Diacritics Restoration Using Neural Networks

Jakub Náplava, Milan Straka, Pavel Straňák, Jan Hajič
Institute of Formal and Applied Linguistics
Charles University, Faculty of Mathematics and Physics
Malostranské náměstí 25, Prague, Czech Republic
{naplava,straka,stranak,hajic}@ufal.mff.cuni.cz

Abstract

In this paper, we describe a novel combination of a character-level recurrent neural-network based model and a language model applied to diacritics restoration. In many cases in the past and still at present, people often replace characters with diacritics with their ASCII counterparts. Despite the fact that the resulting text is usually easy to understand for humans, it is much harder for further computational processing. This paper opens with a discussion of applicability of restoration of diacritics in selected languages. Next, we present a neural network-based approach to diacritics generation. The core component of our model is a bidirectional recurrent neural network operating at a character level. We evaluate the model on two existing datasets consisting of four European languages. When combined with a language model, our model reduces the error of current best systems by 20% to 64%. Finally, we propose a pipeline for obtaining consistent diacritics restoration datasets for twelve languages and evaluate our model on it. All the code is available under open source license on https://github.com/arahusky/diacritics_restoration.

Keywords: neural networks, diacritics, diacritics generation, error correction

1. Introduction

When writing emails, tweets or texts in certain languages, people for various reasons sometimes write without diacritics. When using Latin script, they replace characters with diacritics (e.g. c with acute or caron) by the underlying basic character without diacritics. Practically speaking, they write in ASCII. We offer several possible reasons for this phenomenon:

- Historically, many devices offered only an English keyboard and/or ASCII encoding (for example older mobile phones and SMS).
- Before Unicode became widespread, there were encoding problems among platforms and even among programs on the same platform, and many people still have this in mind.
- Even though text encoding is rarely a problem any more and all modern devices offer native keyboards, some problems persist. In situations of frequent code switching between English and a language with a substantially different keyboard layout, it is very hard to touch type in both layouts. It is much easier to type both languages using the same layout, although one of them without proper diacritics.
- In some circumstances typing with diacritical marks is significantly slower than using just basic latin characters. The most common example is on-screen keyboards on mobile devices. These keyboards do not include the top row (numerical on US English), so languages that use that row for accented characters are much slower to type. Naturally, users type without explicit accents and rely on the auto-completion systems. However, these systems are usually simple, unigram-based, and based on the word form ambiguity for a given language (cf. Table 1), which introduces many errors. Postponing the step of diacritics generation would be beneficial both for typing speed and accuracy.
- For example in Vietnamese, the language with most diacritics in our data (cf. Table 1), both the above problems are very pronounced: Because Vietnamese uses diacritical marks to distinguish both tones (6) and quality of vowels (up to 3), a vowel can have (and often has) 2 marks. This need to provide efficient typing of very many accented characters led to the invention of systems like unikey.org that allow a user to type all the accented characters using sequences of basic letters. For instance to typeset “đường” a user types “dduwowngf”. While this system elegantly solves the problems above with switching keyboard layouts and missing top row of keys, it requires a special software package and it still results in 9 keystrokes to type 5 characters. That is why typing without accents in informal situations like emails or text messages is still common and system for efficient generation of diacritics would be very useful.

Typical languages where approximately half of the words contain diacritics are Czech, Hungarian or Latvian. Nevertheless, as we discuss in Sections 2 and 3, diacritics restoration (also known as diacritics generation or diacritization) is an active problem also in many languages with substantially lower diacritics appearance.

Current approaches to restoration of diacritics (see Section 3) are mostly based on traditional statistical methods. However, in recent years, deep neural networks have shown remarkable results in many areas. To explore their capabilities, we propose a neural based model in Section 4 and evaluate its performance on two datasets in Section 5.

In Section 6, we describe a way to obtain a consistent multilingual dataset for diacritics restoration and evaluate our model on them. The dataset can be downloaded from the published link. Finally, Section 7 concludes this paper with a summary of outcomes.

Language	Words with diacritics	Word error rate of dictionary baseline
Vietnamese	88.4%	40.53%
Romanian	31.0%	29.71%
Latvian	47.7%	8.45%
Czech	52.5%	4.09%
Slovak	41.4%	3.35%
Irish	29.5%	3.15%
French	16.7%	2.86%
Hungarian	50.7%	2.80%
Polish	36.9%	2.52%
Swedish	26.4%	1.88%
Portuguese	13.3%	1.83%
Galician	13.3%	1.62%
Estonian	19.7%	1.41%
Spanish	11.3%	1.28%
Norwegian-Nynorsk	12.8%	1.20%
Turkish	30.0%	1.16%
Catalan	11.1%	1.10%
Slovenian	14.0%	0.97%
Finnish	23.5%	0.89%
Norwegian-Bokmaal	11.7%	0.79%
Danish	10.2%	0.69%
German	8.3%	0.59%
Croatian	16.7%	0.34%

Table 1: Analysis of percentage of words with diacritics and the word error rate of a dictionary baseline. Measured on UD 2.0 data, using the *uninames* method, and CoNLL 17 UD shared task raw data for dictionary. Only words containing at least one alphabetical character are considered.

