International Journal of Engineering and Applied Physics (IJEAP)
Vol. 3, No. 1, January 2023, pp. 653~662
ISSN: 2737-8071

Global and local attention for automatic Arabic text diacritization

Ali Mijlad[1], Yacine El Younoussi[2]
[1,2]SIGL Laboratory, National School of Applied Sciences Tétouan, Abdelmalek Essaadi University, Tétouan, Morocco

Article Info

Article history:

Received Aug 31, 2022
Revised Oct 01, 2022
Accepted Oct 03, 2022

Keywords:

Natural language processing
Encoder-decoder
Attention mechanism
Luong attention
Arabic diacritization

ABSTRACT

Automatic Arabic diacritization is the task to restore diacritic or vowel marks for a non-vocalized Arabic text. This task showed its importance in the natural language processing NLP field and it helps people with specific learning difficulties to access Arabic web content. To tackle the problem, we suggest a letter-based encoder-decoder that uses previous deep learning attention models known as Luong attention. The training of the models knew unstable loss. And, as was expected — from the proposed models — the model that uses local predictive attention achieved the best word and letter error rates. The best-achieved diacritic error rate in the test data is about 26.80%. Nevertheless, the models need improvements in future work.

Corresponding Author:

Ali Mijlad,
SIGL Laboratory National School Of Applied Sciences Tétouan,
Abdelmalek Essaadi University,
Tétouan, Morocco.
Email: a.mijlad@uae.ac.ma

1. INTRODUCTION

Arabic is a Semitic language, which can be classified into three levels: classical Arabic, Modern Standard Arabic MSA, and dialectal Arabic. It follows writing conventions based on consonants and the choice to include diacritics (vowels) or not. When they are included, they can appear above or below the letters. Besides, some reasons why diacritics are omitted are: people with good knowledge of Arabic can infer the vowels in non-diacritized texts, and the speed of typing may decrease if the diacritics are included. Nevertheless, Arabic vowels are important and can be of great help for people starting to learn that language. Additionally, they can enhance the reading and learning abilities of dyslexic learners. As mentioned by Al-Wabil et al.[1], the lack of phonological skills and working memory make an obstacle to learning for Arabic dyslexics. The excessive use of non-diacritized Arabic text on web content excludes that segment of society to access Arabic web content.

Bishara's study[2] shows the liaison between the students with learning disabilities and the orthographic depth of the Arabic language. Among the group of students who learned reading based on the fully diacritized form, they got the best measures of phonological and morphological processing, compared to the group of pupils that learned to read based on partially diacritized texts.

In addition, Arabic diacritics can ameliorate some tasks in the natural language processing NLP field, like text-to-speech TTS. This, in return, is useful to access Arabic web content for visually impaired people by ameliorating — as in the work of Abuali et al.[3] — the screen readers' accuracy. Machine translation, part-of-speech (POS) tagging, text classification, sentiment analysis, and information retrieval, are also tasks that can get influenced by Arabic diacritics. Hence, the importance of automating the Arabic vowels restoration.

To evaluate the performance of an automatic diacritization system, diacritic error rate DER, and word error rate WER are the metrics to use. The former parameter calculates the percentage of characters that have wrong diacritics. The latter returns the ratio of words that have at least one wrong diacritic mark.

Alansary's work[4] is based on rules to solve the problem. For Gal's system[5] it used the hidden Markov model HMM based on the bigram model and the Viterbi algorithm. Elshafei et al.'s work[6] also uses the HMM, after all the distribution of unigrams, bigrams, and trigrams of words are extracted. Ameur et al.[7] proposed a system where Arabic diacritization is done in two steps. First, a word-based HMM restores the diacritics based on the bigram model. Second, for the out-of-vocabulary OOV, a letter-based HMM restores the diacritics based on a 4-gram model. Hadjir et al.'s work[8] also uses the HMM with a Viterbi decoder to restore diacritics.

Rashwan et al.[9] handled the problem by tokenizing the text and applying a memory of context to each token. Then, two parallel deep networks use the words and their contexts. One of those networks retrieves the features, while the other does the Part-of-speech POS tagging. The outputs of those deep nets are fed to another deep network to do the classification.

Belinkov et al.'s work[10] and Abandah et al.[11] used bidirectional recurrent neural networks.

