions had to be simulated with 2D convolutions. Although they are mathematically equivalent, we found that training the model in PyTorch is much more effective if we reduce a spatial dimension to 1 in the 2D convolution (instead reducing the feature size to 1).

Another difficulty is that the model allows arbitrary input lengths, but in ONNX.js the first inference fixes the input sequence length. The solution is to dynamically reload the model. If the input is longer than the current limit, the model is reloaded with double length.

Our demonstration web page with diacritics restoration for the V4 languages is available at `https://web.cs.elte.hu/~csbalint/diacritics/demo.html`.

4 Datasets

4.1 LINDAT

We can generate the data for diacritics restoration in a self-supervised fashion. Grammatically correct sentences from the target language provide the annotated data, which means that the removal of the diacritical marks provide the input.

For training on the V4 languages, we used the datasets provided by Náplava et al. (2018). We will refer to these as LINDAT[1]. We cleaned up the datasets by removing the sentences containing exotic characters (if the length of the character was more than 1 after applying the unidecode Python function, we considered the character exotic). We also cut off all the sentences to a maximum length of 500.

	Train			Dev		
Language	Sequences	Avg.seq.len.	Characters	Sequences	Avg.seq.len.	Characters
Cze	946 k	107.6	101.8 M	14.5 k	114.4	1.66 M
Hun	1 287 k	108.3	139.3 M	14.7 k	120.7	1.77 M
Pol	1 063 k	116.2	123.6 M	14.8 k	121.3	1.80 M
Svk	609 k	106.7	65.1 M	14.9 k	114.7	1.71 M

Table 1: Statistics of the datasets by Náplava et al. (2018) we used for training.

4.2 Hungarian Webcorpora

We also considered two additional corpora for Hungarian. We trained a model on the dataset built from Hungarian Webcorpus 2.0 (HunWeb2)[2] by Nemeskey (2020), and evaluated the models on the dataset built from Hungarian Webcorpus (HunWeb1)[3] by Halácsy et al. (2004). Each corpus contains a large collection of Hungarian text documents. To prepare the data, we extracted sentences from each document until we reached a length limit of 500. After extracting the sequences, we random sampled them, and created the train-dev cuts (Table 2). In the case of HunWeb2, we used the "2017-2018" part of the Common Crawl subcorpus. Not splitting up the documents ensures that the train and dev data do not correlate too much, as we have found that there can be a lot of repeated sentences within one document.

[1] `https://lindat.mff.cuni.cz/repository/xmlui/handle/11234/1-2607`
[2] `https://hlt.bme.hu/en/resources/webcorpus2`
[3] `http://mokk.bme.hu/resources/webcorpus/`

	Train			Dev		
Corpus	Sequences	Avg. seq. len.	Characters	Sequences	Avg. seq. len.	Characters
HunWeb1	96.1 k	405.5	39.0 M	12.0 k	406.8	4.89 M
HunWeb2	73.7 k	498.7	36.8 M	9.2 k	498.4	4.59 M

Table 2: Statistics of the Hungarian Webcorpus based datasets used for training.

We analyzed the datasets in terms of word ambiguity. We first cleaned up the train data by removing all sequences not containing enough diacritical marks. Let us call the base of a word the word we get, after removing the diacritical marks from it. We categorized a base unambiguous, if the data contained only one diacritized version of it. Similarly, a word was categorized ambiguous, if multiple diacritized forms existed in the data. The ambiguity of a word may be due to grammar, or to an error in the corpus, even after the cleanup step performed to decrease the number of such false positives. Unambiguous words can be diacritized with a dictionary-based approach.

In Table 3 we see the statistics related to ambiguous and unambiguous words in the datasets. We see that there are similar amounts of ambiguous and unambiguous words in the data, but the ambiguous words come from a much smaller set of bases. We considered the metric of ambiguous word accuracy versus word accuracy, and we saw that at the beginning of the training ambiguous word accuracy was higher, but as the model improved the two metrics switched places. This can be explained by Table 3.