2. Diacritics Restoration in Languages using Latin Script

Table 1 presents languages using (usually some extended version of) a Latin script. Employing UD 2.0 (Nivre et al., 2017) plain text data, we measure the ratio of words with diacritics, omitting languages with less than 5% of words with diacritics. In eleven of the languages, at least every fifth word contains diacritics; in another eleven languages, at least every tenth word does.

Naturally, high occurrence of words with diacritics does not imply that generating diacritics is an ambiguous task. Consequently, we also evaluate word error rate of a simple dictionary baseline to diacritics restoration: according to a large raw text corpora we construct a dictionary of the most frequent variant with diacritics for a given word without diacritics, and use the dictionary to perform the diacritics restoration.

Table 1 presents the results. We utilized the raw corpora by Ginter et al. (2017) released as supplementary material of CoNLL 2017 UD Shared task (Zeman et al., 2017), which contain circa a gigaword for each language, therefore providing a strong baseline. For nine languages, the word error rate is larger than 2%, and eight more languages have word error rate still above 1%. We conclude that even with a very large dictionary, the diacritics restoration is a challenging task for many languages, and better method is needed.

Letter	Hex code	Unicode name
ø	00F8	LATIN SMALL LETTER O *WITH STROKE*
ł	0142	LATIN SMALL LETTER L *WITH STROKE*
đ	0111	LATIN SMALL LETTER D *WITH STROKE*
ʂ	0282	LATIN SMALL LETTER S *WITH HOOK*
ç	00E7	LATIN SMALL LETTER C *WITH CEDILLA*
š	0161	LATIN SMALL LETTER S *WITH CARON*

Table 2: Unicode characters that cannot be decomposed using NFD (first 4 lines), and those that can. The suffix of the name removed by the *uninames* method is show in italics. As we can see, the strucutre of names is identical, so the method works for all of these characters.

2.1. Methods of Diacritics Stripping

Although there is no standard way of stripping diacritics, a commonly used method is to convert input word to NFD (The Unicode Consortium, 2017, Normalization Form D) which decomposes composite characters into a base character and a sequence of combining marks, then remove the combining marks, and convert the result back to NFC (Unicode Normalization Form C). We dub this method *uninorms*.

We however noted that this method does not strip diacritics for some characters (e.g. for đ and ł).[1] We therefore propose a new method *uninames*, which operates as follows: In order to remove diacritics from a given character, we inspect its name in the Unicode Character Database (The Unicode Consortium, 2017). If it contains a word *WITH*, we remove the longest suffix starting with it, try looking up a character with the remaining name and yield the character if it exists.

The method is illustrated in Table 2, which presents four characters that do not decompose under NFD, but whose diacritics can be stripped by the proposed method.

As shown in Table 3, the proposed *uninames* method recognizes all characters the *uninorms* method does, and some additional ones. Therefore, we employ the *uninames* method to strip diacritics in the paper.

3. Related Work

One of the first papers to describe systems for automatic diacritics restoration is a seminal work by Yarowsky (1999), who compares several algorithms for restoration of diacritics in French and Spanish. Later, models for diacrization in Vietnamese (Nguyen and Ock, 2010), Czech (Richter et al., 2012), Turkish (Adali and Eryiğit, 2014), Ara-

[1] What constitutes a "diacritic mark" is a bit of a problem. On one hand not all characters with a graphical element added to a letter letter contain diacritics, e.g. ¥ (symbol of Japanese Yen) or Đ/ð (Icelandic "eth"). On the other end of the spectrum we have clear diacritics with Unicode canonical decomposition into a letter and a combining mark. Between these clear borders there are the characters that do not have a unicode decomposition, but their names still indicate they are latin letters with some modifier and often they are used the same as characters that do have decomposition. E.g. Norwegian/Danish ø is used exactly like ö in Swedish, it is just an orthographic variation. However while the latter has canonical decomposition in Unicode, the first does not. This is why we opted to treat these characters also as "letters with diacritics".