Metwally et al.'s system[12] firstly does the morphological diacritization and the POS tagging, utilizing the HMM. Secondly, their system restores the same things, but this time for the Out-of-vocabulary OOV words. Lastly, the system does the syntactic vowelization based on POS tags, morphological features, and a Conditional Random Fields (CRFs) classifier.

Zayyan et al.[13] tackled the problem using multi-lexical layers composed of word-based n-gram models, and letter-based n-gram models.

Said et al.'s system[14] is a multilayered sequence. Firstly, common mistakes are auto-corrected and accompanied by tokenization. Secondly, the unseen words are extracted accompanied by inferring the morphological features based on rules and a morphological analyzer. Then comes POS tagging based on an HMM. POS helps in combination with rules to infer the case endings (syntactic diacritics). Eventually, the Conditional Random Field (CRF) model deals with the out-of-vocabulary OOV.

Shaalan et al.'s work[15] uses lexicon retrieval that relies on an Arabic lexicon. The system also uses the bigram model. Besides, a Support Vector Machine SVM is used to do the tokenization and POS tag each token. A decision-maker module uses the three outputs —lexicon, bigram, and POS tags— and selects the internal diacritics. The case ending module determines the POS, chunk position, and case endings based on an SVM model.

Shahrour et al.'s system[16] uses MADAMIRA1 as a baseline. They used syntactic features extracted based on the MaltPaser[17]. To retrieve the morpho-syntactic abstraction they used the J48 classifier.

Chennoufi et al.'s system[18] firstly extracts — based on Alkhalil Morpho Sys2 — the morphological characteristics and diacritization forms of each word. Secondly, based on rules it eliminates inappropriate transitions. This is followed by a word-based HMM. Eventually, a letter-based HMM handles the non-diacritized words.

Fashwan et al.'s work[19] restores the diacritics in two steps. One is morphological-wise using uni-morphological and statistical processing, morphological-based rules, and the processing of the undiacritized words (OOV). The second step restores the case endings using syntactic-based rules.

Darwish et al.[20] use a word-based Viterbi algorithm, morphological patterns, the stems, and the diacritizations of named entities based on sequence labeling and transliteration. SVM ranking, linguistic rules, and morphological patterns are used to infer the case endings.

Alqudah et al.[21] use MADAMIRA first, then its outputs that have higher confidence parameters are fed to a BLSTM.

Abbad et al.[22] use a multi-component model to restore diacritics. First, it starts with a preprocessing phase followed by a deep learning component. Then comes a rule-based correction layer, followed by a trigram and bigram model, which solve the case of undiacritized words. Then comes the phase that uses Levenshtein distance for the remaining non-vocalized words.

In this paper, we are going to present an approach to tackle the problem of diacritic restoration. Our approach mainly uses the attention-based encoder-decoder to solve the issue of diacritization at a letter-based level. We will apply to the problem three attention variants — global, local monotonic, and local predictive — proposed by Luong et al.[23] for the translation problem, and we are expecting that the local-predictive-based model will have the best results as it did for the translation task.

[1] https://camel.abudhabi.nyu.edu/madamira/
[2] https://sourceforge.net/projects/alkhalil/

This work gets its novelty, compared to previous work, from the fact that we are comparing three attention-based deep learning models applied to a character-level — each input token (respectively, output token) is a character — diacritization problem because the attention-based models were applied at first to translation problems. This work also shows which of the models to use to solve the diacritization issue, and it gives what kind of ameliorations the preprocessing phase and the models need for future enhancement.

2. RESEARCH METHOD

2.1. Additional Arabic knowledge background

The basic vowels are mainly grouped into four types. Table 1 presents those four types, where each diacritic mark is written with the letter "ب" /b/ to make a good illustration. There are short vowels which are Fathah, Dammah, and Kasrah. Then there is the Tanwin group (nunations) in which every diacritic is written as a double short vowel. When read, the Tanwin is pronounced as a short vowel with an /n/ sound at the end. The third group is composed of Shaddah also known as germination. When written it is present in combination with short vowels, or Tanwin. The last group is composed of Sukun, which is assigned to consonants to indicate the absence of vowels and the near end of a syllable.