			Unambiguous		Ambiguous		Ratio	
Corpus	Sequences	Words	Words	Bases	Words	Bases	Words	Bases
HunWeb1	73.3 k	13.0 M	7.50 M	635 k	5.51 M	16.6 k	1.36	38.2
HunWeb2	69.8 k	14.0 M	7.87 M	746 k	6.08 M	20.2 k	1.29	36.9

Table 3: Word statistics of the filtered Webcorpus based datasets for Hungarian.

5 Experimental Setup

In terms of model architecture we used the following hyperparameters. The character embedding dimension was set to 50. After the embedding, the vectors are upsampled to dimension 250, which is the channel size. The network contains 4 residual block layers with dilation factors of 1,2,4, and 8, respectively. Each block contains 2 convolutional layers, each followed by batch normalization, ReLU, and spatial dropout layers with a rate of 0.2, respectively. The convolutions have a kernel size of 5. In the convolutions, zero padding is used to ensure that the output is the same length as the input.

We augmented the training data before each epoch in the training. If a character had a diacritical mark, we removed it with a probability of 80%. In real world use, the absence of diacritical marks might only be partial.

The model implemented in PyTorch was trained on 4 Nvidia RTX 2080 Ti graphics cards. Training took approximately one day per model. Our model is available at `https://github.com/aielte-research/Diacritics_restoration`.

6 Results

We calibrated our model lightweight enough to be converted to HTML. For Hungarian we took Hunaccent (Ács and Halmi, 2016) as a direct comparison. Hunaccent is decision tree based, and it shares our goal to implement a small footprint restorator. Moreover, it also can be run locally in a browser. To ensure a fair comparison, we set up our model to have a size similar to the 12.1 MB of the trained model of Hunaccent. The raw ONNX file of our trained model is 9.65 MB and our demo HTML file is 12.88 MB. The HTML file contains the ONNX file as a Base64 encoded string.

Model	Train data	Eval data	Character	Vowel	Alpha-word	Sequence
Hunaccent	HunWeb1	HunWeb1	0.9874	0.9619	0.9129	0.0868
		HunWeb2	0.9838	0.9511	0.8942	0.0055
		LINDAT	0.9834	0.9509	0.8934	0.2732
A-TCN	HunWeb2	HunWeb1	**0.9968**	**0.9903**	**0.9778**	**0.4223**
		HunWeb2	**0.9965**	**0.9893**	**0.9764**	**0.3246**
		LINDAT	0.9952	0.9876	0.9715	0.6596
A-TCN	LINDAT	HunWeb1	0.9945	0.9834	0.9617	0.2724
		HunWeb2	0.9940	0.9819	0.9596	0.1352
		LINDAT	**0.9975**	**0.9925**	**0.9824**	**0.7890**

Table 4: Accuracy comparison for Hungarian diacritic restoration between the baseline (Hunaccent) and our model (A-TCN). We used the pretrained Hunaccent model provided by the authors. The numbers indicate the results on non-augmented, fully dediacritized input.

Compared to the baseline, our model achieved significantly better results in all of the metrics we considered (Table 4). *Character* accuracy measures the ratio of the correct characters in the output. *Important character* accuracy is measured on characters for which diacritical marks are applicable. In the case of the Hungarian language, these characters are the vowels. *Alpha-word* accuracy is measured by the ratio of the correct words in the output, where only the words are considered which contain at least one alphabetical character. *Sequence* accuracy is measured by the ratio of flawless sequences, which is inversely proportional to the average length of the sequences.

In Table 5 we can see the effect of the augmentation. Hunaccent performs better on data where all of the diacritics are missing, while our model performs slightly better, but almost the same when we leave about 20% of the diacritical marks.

	Hunaccent		A-TCN	
Eval. type	Vowel	Alpha-word	Vowel	Alpha-word
Augmented	0.9400	0.8705	**0.9908**	**0.9795**
Non-augmented	**0.9511**	**0.8942**	0.9893	0.9764

Table 5: Performance comparison of Hunaccent and A-TCN (augmented training) on the augmented and the non-augmented task (HunWeb2).

