Proceedings of the 13th Conference on Language Resources and Evaluation (LREC 2022), pages 4253–4259
Marseille, 20-25 June 2022

Dilated Convolutional Neural Networks for Lightweight Diacritics Restoration

Bálint Csanády, András Lukács
Department of Computer Science, AI Research Group
Institute of Mathematics, Eötvös Loránd University
Pázmány Péter stny. 1/c, 1036 Budapest, Hungary
csbalint@protonmail.ch, lukacs@cs.elte.hu

Abstract

Diacritics restoration has become a ubiquitous task in the Latin-alphabet-based English-dominated Internet language environment. In this paper, we describe a small footprint 1D dilated convolution-based approach which operates on a character-level. We find that neural networks based on 1D dilated convolutions are competitive alternatives to solutions based on recurrent neural networks or linguistic modeling for the task of diacritics restoration. Our approach surpasses the performance of similarly sized models and is also competitive with larger models. A special feature of our solution is that it even runs locally in a web browser. We also provide a working example of this browser-based implementation. Our model is evaluated on different corpora, with emphasis on the Hungarian language. We performed comparative measurements about the generalization power of the model in relation to three Hungarian corpora. We also analyzed the errors to understand the limitation of corpus-based self-supervised training.

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

1. Introduction

Many languages, including most European languages, have alphabets where some of the characters are derived from base characters using *diacritical marks*. 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 the absence of diacritical marks can still be prominent.

Diacritics restoration is a useful preprocessing step for many NLP tasks, e.g. question answering (Abdelnasser et al., 2014). On the other hand, diacritics 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 diacritics 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 diacritics restoration using a neural architecture based on 1D convolutions, the so called Acausal Temporal Convolutional Networks (A-TCN). Models based on A-TCN have comparable performance to BiLSTM-s (Alqahtani et al., 2019), which is also confirmed by our present research.

Our experiments are focused on the Hungarian language. 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 outperforms Hunaccent by a large margin, and performs comparable to the recurrent model in Náplava et al. (2018). Moreover our method is also language-agnostic.

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 the alphabet of the language. An alternative way to model the restoration task could have been to produce the possible diacrit-

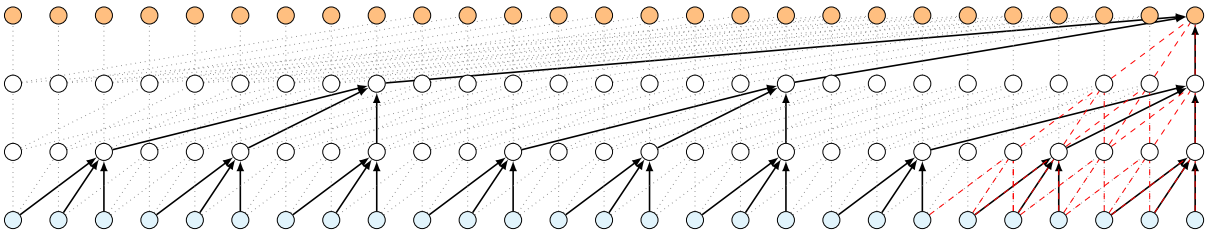

Figure 1: TCN architecture (kernel size: 3, dilation factors: 1,3,9). Red dashed: without dilation.

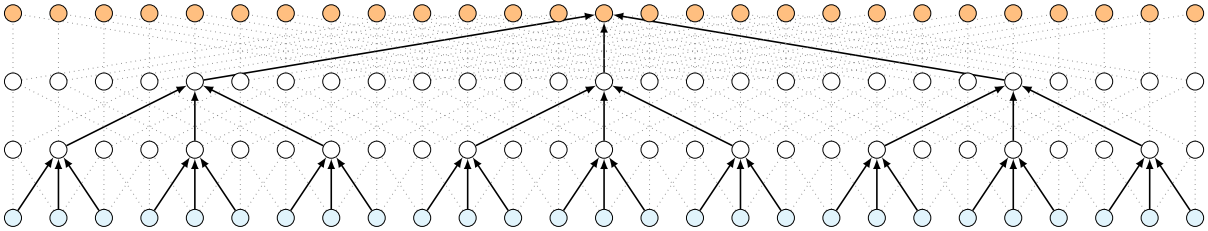

Figure 2: A-TCN architecture (kernel size: 3, dilation factors: 1,3,9).

ical 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 receptive field 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 normalization layer, activation function, and dropout. This is repeated b times (often $b = 2$).