Lowercased letters with diacritics

í 15.7%	ů 2.5%	ş	ī	ņ	ō
á 11.7%	ú 1.6%	ā	ē	ū	ŕ
é 9.8%	ö 1.5%	ñ	ą	ï	Ĺ
ě 6.6%	ă 1.3%	**ł**	ò	ļ	ţ
ä 6.0%	**ø** 1.1%	ć	ż	ń	ì
č 5.5%	à 0.9%	ň	ğ	ù	ó
ř 5.2%	ç 0.9%	â	ś	û	ḥ
ž 4.9%	ü 0.8%	õ	ď	ű	ă
ý 4.5%	ã 0.8%	ť	ô	ķ	ĺ
š 4.4%	è 0.6%	ê	**đ**	ģ	ŕ
ó 3.2%	î 0.5%	ę	ő	ź	ά
å 2.9%	ț 0.5%	ş	ľ	ë	**ş**

Table 3: Most frequent characters with diacritics from data listed in Table 1, together with their relative frequency. The bold characters are recognized only using the *uninames* method.

bic (Azmi and Almajed, 2015) Croatian, Slovenian, Serbian (Ljubešic et al., 2016), and many other languages were published. The system complexity ranges from simplest models, that for each word apply its most frequent translation as observed in the training data, to models that incorporate language models, part-of-speech tags, morphological and many other features. One of the most similar model to ours is a system by Belinkov and Glass (2015) who used recurrent neural networks for Arabic diacritization.

4. Model Architecture

The core of our model (see Figure 1) is a bidirectional recurrent neural network, which for each input character outputs its correct label (e.g. its variant with diacritics). The input and output vocabularies contain a special out-of-alphabet symbol.

The input characters are embedded, i.e. each character in the input sentence is represented by a vector of d real numbers. The character embeddings are initialized randomly and updated during training.

The embeddings are fed to a bidirectional RNN (Graves and Schmidhuber, 2005). The bidirectional RNN consists of two unidirectional RNNs, one reading the inputs in standard order (forward RNN) and the other in reverse order (backward RNN). The output is then a sum of forward and backward RNN outputs. This way, bidirectional RNN is processing information from both preceding and following context. The model allows an arbitrary number of stacked bidirectional RNN layers.

The output of the (possibly multilayer) bidirectional RNN is at each time step reduced by an identical fully connected layer to an o-dimensional vector, where o is the size of the output alphabet. A nonlinearity is then applied to these reduced vectors.

Finally, we use a softmax layer to produce a probability distribution over output alphabet at each time step.

The loss function is the cross-entropy loss summed over all outputs.

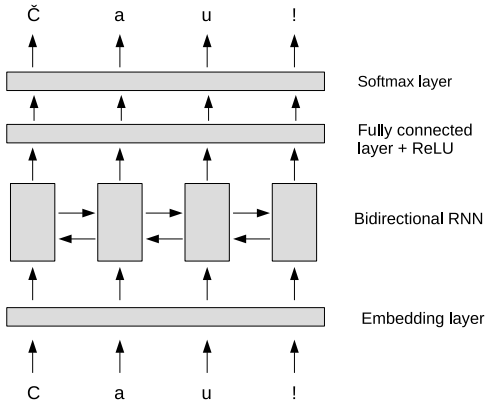

Figure 1: Visualisation of our model.

4.1. Residual connections

The proposed model allows an arbitrary number of stacked RNN layers. The model with multiple layers allows each stacked layer to process more complex representation of current input. This naturally brings potential to improve accuracy of the model.

As stated by (Wu et al., 2016), simple stacking of more RNN layers works only up to a certain number of layers. Beyond this limit, the model becomes too difficult to train, which is most likely caused by vanishing and exploding gradient problems (Pascanu et al., 2013). To improve the gradient flow, (Wu et al., 2016) incorporate residual connections to the model. To formalize this idea, let RNN_i be the i-th RNN layer in a stack and $x_0 = (\mathit{inp}_1, \mathit{inp}_2, \ldots, \mathit{inp}_N)$ input to the first stacked RNN layer RNN_0. The model we have proposed so far works as follows:

$$o_i, c_i = \mathit{RNN}_i(x_i)$$
$$x_{i+1} = o_i$$
$$o_{i+1}, c_{i+1} = \mathit{RNN}_{i+1}(x_{i+1}),$$

where o_i is the output of i-th stacked RNN layer and c_i is a sequence of its hidden states. The model with residual connections between stacked RNN layers then works as follows:

$$o_i, c_i = \mathit{RNN}_i(x_i)$$
$$x_{i+1} = o_i + x_i$$
$$o_{i+1}, c_{i+1} = \mathit{RNN}_{i+1}(x_{i+1})$$

4.2. Decoding

For inference we use a left-to-right beam search decoder combining the neural network and the language model likelihoods. The process is a modified version of standard beam search used by Xie et al. (2016) for decoding sequence-to-sequence models.