In terms of functionality, we can have two diacritic classes [24]. First, there is the core-diacritic class, which plays a morphological or lexical role. This class disambiguates the lexical features of the word; otherwise, it defines the meaning of the word. Second, there is the case ending or the inflectional class that disambiguates the syntactic features of the same lexeme in a sentence.

Table 1: Arabic diacritics with their International Phonetic Alphabet representation (IPA)

Diacritic Mark	Name	Type	Transl.	IPA	Word Position
بَ	Fathah	Short Vowels	a	/a/	Any
بُ	Dammah	Short Vowels	u	/u/	Any
بِ	kasrah	Short Vowels	i	/i/	Any
بًا	Tanwin Fath	Tanwin	F	/an/	End
بٌ	Tanwin Damm	Tanwin	N	/un/	End
بٍ	Tanwin Kasr	Tanwin	K	/in/	End
بّ	Shaddah	Shaddah	~	:	Any
بْ	Sukun	Sukun	o	∅	Any

2.2. Preprocessing

In this work, we used some preprocessing tasks used in the work of Abbad et al.[22]. Only the characters that have a relation to diacritic restoration are kept. The numbers are replaced by 0. The other chars are removed before the vowel restoration process. Then, from that sentence, we get a sequence input and a sequence output. The input sequence contains the mapping to a set of numeric labels representing the letters, while the output sequence contains the mapping to a set of numeric labels representing the eight mentioned diacritics in addition to the empty (absence of diacritic), start, and end tokens. Each input (respectively, output) sequence is padded with zeros to an input max length (respectively, an output max length).

2.3. Global attention mechanism applied to diacritic restoration

Our proposed system is based on the sequence to sequence with the attention mechanism model. We took the problem as a translation problem, where each input token is a letter and each target token is a diacritic mark. The model uses the baseline form of the Luong attention mechanism[23], also known as global attention. The aforementioned attention takes into consideration all the input sequence letters to make predictions. Figure 1 shows how Luong's attention works:

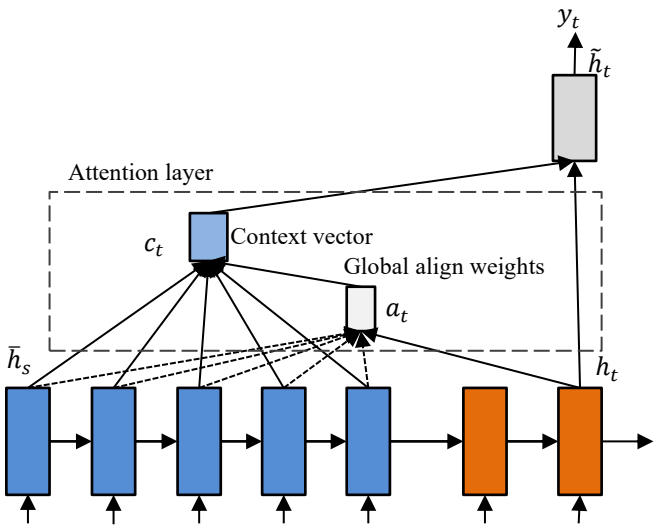

Figure 1: Global attention model by Luong et al.

2.3.1. Encoder :

The encoder is composed of an embedding layer followed by a Gated Recurrent Unit GRU layer. The GRU inputs at each timestep t the embedded representation of x_t and the previously hidden state h_{t-1} to produce the hidden state h_t as follows:

$$h_t = GRU(embed(x_t), h_{t-1}) \quad (1)$$

The encoder output, at the end of the sequence, all the hidden states expressed as $\bar{h}_s = \{h_1, \dots, h_{T_x}\}$, where T_x is the length of the sequence.

2.3.2. Decoder:

The decoding does the following loop up to the end of the sequence:

i. First computes the hidden state h_t based on the previously hidden state h_{t-1} and the previous output y_{t-1}

ii. Align the source hidden state set $\bar{h}_s$ with hidden state h_t to calculate the score as follows:

$$score(h_t, \bar{h}_s) = a(h_t, \bar{h}_s) = V_a Tanh(W_a[h_t, \bar{h}_s]) \quad (2)$$

V_a here is a trainable weight vector, and W_a is a trainable weight matrix.