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, the base TCN architecture has to be slightly modified as seen in Fig. 2. The modified TCN architecture is called acausal TCN, or A-TCN for short (Alqahtani et al., 2019).

3. ONNX compatibility

Our model is compatible with ONNX (Bai et al., 2017), a cross-platform neural network format. ONNX serves as an intermediary format, it can be imported by ONNX.js (Wang et al., 2018), a JavaScript library, which 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 have to be simulated with 2D convolutions. Although they are mathematically equivalent, we found that training the model in PyTorch is much more effective if spatial dimension is reduced to 1 in the 2D convolution (instead of 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.

Since we submitted the paper, ONNX.js got replaced by ONNX Runtime Web (Wang and others, 2021). ONNX Runtime Web has fixed some of the above issues, such as the lack of LSTM and 1D convolution support. The current version of our project uses ONNX Runtime Web.

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

4. Datasets

The data for training diacritics restoration can be generated 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.

We used the datasets provided by Náplava et al. (2018) for training on four Central European languages (Czech, Hungarian, Polish and Slovak). We will refer to these datasets as LINDAT. The datasets were cleaned

Language	Train			Dev		
	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	1287 k	108.3	139.3 M	14.7 k	120.7	1.77 M
Pol	1063 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 LINDAT datasets.

Corpus	Sequences	Words	Unambiguous		Ambiguous		Ratio	
			Words	Bases	Words	Bases	Words	Bases
HunWeb1	649 k	35.7 M	18.2 M	979 k	17.6 M	29.3 k	1.032	33.5
HunWeb2	6.16 M	403.0 M	118.6 M	4.51 M	284.4 M	179.2 k	0.417	25.2

Table 2: Word ambiguity statistics of the Webcorpus-based datasets for Hungarian.

up by removing the sentences containing exotic characters (we considered the character exotic if applying the unidecode Python function on the character yielded a string more than one character long). We also cut off all the sentences to a maximum length of 500. Table 1 shows the statistics of the datasets.

Two additional corpora were considered for training and evaluating on Hungarian. A model was trained on the dataset built from Hungarian Webcorpus 2.0, hereinafter referred to as HunWeb2 (Nemeskey, 2020). The models were also evaluated on the dataset built from the earlier Hungarian Webcorpus (HunWeb1) 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 3). We also cleaned up the data by removing all sequences not containing enough diacritical marks, as some part of the corpus contains sentences wich partially or completely lack the proper diacritical marks. In the case of HunWeb2, we used the "2017-2018" part of the Common Crawl subcorpus. We decided not to split up the documents to multiple chunks or sentences, as we have found that there can be a lot of repeated sentences within one document, which would skew the evaluation metrics.

Dataset		Seqs	Avg. seq. len.	Chars
HunWeb1	Dev	10 k	409.3	4.09 M
HunWeb2	Train	6.16 M	474.0	2.92 G
	Dev	10 k	474.1	4.74 M

Table 3: Statistics of the additional datasets for Hungarian.

5. Word Ambiguity

Ambiguous words pose a limit to dictionary-based solutions for solving diacritics restoration. For example we have several options when we want to diacritize the Hungarian word *koros*: *körös* (containing circles or a Hungarian river), *kóros* (sick), *kórós* (containing weeds), *koros* (old) and even *kőrös* (geographical name).

We analyzed the Hungarian Webcorpus-based datasets in terms of apparent word ambiguity. 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 was performed to decrease the number of such false positives. Unambiguous words can be diacritized with a dictionary-based approach.

In Table 2 we see the statistics related to ambiguous and unambiguous words in the datasets. There are similar amounts of ambiguous and unambiguous words in the data (though as the number of sequences increases, the chance for false positives and rare variants also increases), but the ambiguous words come from a much smaller set of bases.

The metric of ambiguous word accuracy is dependent on which words are classified as ambiguous, which makes it unsuitable to compare the performance of different models. Nevertheless when compared to (alpha) word accuracy, we saw that ambiguous word accuracy was higher at the beginning of the training, but as the model improved the two metrics switched places. This might be explained by the numbers in Table 2, as the ambiguous bases are harder to correctly diacritize on the long run, but there are substantially more unambiguous bases, and the model might need more examples to memorize them.

6. 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.