Let b denote the beam size. The hypotheses in the beam are initialized with the b most probable first characters. In each step, all beam hypotheses are extended with b most probable variants of the respective character, creating b^2 hypotheses. These are then sorted and the top b of them are kept.

Whenever a space is observed in the output, all affected hypotheses are reranked using both the RNN model output

probabilities and language model probabilities. The hypothesis probability in step k can be computed as:

$$P(y_{1:k}|x) = (1-\alpha)\log P_{NN}(y_{1:k}|x) + \alpha \log P_{LM}(y_{1:k}),$$

where x denotes the input sequence, y stands for the decoded symbols contained within current hypothesis, P_{NN} and P_{LM} are neural network and language model probabilities and the hyper-parameter α determines the weight of the language model. To keep both $\log P_{NN}$ and $\log P_{LM}$ terms within a similar range in the decoding, we compute the $\log P_{NN}$ as the mean of output token log probabilities and additionally normalize P_{LM} by the number of words in the sequence.
To train the language model as well as to run it, we use the open-source KenLM toolkit (Heafield, 2011).

5. Experiments

To compare performance of our model with current approaches, we perform experiments using two existing datasets. The first one, created by Ljubešic et al. (2016), consists of Croatian, Serbian and Slovenian sentences from three sources: Wikipedia texts, general Web texts and texts from Twitter. Since Web data are assumed to be the noisiest, they are used only for training. Wikipedia and Twitter testing sets should then cover both standard and non-standard language. The second evaluation dataset we utilize consists of Czech sentences collected mainly from newspapers, thus it covers mostly standard Czech.

5.1. Training and Decoding Details

We used the same model configuration for all experiments. The bidirectional RNN has 2 stacked layers with residual connections and utilizes LSTM units (Hochreiter and Schmidhuber, 1997) of dimension 300. Dropout (Srivastava et al., 2014) at a rate of 0.2 is used both on the embedded inputs and after each bidirectional layer. All weights are initialized using Xavier uniform initializer (Glorot and Bengio, 2010).
The vocabulary of each experiment consists of top 200 most occurring characters in a training set and a special symbol (<UNK>) for unknown characters.
To train the model, we use the Adam optimizer (Kingma and Ba, 2014) with learning rate 0.0003 and a minibatch size of 200. Each model was trained on a single GeForce GTX 1080 Ti for approximately 4 days. After training, the model with the highest accuracy on the corresponding development set was selected.
To estimate the decoding parameter α, we performed an exhaustive search over [0,1] with a step size of 0.1. The parameter was selected to maximize model performance on a particular development set. All results were obtained using a beam width of 8.

5.2. Croatian, Serbian and Slovenian

The original dataset contains training files divided into Web, Twitter and Wikipedia subsets. However, Ljubešic et al. (2016) showed that concatenating all these language-specific sets for training yields best results. Therefore, we used only concatenated files for training the models for each of three languages in our experiments. The training files contain 17 968 828 sentences for Croatian, 11 223 924 sentences for Slovenian and 8 376 810 sentences for Serbian.[2] All letters in the dataset are lowercased.
To remove diacritics from the collected texts, Ljubešic et al. (2016) used a simple script that replaced four letters (ž, ć, č, š) with their ASCII counterparts (z, c, c, s), and one letter (đ) with its phonetic transcription (dj). This results in the input and target sentences having different length. Since our model requires both input and target sentences to have the same length, additional data preprocessing was required before feeding the data into the model: we replace all occurrences of the *dj* sequence in both input and target sentences by a special token, and replace it back to *dj* after decoding.
The results of the experiment with comparison to previous best system (*Lexicon*, *Corpus*) are presented in Table 4. The *Lexicon* method replaces each word by its most frequent translation as observed in the training data. The *Corpus* method extends it via log-linear model with context probability. These methods were evaluated by Ljubešic et al. (2016) and the *Corpus* method is to the best of our knowledge state-of-the-art system for all three languages. System accuracy is measured, similarly to the original paper, on all words which have at least one alphanumerical character.
We incorporated the same language models as used by the authors of the original paper. There are two points in the results we would like to stress:

- Our system with language model reduces error by more than 30% on wiki data and by more than 20% on tweet data. Moreover, our model outperforms the current best system on wiki data even if it does not incorporate the additional language model, which makes the model much smaller (~30MB instead of several gigabytes of the language model).
- Diacritics restoration problem is easier on standard language (wiki) than on non-standard data (tweets). This has, in our opinion, two reasons. First, the amount of wiki data in the training sets is substantially higher than the amount of non-standard data (tweets). This makes the model fit more standard data. Second, due to lower language quality in Twitter data, we suppose that the amount of errors in the gold data is higher.