iii. Calculate the attention weight vector:

$$\alpha_t = Softmax(score(h_t, \bar{h}_s)) \quad (3)$$

iv. Determine the context vector as a weighted average over all source hidden states:

$$c_t = \alpha_t \times \bar{h}_s \quad (4)$$

v. Given W_c a trainable weight matrix, the attentional hidden state is calculated:

$$\tilde{h}_t = tanh\ (W_c[c_t, h_t]) \quad (5)$$

vi. The attentional hidden state $\tilde{h}_t$ is weighted – based on the trainable weight matrix W_s – and input to a $Softmax$ layer to produce the expected char at the time step t :

$$P(y_t / y < t, x) = Softmax(W_s . \tilde{h}_t) \quad (6)$$

2.4. Local attention applied to diacritic restoration

We tackled here the problem by the use of the local forms of the Luong attention mechanism[23]. The local attention variants proposed by Luong et al.[23] concentrate only on a smaller subset of the source positions per target token. Figure 2 shows Luong local attention model:

2.4.1. Encoder:

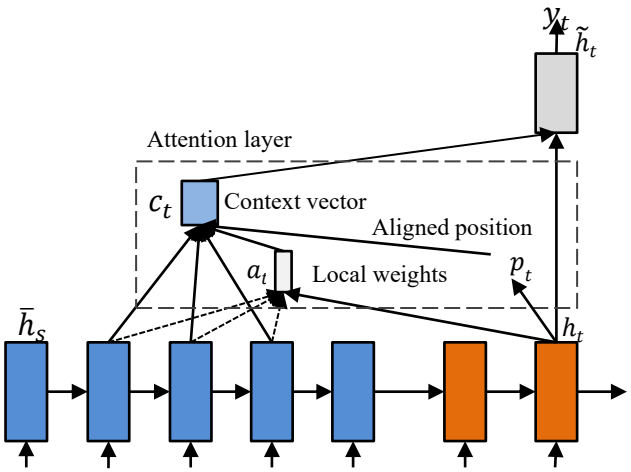

Figure 2: Local attention model by Luong et al.

It does the same thing as the one defined in the global attention model part. The encoder here outputs a set of hidden states $H = \{h_1, \dots, h_{T_x}\}$, where T_x is the length of the sequence.

2.4.2. Decoder:

The decoder in the sequence to sequence with local attention, at each time step it does the following:

i. First, it takes the hidden state h_t at the top layer of the GRU.

ii. The model predicts a single alignment position p_t, this help align the position of the current target t with the source token positions. This alignment can be either monotonic or predictive.

The monotonic alignment assumes that the source and target are monotonically aligned. While the predictive one does not assume that rough alignment and it is calculated as:

$$p_t = S_l . Sigmoid(V_p Tanh(W_p h_t)) \quad (7)$$

Where V_p and W_p are trainable parameters, S_l: is the length of the input sentence.

The alignment p_t — either monotonic or predictive — is used to determine, at each time step, the windows $[p_t - D, p_t + D]$. This gives the encoder hidden states $\bar{h}_s$ to be taken into consideration.

iii. The alignment of the source states $\bar{h}_s$ with the hidden state h_t is calculated as in equation (2).

iv. Calculate attention weights; for local monotonic, it is calculated as in the equation (3), while for the local predictive it uses, in addition, a Gaussian distribution centered at p_t as follows:

$$\alpha_t = Softmax(score(h_t, \bar{h}_s) \times exp\left(-\frac{(s-p_t)^2}{2\sigma^2}\right) \quad (8)$$

Where *s* are integers within the window $[p_t - D, p_t + D]$, and $\sigma = D/2$

v. The context vector is calculated as in equation (4).

vi. Attentional hidden state $\tilde{h}_t$ is calculated as in equation (5).

2.5. Dataset

We used in this work the free-available benchmark dataset used in the comparative study of Fadel et al.[25]. That dataset was derived from different chosen books from the Tashkeela corpus. Then it was — in the same work of Fadel et al.[25] — cleaned, processed, and split into a training set with a percentage of 90%, and the remaining 10% was split in half for the validation and testing sets.

From those three files provided by the mentioned work, we derived three other files, where each input sentence of the model has a maximum number of chars equal to 100. We have chosen that length because of the hardware limitations we have, and the model we have here is a letter-based one.

We extracted more information about the final dataset we used. Table 2 shows more information about the tokens, chars, and sentences in the different datasets. Table 3 shows the percentage of the unseen words in the validation and blind sets in comparison to the train set.