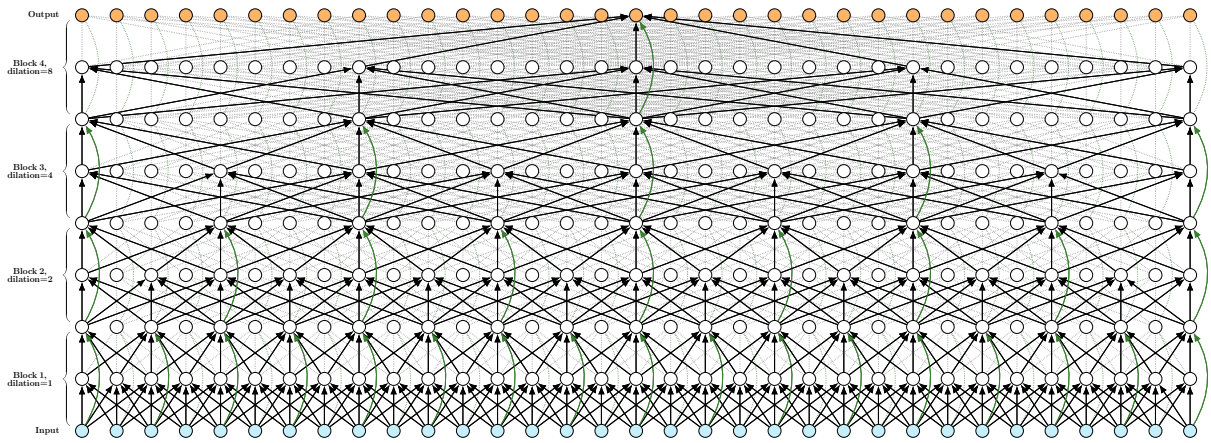

Figure 3: Illustration of the hyperparameter setup used in the experiments (number of blocks: 4, block size: 2, dilation base: 2, window size: 5).

Due to this upsampling, the embedding dimension can be chosen flexibly. Upsampling is done by an 1×1 convolution after permuting the dimensions from $50 \times 1 \times n$ to $1 \times 50 \times n$, where n is the sequence length. This effectively means that the input is concatenated by itself 5 times over, allowing for a scalar multiplier for each copy. This simpler approach was chosen instead of the usual upsampling for performance reasons.

The network contains 4 residual block layers with dilation factors of 1,2,4, and 8, respectively (Fig. 3). 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. When trained on HunWeb2 we limited each epoch to 100000 sequences. The batches contain the same 200 sequences every time (augmentation is applied each time a batch is accessed), but in each epoch we train on a random 500 batches.

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`.

7. Results

We calibrated our model size lightweight enough so that it runs efficiently in a browser. For Hungarian

Model	Train data	Eval data	Character	Vowel	Alpha-word	Sequence
Copy		HunWeb1	0.8979	0.6929	0.4768	0.0000
		HunWeb2	0.9020	0.7042	0.4997	0.0000
		LINDAT	0.9043	0.7134	0.5093	0.0269
Hunaccent	HunWeb1	HunWeb1	0.9886	0.9657	0.9207	0.0398
		HunWeb2	0.9855	0.9563	0.9049	0.0087
		LINDAT	0.9834	0.9509	0.8934	0.2732
Dictionary	HunWeb2	HunWeb1	0.9960	0.9879	0.9772	0.3511
		HunWeb2	0.9965	0.9894	0.9791	0.3329
		LINDAT	0.9942	0.9831	0.9698	0.6551
A-TCN	HunWeb2	HunWeb1	**0.9987**	**0.9961**	**0.9907**	**0.6574**
		HunWeb2	**0.9988**	**0.9964**	**0.9916**	**0.6424**
		LINDAT	0.9974	**0.9941**	**0.9862**	**0.8087**
A-TCN	LINDAT	HunWeb1	0.9950	0.9850	0.9649	0.2683
		HunWeb2	0.9945	0.9834	0.9621	0.1556
		LINDAT	**0.9975**	0.9925	0.9824	0.7890

Table 4: Accuracy comparison for Hungarian diacritics restoration between the baseline (Hunaccent) and the 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.

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. We considered the pretrained Hunaccent model provided by the authors Ács and Halmi (2016). 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 10.11 MB and our demo HTML file is 13.49 MB. The HTML file contains the ONNX file as a Base64 encoded string.