5.3. Czech

The second experiment we conducted is devoted to diacritics restoration in Czech texts. To train both the neural network and language models, we used the SYN2010 corpus (Křen et al., 2010), which contains 8 182 870 sentences collected from Czech literature and newspapers. To evaluate the model, PDT3.0 (Hajič et al., 2018) testing set with 13 136 sentences originating from Czech newspapers is used. Both the training and testing set, thus, contain mainly standard Czech. For language model training, we consider only those {2,3,4,5}-grams that occurred at least twice, and use default KenLM options.
Table 5 presents a comparison of our model performance with Microsoft Office Word 2010, ASpell, CZACCENT (Rychlý, 2012) and Korektor (Richter et al., 2012), the latter being the state-of-the-art system of diacritics restoration

[2]The Serbian dataset is based on a Latin script.

System	wiki			tweet		
	hr	sr	sl	hr	sr	sl
Lexicon	0.9936	0.9924	0.9933	0.9917	0.9893	0.9820
Corpus	0.9957	0.9947	0.9962	0.9938	0.9917	0.9912
Our model	0.9967	0.9961	0.9970	0.9932	0.9939	0.9882
Our model + LM	**0.9973**	**0.9968**	**0.9974**	**0.9951**	**0.9944**	**0.9930**
Error reduction	36.81%	39.74%	30.45%	21.62%	32.14%	20.77%

Table 4: Results obtained on Crotian (HR), Serbian (SR) and Slovenian (SL) Wikipedia and Twitter testing sets. Note that the word accuracy presented in the table is not measured on all words, but only on words having at least one alphanumerical character.

System	Word accuracy
Microsoft Office Word 2010 (*)	0.8910
ASpell (*)	0.8839
Lexicon	0.9527
CZACCENT	0.9607
Corpus	0.9713
Korektor	0.9861
Our model	0.9887
Our model + LM	**0.9951**
Error reduction	64.75%

Table 5: Comparison of several models of restoration of diacritics for Czech. The (*) denotes reduced test data (see text).

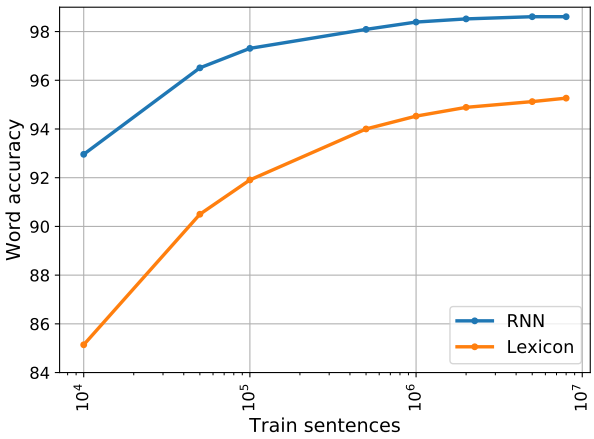

Figure 2: Comparison of RNN and Lexicon performance with varrying training data size.

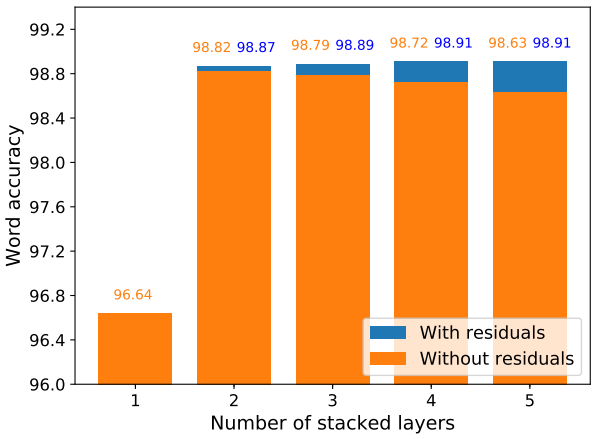

Figure 3: Effect of using residual connections with respect to the number of stacked layers.

for Czech. Note that evaluation using Microsoft Office Word 2010 and ASpell was performed only on the first 746 (636) sentences, because it requires user interaction (confirming the suggested alternatives).

As the results show, models that are not tuned to the task of diacritics restoration perform poorly. Our model combined with a language model reduces the error of the previous state-of-the-art system by more than 60%; our model achieves slightly higher accuracy than Korektor even if no language model is utilized.