Table 2. Some statistics about the datasets used in this work

	Train set	Validation set	test set
Characters	7622220	377897	393552
Tokens	1177062	58635	60591
Numbers	2301	123	138
Digits	8978	480	507
Arabic words	1015627	50387	52179
Arabic letters	4015335	198913	207293
Diacritics	3431148	170025	177099
Undiacritized forms	75615	12747	13440
Diacritized forms	114792	16507	17233
Number of sentences	121531	6156	6417

Table 3. Percentage of out-of-vocabulary (OOV) in validation and blind sets compared to the training set

	Validation set	Test set
Diacritized OOV Arabic words (%)	18.78	20.03
Undiacritized OOV Arabic words	13.97	15.59

2.6. Evaluation metrics

As in previous works to evaluate our model, we used the diacritic and word error rates. Specifically, the ones defined in the work of Fadel et al.[25].

i. Diacritic error rate (DER) is the rate to attribute wrong diacritics to chars in the text.
ii. Word error rate (WER) is the rate of words that have at worst one wrong vowel mark.

Those metrics — when calculated in Fadel et al.[25]'s work— non-Arabic characters are ignored, which is not the case in some previous works. This can give an unbiased view of the results. Furthermore, you have the choice to include the 'no-diacritic' option, which includes or does not the number of undiacritized letters from the original text. Additionally, the case-ending count can be included or not while computing those metrics.

We also report the Sparse-Categorical-Crossentropy loss of the models during the training.

3. RESULTS AND DISCUSSIONS

3.1. Loss versus epochs:

In figure 3, the global-attention-based model has achieved the lowest validation loss value of 0.11375 at epoch number 2. We can see here that the model has low losses from the first epoch. We set the model to patience for three additional epochs to verify if the validation loss could reach another minimum. After the second epoch, the model overfits the data. The overfitting explanation is that the model could be too complicated for the data. That is why we saved the model at epoch 2.

Figure 4 shows that the local-monotonic-based model has achieved the lowest loss at epoch 5 with a value of 0.0859. Here from the first epoch, the loss values were higher, and the model took 5 epochs to find a minimal validation loss. Here also the early stopping helped the model not to go further in the training and saved the best weights of the model at epoch 5.