For Hungarian we compared the datasets in terms of performance of the trained models (Table 4). Our tests indicate that our HunWeb2-based dataset yields better results. This is partly due to the fact that as shown in Table 6, the model seems to overfit when trained on the dataset provided by Náplava et al. (2018). The train and dev data are likely not independent enough.

	Train		Dev	
Dataset	Vowel	Alpha-word	Vowel	Alpha-word
HunWeb2	0.9924	0.9828	0.9893	0.9764
LINDAT	0.9922	0.9816	0.9925	0.9824

Table 6: Train and dev accuracies of the same model trained on HunWeb2 and LINDAT. The model seems to overfit on LINDAT.

Language	Character	Important Character	Alpha-word	Sentence
Cze	0.9966	0.9944	0.9783	0.7344
Hun	0.9975	0.9925	0.9824	0.7890
Pol	0.9987	0.9970	0.9903	0.8810
Svk	0.9966	0.9947	0.9784	0.7420

Table 7: Accuracies on V4 languages trained on the dataset provided by Náplava et al. (2018).

The performance of our model on the V4 languages can be seen in Table 7. The results indicate that our model is language-agnostic and works well for

its size for multiple different languages. The alpha-word accuracies are slightly below the ones reported by Náplava et al. (2018).

7 Error Analysis

The confusion matrix of the A-TCN model (trained and evaluated on HunWeb2) can be seen in Table 9. Even though our model can output every character in the vocabulary at each position, the only confused characters were vowels with the same base. We included precision (PPV) an recall (TPR) in the table. The overall weighted F1 score for vowels is 0.990.

Actual Vowel

Predicted Vowel

	o	ó	ö	ő	PPV
o	151k	686	523	251	0.990
ó	849	39.1k	58	201	0.973
ö	399	32	38.2k	118	0.986
ő	397	145	123	35.1k	0.981
TPR	0.982	0.978	0.982	0.984	

	u	ú	ü	ű	PPV
u	42.3k	180	188	55	0.990
ú	253	11.6k	18	29	0.975
ü	170	12	22.1k	19	0.991
ű	93	26	65	7618	0.976
TPR	0.988	0.982	0.988	0.987	

	a	á	PPV
a	337k	2452	0.993
á	2146	131k	0.984
TPR	0.994	0.982	

	e	é	PPV
e	376k	3386	0.991
é	2264	125k	0.982
TPR	0.994	0.974	

	i	í	PPV
i	164k	286	0.998
í	478	23.7k	0.980
TPR	0.997	0.988	

Table 8: Vowel confusion matrix

We performed a small-scale manual evaluation of the A-TCN model. After inference on the evaluation dataset, we selected 100 random errors to be manually classified in the following categories.

1. The error is false positive due to a corpus error, the model output is the correct form.
2. The input is ambiguous at word level, but the model output does not fit grammatically in the sentence.
3. The output is not wrong grammatically, but does not agree with the wider context of the text.
4. Though the model output and the ground truth are different, they both are adequate.
5. The error occurred in a named entity.
6. None of the above, true error.

According to the manual evaluation (Table 9) around 50% of the errors belong to categories 2 and 6. We can reasonably expect to reduce these errors by increasing the size of the model, both to increase the perceived vocabulary of the

Error class	Ratio
1. Corrected Input	0.11
2. Word Ambiguous Input	0.27
3. Grammar Ambiguous Input	0.11
4. Context Ambiguous Input	0.12
5. Named Entity	0.17
6. Incorrect Output	0.22

Table 9: Error classes of the Hungarian A-TCN model.

model, and also to enable a larger context window to draw information from, as some of the grammatical context is likely too far away for the model with the current hyperparameters. Named entity errors are a bit harder to reduce, since they are often less frequent or more ambiguous in the corpus. Errors due to ambiguous input in terms of grammar could be harder to reduce as they sometimes require more insight.

8 Conclusion

We presented a model of small size based on 1D convolutional neural network for diacritic restoration. Furthermore, the model is ONNX.js compatible, so it can even be used in a web browser. The model was evaluated for V4 languages and it performed similarly well compared to other larger models and outperformed models of similar size. In the case of the Hungarian language, we considered three data sets and studied the generalizing power of the model between data sets.