5.3.1. Ablation Experiments

One of the reasons why deep learning works so well is the availability of large training corpora. This motivates us to explore the amount of data our model needs to perform well. As Figure 2 shows, the RNN model trained on 50 000 random sentences from SYN2010 corpus performs better on the PDT3.0 testing set than the *Lexicon* baseline trained on full SYN2010 corpus. Further, up to 5M sentences the performance of the RNN model increases with the growing training set size. We do not observe any performance difference between the RNN model trained on 5M and 8M sentences.

The second ablation experiment examines the effect of residual connections. We trained models with 2, 3, 4 and 5 stacked layers each with and without residual connections. We also trained a simple model with 1 bidirectional layer without residual connections. The results of this experiment are presented in Figure 3. Apart from the big difference in word accuracy between the model with 1 layer and other models, we can see that models with residual connections perform generally better than when no residual connections are incorporated. It is also evident that when more layers are added in stack, performance of models without residual connections deteriorates while performance of models with additional residual connections does not.

6. New Multilingual Dataset

As discussed in the preceding sections, diacritics restoration is an active field of research. However, to the best of our knowledge, there is no consistent approach to obtaining datasets for this task. When a new diacritics restoration sys-

tem is published, a new dataset is typically created both for training and testing. This makes it difficult to compare performance across systems. We thus propose a new pipeline for obtaining consistent multilingual datasets for the task of diacritics restoration.

6.1. Dataset

As the data for diacritics restoration need to be clean, we decided to utilize Wikipedia for both development and testing sets. Because there may be not enough data to train potential diacritics restoration systems on Wikipedia texts only, we further decided to create training sets from the (general) Web. We chose two corpora for this task: the W2C corpus (Majliš, 2011) with texts from Wikipedia and the general Web in 120 languages, and the CommonCrawl corpus with language annotations generated by Buck et al. (2014) with a substantially larger amount of general Web texts in more than 150 languages.

To create training, development and testing data from the Wikipedia part of the W2C corpus, its data are first segmented into sentences, these are then converted to lowercase and finally split into disjoint training, development and testing set. The split was performed in such a way that all three sets consist of sentences collected from whole articles rather then being randomly sampled across all articles. Each testing set consists of 30 000 sentences, development set of approximatelly 15 000 sentences and the rest of the data are preserved for training set.

The pipeline for creating additional training data from the CommonCrawl corpus starts with the removal of invalid UTF8 data and Wikipedia data. These filtered data are then segmented into sentences and converted to lowercase. Since these data come from general Web and may be noisy (e.g. contain sentences with missing diacritics), only those sentences that have at least 100 characters and contain at least a certain amount of diacritics are preserved. The constant determining the minimum amount of diacritics is language specific and is derived from Table 1. Finally, sentence intersection with existing development and testing set is removed and maximally ten similar sentences are preserved in the training data. Since both baseline methods (*Lexicon* and *Corpus*) require data to be word tokenized, all texts are also word tokenized.

The dataset was created for 12 languages (see Table 6), where the additional training sets were generated from the *2017_17* web crawl. Complete dataset can be downloaded from `http://hdl.handle.net/11234/1-2607`.

6.2. Experiments

We train and evaluate our model on the created dataset and compare its performance to two baseline methods. The same model hyperparameters as described in Section 5.1 are used, except for the RNN cell dimension, which is 500.

Training was performed in two phases. First, each language specific model was trained on particular CommonCrawl (Web) training set for approximately four days. Then, each model was fine-tuned with a smaller learning rate 0.0001 on respective Wikipedia training set for three more days. Finally, as all models seemed to be continuously improving on the development sets, we took the last model checkpoints for evaluating.

Both baseline methods and language models were trained on concatenation of Wikipedia and CommonCrawl training data. For language model training, we considered only those {2,3,4,5}-grams that occurred at least twice, and used default KenLM options.

To measure model performance, modified word error accuracy is used. The alpha-word accuracy considers only words that consist of at least one alphabetical character, because only these can be potentially diacritized. The testing set results of *Lexicon* and *Corpus* baselines, as well as of our models before and after fine-tuning, and with a language model are presented in Table 6.

As results show, our model outperforms both baselines even if no language model is used. Moreover, incorporation of the language model helps the model perform better as well as does model fine-tuning. Without fine-tuning, all models but the Romanian outperform baselines. We suspect that the reason why the Romanian model before fine-tuning performs worse than the *Corpus* method is that non-standard Web data differ too much from standard data from Wikipedia. It is also an interesting fact that the biggest error reduction is at Vietnamese and Romanian which seem to be most difficult for both baseline methods.