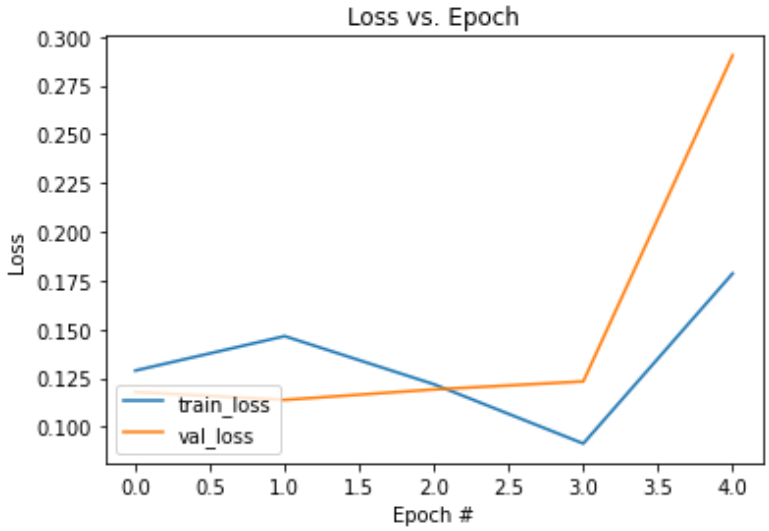

Figure 3. Loss vs epoch for the global-attention-based model

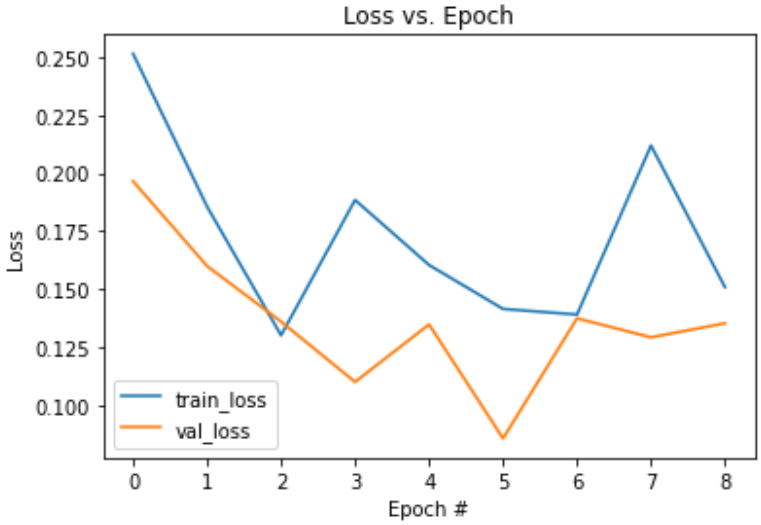

Figure 4: Loss vs epoch for the local-monotonic-attention-based model

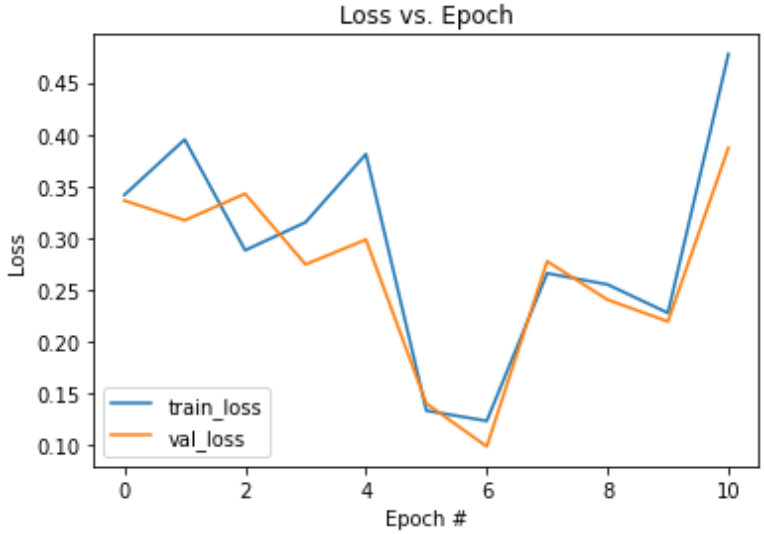

Figure 5. Loss vs epoch for the local-predictive-attention-based model

Figure 5 shows the loss improvement during the training phase of the local-predictive-based model. We saved the model at epoch number 6 where the model reach a low validation loss of value 0.09910.

Generally, the loss of the three models is not going smoothly to the minimum value. Those oscillations of the loss — we suggest — are due to the size of the network, or also the learning rate. Since the models we suggest have a small network size in terms of the number of neurons, this causes a lack of relationship representations; and that makes each batch change the network considerably.

3.2. The Diacritic and word error rates:

Tables 4 and 5 show the diacritic and word error rates achieved on the test set by the three models. The model with local predictive attention performs better on the blind dataset. We also can see that the monotonic performs badly compared to the global attention; this can be explained by the fact the output tokens are not all the time aligned with input tokens. One solution to this could be to augment the window$[p_t - D, p_t + D]$. Due to the instability of the learning of the three models, we can have a hint about the high error rates. We suggest in the next subsection solutions to address the problems of high error rates and unstable loss values.

Table 4: diacritic error rates of the three models

Systems	**Diacritic Error Rate: DER (%)**			
	Including the 'no-diacritic' class		**Excluding the 'no-diacritic' class**	
	With case ending	**Without case ending**	**With case ending**	**Without case ending**
Global-attention-based model	43.90	41.26	49.27	46.11
Local-monotonic-attention-based model	50.10	48.07	52.85	50.30
Local-predictive-attention-based model	**26.80**	**23.94**	**31.59**	**28.17**

Table 5: word error rates of the three models

Systems	Diacritic Error Rate: WER (%)			
	Including the 'no diacritic' class		Excluding the 'no-diacritic' class	
	With case ending	Without case ending	With case ending	Without case ending
Global-attention-based model	73.06	62.92	72.29	62.38
Local-monotonic-attention-based model	73.10	65.69	72.01	64.73
Local-predictive-attention-based model	**57.34**	**44.60**	**56.75**	**44.30**

3.3. Recommendations to ameliorate the systems:

We suggest the following:

- Change the size of the networks: The networks do not have a sufficient number of neurons (we tried a smaller size due to hardware limitations) to represent the relationships. That could be the reason for the oscillating loss.
- Find the right learning rate.
- Add depth to the network, to allow learning more abstract features.
- Try some regularization techniques for the models.
- Modify the output data architecture; we suggest here two ways of representing the output data:

i. Two different output sequences: one sequence is for the presence of germination diacritic, and the other output sequence contains all diacritics except the germination diacritic. This way the two output sequences are roughly aligned with the input sequence.
ii. One output sequence: here the output sequence takes into account that the germination is present in a sequence it is combined with the following diacritic (either a short vowel, a nunation, or the empty vowel). This also will make all the input and output data roughly aligned. That rough alignment can make the local-monotonic attention perform better.

4. Conclusion

Automatic Arabic diacritization is a natural language processing NLP task that aims to restore diacritic marks for Arabic text. It helps novice learners and people with specific learning difficulties to have access to web Arabic web content. It can also be useful for more NLP tasks.

In this paper, we presented three attention mechanisms[23] applied to that task. As expected, the predictive-attention-based model achieved better results compared to other attention-based models. However, due to the hardware limitation, we chose a small dataset and small network sizes. This led to unstable loss during training which caused bad error rates. We are aiming to ameliorate the three systems and compare them to relevant related work.

REFERENCES

[1] A. Al-Wabil, P. Zaphiris, and S. Wilson, "Web design for dyslexics: Accessibility of Arabic content," in Computers Helping People with Special Needs, 2006, vol. 4061, pp. 817–822, doi: https://doi.org/10.1007/11788713_119.

[2] S. Bishara, "The orthographic depth and promotion of students with learning disabilities," Cogent Educ., vol. 6, no. 01, 2019, doi: https://doi.org/10.1080/2331186X.2019.1646384.

[3] B. Abuali and M.-B. Kurdy, "Full Diacritization of the Arabic Text to Improve Screen Readers for the Visually Impaired," Adv. Human-Computer Interact., vol. 2022, 2022, doi: https://doi.org/10.1155/2022/1186678.

[4] S. Alansary, "Alserag: An Automatic Diacritization System for Arabic," in Intelligent Natural Language Processing: Trends and Applications (Studies in Computational Intelligence, 740), 1st ed., K. Shaalan, A. E. Hassanien, and F. Tolba, Eds. Springer, 2018, pp. 523–543.

[5] Y. Gal, "An HMM approach to vowel restoration in Arabic and Hebrew," in Proceedings of the ACL-02 Workshop on Computational Approaches to Semitic Languages, 2002, pp. 1–7, doi: 10.3115/1118637.1118641.

[6] M. Elshafei, H. Al-Muhtaseb, and M. M. Alghamdi, "Statistical methods for automatic diacritization of Arabic text," in proceedings 18th National computer Conference, 2006, pp. 301–306, [Online]. Available: http://almuhtaseb.net/Research/StaMetAutDiaArTex.pdf.

[7] M. H. Ameur, Y. Moulahoum, and A. Guessoum, "Restoration of Arabic Diacritics Using a Multilevel Statistical Model," in 5th International Conference on Computer Computer Science and Its Applications (CIIA), 2015, pp. 181–192, doi: 10.1007/978-3-319-19578-0_15.

[8] I. Hadjir, M. Abbache, and F. Z. Belkredim, "An Approach for Arabic Diacritization," in 24th International Conference on Applications of Natural Language to Information Systems, NLDB 2019, 2019, vol. 11608, pp. 337–344, doi: https://doi.org/10.1007/978-3-030-23281-8_29.

[9] M. Rashwan, A. Al Sallab, H. Raafat, and A. Rafea, "Automatic Arabic diacritics restoration based on deep nets," in Proceedings of the EMNLP 2014 Workshop on Arabic Natural Language Processing (ANLP), 2014, pp. 65–72, doi: 10.3115/v1/W14-3608.

[10] Y. Belinkov and J. Glass, "Arabic Diacritization with Recurrent Neural Networks," in Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, 2015, pp. 2281–2285, doi: 10.18653/v1/D15-1274.