Compared to Hunaccent, our model achieved significantly better results in all of the metrics we considered. We also measured the performance of a simple dictionary based method. From the HunWeb2-based training data, we created a word dictionary containing each word base we encountered. The most frequent diacritization was chosen for each base. Náplava et al. (2018) reports an alpha word accuracy of 0.9902 on Hungarian (LINDAT). Their model is LSTM-based and has a reported size of around 30 MB.

Table 4 contains the results. We also added a line called *Copy*, which measures the accuracy what we get if we simply copy the input without adding any diacritical marks. *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 remove only about 80% of the diacritical marks.

Model	Eval. task	Vowel	Alpha-word
Hunaccent	Aug.	0.9441	0.8785
	Non-aug.	**0.9563**	**0.9048**
A-TCN	Aug.	**0.9967**	**0.9925**
	Non-aug.	0.9964	0.9915

Table 5: Performance comparison of Hunaccent (normal training) 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 size difference between the training data. When trained on a smaller size HunWeb2-based dataset, the model still performed better. This might be explained by Table 6, as the model seems to overfit when trained on LINDAT. The train and dev data are likely not independent enough.

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

The performance of our model on four Central European languages from the LINDAT corpus 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).

Lang.	Chr.	Imp. Chr.	α-word	Seq.
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 languages trained on the LINDAT dataset.

8. Error Analysis

The confusion matrix of the A-TCN model (trained and evaluated on HunWeb2) can be seen in Table 8. 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 0.996.

We performed a small-scale manual evaluation of the A-TCN model. After inference on the evaluation dataset, we selected 500 random errors to be manually classified in the following categories.

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

According to the manual evaluation (Table 9) around 30% of the errors belong to categories 3 and 7. We can

Predicted Vowel

Actual Vowel

	o	ó	ö	ő	TPR
o	156 k	302	164	118	0.996
ó	297	42.8 k	28	77	0.991
ö	166	28	42.6 k	92	0.993
ő	67	68	46	38.3 k	0.995
PPV	0.997	0.991	0.994	0.993	

	u	ú	ü	ű	TPR
u	43.2 k	91	54	23	0.996
ú	82	12.4 k	3	14	0.992
ü	72	21	23.6 k	46	0.994
ű	19	15	24	8263	0.993
PPV	0.996	0.990	0.997	0.990	

	a	á	TPR
a	345 k	856	0.998
á	826	138 k	0.994
PPV	0.998	0.994	

	e	é	TPR
e	391 k	926	0.998
é	1313	132 k	0.990
PPV	0.997	0.993	

	i	í	TPR
i	169 k	220	0.999
í	159	25.0 k	0.994
PPV	0.999	0.991	

Table 8: Vowel confusion matrix

reasonably expect to reduce these errors by increasing the size of the model, both to increase the perceived vocabulary of the 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.

Error class	Ratio
1. Corpus error	0.062
2. Corrected corpus error	0.128
3. Word Ambiguous Input	0.186
4. Grammar Ambiguous Input	0.158
5. Context Ambiguous Input	0.124
6. Named Entity	0.256
7. Incorrect Output	0.126

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

9. Conclusion

We presented a neural network model of small size based on 1D convolutions for diacritics restoration. Furthermore, the model is ONNX.js (ONNX Runtime Web) compatible, so it can even be used in a web browser. The model was evaluated on four Central European 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 train a larger, but still browser-compatible model, and plan to further improve the model architecture, and consider more diacritics-heavy languages.

10. 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 Thematic Excellence Programme TKP2021-NKTA-62.

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 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.

11. Bibliographical References

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

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

Alqahtani, S., Mishra, A., and Diab, M. (2019). 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)*, pages 1442–1448.

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

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

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

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

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

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

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

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

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

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

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

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

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

Wang, Y. et al. (2021). ONNX Runtime Web. `https://github.com/microsoft/onnxruntime/tree/master/js/web`.

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

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

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

12. Language Resource References

Halácsy, P., Kornai, A., Németh, L., Rung, A., Szakadát, I. and Trón, V. (2004). *Hungarian Webcorpus*. `http://mokk.bme.hu/resources/webcorpus/`.

Náplava, J., Straka, M., Hajič, J. and Straňák, P. (2018). *Corpus for training and evaluating diacritics restoration systems*. `https://lindat.mff.cuni.cz/repository/xmlui/handle/11234/1-2607`.

Nemeskey, D. M. (2020). *Hungarian Webcorpus 2.0*. `https://hlt.bme.hu/en/resources/webcorpus2`.