7. Conclusion

In this work, we propose a novel combination of recurrent neural network and a language model for performing diacritics restoration. The proposed system is language agnostic as it is trained solely from parallel corpora of texts without diacritics and diacritized texts. We test our system on two existing datasets comprising of four languages, and we show that it outperforms previous state-of-the-art systems. Moreover, we propose a pipeline for generating consistent multilingual diacritics restoration datasets, run it on twelve languages, publish the created dataset, evaluate our system on it and provide a comparison with two baseline methods. Our method outperforms even the stronger contextual baseline method on the new dataset by a big margin.

Future work includes detailed error analysis, which could reveal types of errors made by our system. Since certain words may be correctly diacritized in several ways given the context of the whole sentence, such error analysis could also set the language specific limit on the accuracy that can be achieved. Further, when designing our multilingual dataset we decided to use testing sets with sentences from Wikipedia articles. This was well motivated as we wanted it to contain sentences with proper diacritics. However, such testing sets contain mainly standard language and are thus worse for comparison of models aiming to generate diacritics for non-standard language. Therefore, we plan to create additional development and testing sets in the future work.

While experimenting with the model on Czech we found out that when it is trained to output instructions (e.g. add caron) instead of letters, it performs better. Future work thus also includes thorough inspection of this behavior when applied to all languages.

Finally, the system achieves better results when a language model is incorporated while inferring. Because the use of an external model both slows down the inferring process and

Language	Wiki sentences	Web sentences	Words with diacritics	Lexicon	Corpus	Our model w/o finetuning	Our model	Our model + LM	Error reduction
Vietnamese	819 918	25 932 077	73.63%	0.7164	0.8639	0.9622	0.9755	0.9773	83.33%
Romanian	837 647	16 560 534	24.33%	0.8533	0.9046	0.9018	0.9799	0.9837	82.96%
Latvian	315 807	3 827 443	39.39%	0.9101	0.9457	0.9608	0.9657	0.9749	53.81%
Czech	952 909	52 639 067	41.52%	0.9590	0.9814	0.9852	0.9871	0.9906	49.20%
Polish	1 069 841	36 449 109	27.09%	0.9708	0.9841	0.9891	0.9903	0.9955	71.64%
Slovak	613 727	12 687 699	35.60%	0.9734	0.9837	0.9868	0.9884	0.9909	44.21%
Irish	50 825	279 266	26.30%	0.9735	0.9800	0.9842	0.9846	0.9871	35.55%
Hungarian	1 294 605	46 399 979	40.33%	0.9749	0.9832	0.9888	0.9902	0.9929	58.04%
French	1 818 618	78 600 777	14.65%	0.9793	0.9931	0.9948	0.9954	0.9971	58.11%
Turkish	875 781	72 179 352	25.34%	0.9878	0.9905	0.9912	0.9918	0.9928	24.14%
Spanish	1 735 516	80 031 113	10.41%	0.9911	0.9953	0.9956	0.9958	0.9965	25.57%
Croatian	802 610	7 254 410	12.39%	0.9931	0.9947	0.9951	0.9951	0.9967	36.92%

Table 6: Results obtained on new multilingual dataset. Note that the alpha-word accuracy presented in the table is measured only on those words that have at least one alphabetical character. The last column presents errror reduction of our model combined with language model compared to the Corpus method.

requires significantly more memory, it would be desirable to train the model in such way that no additional language model is needed. We suspect that multitask learning (e.g. training the model also to predict next/previous letter) may compensate for the absence of a language model.

8. Acknowledgements

The research described herein has been supported by the Grant No. DG16P02R019 of the Ministry of Culture of the Czech Republic. Data has been used from and stored into the repository of LINDAT/CLARIN, a large research infrastructure supported by the Ministry of Education, Youth and Sports of the Czech Republic under projects LM2015071 and CZ.02.1.01/0.0/0.0/16_013/0001781.
Thanks to NGỤY Giang Linh for the information and examples on Vietnamese.

9. Bibliographical References

Adali, K. and Eryiğit, G. (2014). Vowel and diacritic restoration for social media texts. In *Proceedings of the 5th Workshop on Language Analysis for Social Media (LASM)*, pages 53–61.

Azmi, A. M. and Almajed, R. S. (2015). A survey of automatic arabic diacritization techniques. *Natural Language Engineering*, 21(3):477–495.

Belinkov, Y. and Glass, J. (2015). Arabic diacritization with recurrent neural networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2281–2285.

Buck, C., Heafield, K., and Van Ooyen, B. (2014). N-gram counts and language models from the common crawl. In *LREC*, volume 2, page 4.