[11] G. A. Abandah, A. Graves, B. Al-Shagoor, A. Arabiyat, F. Jamour, and M. Al-Taee, "Automatic diacritization of Arabic text using recurrent neural networks," Int. J. Doc. Anal. Recognit., vol. 18, no. 2, pp. 183–197, 2015, doi: https://doi.org/10.1007/s10032-015-0242-2.

[12] A. S. Metwally and M. A. Rashwan, "A multi-layered approach for Arabic text diacritization," in 2016 IEEE International Conference on Cloud Computing and Big Data Analysis (ICCCBDA), 2016, pp. 389–393, doi: 10.1109/ICCCBDA.2016.7529589.

[13] A. A. Zayyan, M. Elmahdy, H. Binti Husni, and J. M. Al Ja'am, "Automatic Diacritics Restoration for Dialectal Arabic Text," Int. J. Comput. Inf. Sci., vol. 12, no. 2, pp. 159–165, 2016, doi: 10.21700/ijcis.2016.119.

[14] A. Said, M. El-sharqwi, A. Chalabi, and E. Kamal, "A Hybrid Approach for Arabic Diacritization," in 18th International Conference on Applications of Natural Language to Information Systems, NLDB 2013, 2013, vol. 7934, pp. 53–64, doi: 10.1007/978-3-642-38824-8.

[15] K. Shaalan, H. M. Abo Bakr, and I. Ziedan, "A hybrid approach for building Arabic diacritizer," in The EACL 2009 workshop on computational approaches to semitic languages, 2009, pp. 27–35, [Online]. Available: http://dl.acm.org/citation.cfm?id=1621774.1621780.

[16] A. Shahrour, S. Khalifa, and N. Habash, "Improving Arabic Diacritization through Syntactic Analysis," in Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, 2015, pp. 1309–1315, doi: 10.18653/v1/D15-1152.
[17] J. Nivre, J. Hall, and J. Nilsson, "MaltParser: A Data-Driven Parser-Generator for Dependency Parsing," in Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC'06), 2006, pp. 2216–2219, [Online]. Available: http://www.lrec-conf.org/proceedings/lrec2006/pdf/162_pdf.pdf.
[18] A. Chennoufi and A. Mazroui, "Morphological, syntactic and diacritics rules for automatic diacritization of Arabic sentences," J. King Saud Univ. - Comput. Inf. Sci., vol. 29, no. 2, pp. 156–163, 2017, doi: 10.1016/j.jksuci.2016.06.004.
[19] A. Fashwan and S. Alansary, "SHAKKIL: an automatic diacritization system for modern standard Arabic texts," in Proceedings of the Third Arabic Natural Language Processing Workshop, 2017, pp. 84–93, doi: 10.18653/v1/W17-1311.
[20] K. Darwish, H. Mubarak, and A. Abdelali, "Arabic Diacritization: Stats, Rules, and Hacks," in Proceedings of the Third Arabic Natural Language Processing Workshop, 2017, pp. 9–17, doi: 10.18653/v1/W17-1302.
[21] S. Alqudah, G. Abandah, and A. Arabiyat, "Investigating Hybrid Approaches for Arabic Text Diacritization with Recurrent Neural Networks," in 2017 IEEE Jordan Conference on Applied Electrical Engineering and Computing Technologies (AEECT), 2017, pp. 1–6, doi: 10.1109/AEECT.2017.8257765.
[22] H. Abbad and S. Xiong, "Multi-components System for Automatic Arabic Diacritization," in 42nd European Conference on Information Retrieval, ECIR 2020, 2020, vol. 12035, pp. 341–355, doi: https://doi.org/10.1007/978-3-030-45439-5_23.
[23] M.-T. Luong, H. Pham, and C. D. Manning, "Effective Approaches to Attention-based Neural Machine Translation," in Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, 2015, pp. 1412–1421, doi: 10.18653/v1/D15-1166.
[24] O. Hamed and T. Zesch, "A Survey and Comparative Study of Arabic Diacritization Tools," J. Lang. Technol. Comput. Linguist., vol. 32, no. 1, pp. 27–47, 2017.
[25] A. Fadel, I. Tuffaha, B. Al-Jawarneh, and M. Al-Ayyoub, "Arabic Text Diacritization Using Deep Neural Networks," in 2019 2nd International Conference on Computer Applications & Information Security (ICCAIS), May 2019, pp. 1–7, doi: 10.1109/CAIS.2019.8769512.