Further research is needed to expand the applicability of the model to correcting general errors in texts, including spelling. We plan to try a larger, but still browser-compatible model, and at least in the case of Hungarian, on a larger training data set.

Acknowledgments

The research was partially supported by the Ministry of Innovation and Technology NRDI Office within the framework of the Artificial Intelligence National Laboratory Program, the Hungarian National Excellence Grant 2018-1.2.1-NKP-00008 and the grant EFOP-3.6.3-VEKOP-16-2017-00002.

The second author was supported by project "Application Domain Specific Highly Reliable IT Solutions" implemented with the support provided from the National Research, Development and Innovation Fund of Hungary, financed under the Thematic Excellence Programme TKP2020-NKA-06 (National Challenges Subprogramme) funding scheme.

We managed our experiments using `https://neptune.ai`. We would like to thank the Neptune Labs team for providing us access to the team version and the technical support.

We would like to thank Dániel Varga for drawing our attention to the problem of lightweight diacritics reconstruction, and Judit Ács for her help with NLP issues.

Bibliography

Abdelnasser, H., Ragab, M., Mohamed, R., Mohamed, A., Farouk, B., El-Makky, N.M., Torki, M.: Al-bayan: an arabic question answering system for the holy quran. In: Proceedings of the EMNLP 2014 Workshop on Arabic Natural Language Processing (ANLP). pp. 57–64 (2014)

Ács, J., Halmi, J.: Hunaccent: Small footprint diacritic restoration for social media. In: Normalisation and Analysis of Social Media Texts (NormSoMe) Workshop Programme. p. 1 (2016)

Alqahtani, S., Mishra, A., Diab, M.: Efficient convolutional neural networks for diacritic restoration. In: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP). pp. 1442–1448 (2019)

Bai, J., Lu, F., Zhang, K., et al.: ONNX: Open Neural Network Exchange. `https://github.com/onnx/onnx` (2017)

Bai, S., Kolter, J.Z., Koltun, V.: An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. arXiv preprint arXiv:1803.01271 (2018)

Galla, C.K.: Indigenous language revitalization and technology: From traditional to contemporary domains. Indigenous language revitalization: Encouragement, guidance & lessons learned pp. 167–182 (2009)

Halácsy, P., Kornai, A., Németh, L., Rung, A., Szakadát, I., Trón, V.: Creating open language resources for hungarian. In: Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC'04) (2004)

He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016)

Hucko, A., Lacko, P.: Diacritics restoration using deep neural networks. In: 2018 World Symposium on Digital Intelligence for Systems and Machines (DISA). pp. 195–200. IEEE (2018)

Kornai, A.: Digital language death. PloS one 8(10), e77056 (2013)

Laki, L.J., Yang, Z.G.: Automatic diacritic restoration with transformer model based neural machine translation for east-central european languages. In: ICAI. pp. 190–202 (2020)

Náplava, J., Straka, M., Straková, J.: Diacritics restoration using BERT with analysis on czech language. The Prague Bulletin of Mathematical Linguistics No. 116, pp. 27–42 (2021)

Náplava, J., Straka, M., Straňák, P., Hajic, J.: Diacritics restoration using neural networks. In: Proceedings of the eleventh international conference on language resources and evaluation (LREC 2018) (2018)

Nemeskey, D.M.: Natural Language Processing Methods for Language Modeling. Ph.D. thesis, Eötvös Loránd University (2020)

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

Oord, A.v.d., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., Kavukcuoglu, K.: WaveNet: A generative model for raw audio. arXiv preprint arXiv:1609.03499 (2016)

Wang, Y., Seshadri, H., et al.: ONNX.js. `https://github.com/microsoft/onnxjs` (2018)

Yarowsky, D.: A comparison of corpus-based techniques for restoring accents in Spanish and French text. In: Natural language processing using very large corpora, pp. 99–120. Springer (1999)

Yu, F., Koltun, V.: Multi-scale context aggregation by dilated convolutions. arXiv preprint arXiv:1511.07122 (2015)