Ginter, F., Hajič, J., Luotolahti, J., Straka, M., and Zeman, D. (2017). CoNLL 2017 shared task - automatically annotated raw texts and word embeddings. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics, Charles University.

Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 249–256.

Graves, A. and Schmidhuber, J. (2005). Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Networks*, pages 5–6.

Hajič, Jan and Bejček, Eduard and Bémová, Alevtina and Buráňová, Eva and Hajičová, Eva and Havelka, Jiří and Homola, Petr and Kárník, Jiří and Kettnerová, Václava and Klyueva, Natalia and Kolářová, Veronika and Kučová, Lucie and Lopatková, Markéta and Mikulová, Marie and Mírovský, Jiří and Nedoluzhko, Anna and Pajas, Petr and Panevová, Jarmila and Poláková, Lucie and Rysová, Magdaléna and Sgall, Petr and Spoustová, Johanka and Straňák, Pavel and Synková, Pavlína and Ševčíková, Magda and Štěpánek, Jan and Urešová, Zdeňka and Vidová Hladká, Barbora and Zeman, Daniel and Zikánová, Šárka and Žabokrtský, Zdeněk. (2018). *Prague Dependency Treebank 3.5*. Institute of Formal and Applied Linguistics, LINDAT/CLARIN, Charles University.

Heafield, K. (2011). Kenlm: Faster and smaller language model queries. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 187–197. Association for Computational Linguistics.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

Kingma, D. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Křen, M., Bartoň, T., Cvrček, V., Hnátková, M., Jelínek, T., Kocek, J., Novotná, R., Petkevič, V., Procházka, P., Schmiedtová, V., et al. (2010). Syn2010: žánrově vyváženỳ korpus psané češtiny. *Ústav Českého národního korpusu FF UK, Praha*.

Ljubešic, N., Erjavec, T., and Fišer, D. (2016). Corpus-based diacritic restoration for south slavic languages. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016). European Language Resources Association (ELRA)(may 2016)*.

Majliš, M. (2011). W2c–web to corpus–corpora.

Nguyen, K.-H. and Ock, C.-Y. (2010). Diacritics restoration in vietnamese: letter based vs. syllable based model. *PRICAI 2010: Trends in Artificial Intelligence*, pages 631–

636.

Nivre, J., Agić, Ž., Ahrenberg, L., et al. (2017). Universal Dependencies 2.0. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics, Charles University, Prague.

Pascanu, R., Mikolov, T., and Bengio, Y. (2013). On the difficulty of training recurrent neural networks. *ICML (3)*, 28:1310–1318.

Richter, M., Straňák, P., and Rosen, A. (2012). Korektor-a system for contextual spell-checking and diacritics completion. In *COLING (Posters)*, pages 1019–1028.

Rychlý, P. (2012). Czaccent–simple tool for restoring accents in czech texts. *RASLAN 2012 Recent Advances in Slavonic Natural Language Processing*, page 85.

Srivastava, N., Hinton, G. E., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *Journal of machine learning research*, 15(1):1929–1958.

The Unicode Consortium. (2017). *The Unicode Standard, Version 10.0.0*. The Unicode Consortium, Mountain View, CA.

Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., et al. (2016). Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Xie, Z., Avati, A., Arivazhagan, N., Jurafsky, D., and Ng, A. Y. (2016). Neural language correction with character-based attention. *arXiv preprint arXiv:1603.09727*.

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.

Zeman, D., Popel, M., Straka, M., Hajič, J., Nivre, J., Ginter, F., Luotolahti, J., Pyysalo, S., Petrov, S., Potthast, M., Tyers, F., Badmaeva, E., Gökırmak, M., Nedoluzhko, A., Cinková, S., Hajič jr., J., Hlaváčová, J., Kettnerová, V., Urešová, Z., Kanerva, J., Ojala, S., Missilä, A., Manning, C., Schuster, S., Reddy, S., Taji, D., Habash, N., Leung, H., de Marneffe, M.-C., Sanguinetti, M., Simi, M., Kanayama, H., de Paiva, V., Droganova, K., Martínez Alonso, H., Uszkoreit, H., Macketanz, V., Burchardt, A., Harris, K., Marheinecke, K., Rehm, G., Kayadelen, T., Attia, M., Elkahky, A., Yu, Z., Pitler, E., Lertpradit, S., Mandl, M., Kirchner, J., Fernandez Alcalde, H., Strnadova, J., Banerjee, E., Manurung, R., Stella, A., Shimada, A., Kwak, S., Mendonça, G., Lando, T., Nitisaroj, R., and Li, J. (2017). CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*. Association for Computational Linguistics.