INTERSPEECH 2023
20-24 August 2023, Dublin, Ireland

Take the Hint: Improving Arabic Diacritization with Partially-Diacritized Text

Parnia Bahar[1], *Mattia Di Gangi*[1], *Nick Rossenbach*[1,2], *Mohammad Zeineldeen*[1,2]

[1]AppTek GmbH, Aachen, Germany
[2]RWTH Aachen University, Aachen, Germany

pbahar,mdigangi,nrossenbach,mzeineldeen@apptek.com

Abstract

Automatic Arabic diacritization is useful in many applications, ranging from reading support for language learners to accurate pronunciation predictor for downstream tasks like speech synthesis. While most of the previous works focused on models that operate on raw non-diacritized text, production systems can gain accuracy by first letting humans partly annotate ambiguous words. In this paper, we propose 2SDiac, a multi-source model that can effectively support optional diacritics in input to inform all predictions. We also introduce Guided Learning, a training scheme to leverage given diacritics in input with different levels of random masking. We show that the provided hints during test affect more output positions than those annotated. Moreover, experiments on two common benchmarks show that our approach i) greatly outperforms the baseline also when evaluated on non-diacritized text; and ii) achieves state-of-the-art results while reducing the parameter count by over 60%.

Index Terms: Arabic text diacritization, partially-diacritized text, Arabic natural language processing

1. Introduction

In Arabic, each written word form can represent up to dozens of pronounced words with different meanings because only consonants and long vowels are written. To determine pronunciation and disambiguate words, the text is annotated via secondary characters known as *diacritics*, which correspond to phonological information. Diacritics can significantly affect the meaning of sentences, and they are important for better readability and downstream tasks like automatic speech recognition [1], text-to-speech (TTS) [2], which in turn is important for automatic dubbing [3]. Native speakers find it more natural to read without diacritics, and almost all available modern standard Arabic (MSA) texts are not annotated with diacritics. Diacritics are generally present only in religious texts (notably the Holy Quran) or children's books, both requiring a high pronunciation precision. The generalized absence of diacritics from text poses a challenge to Arabic natural language processing (NLP).

Arabic text diacritization is the task of recovering missing diacritics in text. As such, the input is non-diacritized text, and the output is the text augmented with diacritics. In some cases, Arabic texts can come with a small portion of diacritical hints to disambiguate words. And, a partial diacritization of the text can improve the reading and typing speeds of humans [4]. Moreover, depending on the business applications, a limited human effort to add some manual diacritics can help improve the automatic diacritization quality while keeping the overall costs relatively low. With the goal of building high-quality systems that can further leverage partially-diacritized input for higher accuracy, in this paper we introduce:

1. **2SDiac** (**2**-**S**ource **Diac**ritizer), a bi-source model that takes Arabic text at character level as one source, and the corresponding optional diacritics as the second source.
2. **Guided Learning**, a training scheme for 2SDiac inspired by noisy auto-encoding for training with partially-diacritized text in the input.

2SDiac is based on a simple design, is parameter efficient, and is trained only on the diacritization task. On the common Tashkeela and ATB benchmarks, 2SDiac greatly outperforms single-source baseline models when no diacritics are provided. Furthermore, 2SDiac's results are similar to state-of-the-art models (with >13M to >800M parameters), while having a fraction of their parameters (4.9M).

2. Related Work

Early approaches have been fully based on linguistic rules with morphological analyzers as their main component [5, 6]. Research has then shifted to rely more on statistical methods [7, 8, 9, 10, 11], particularly neural networks, which are state-of-the-art due to their capability to learn contextual knowledge.

Recurrent long short-term memory (LSTM) networks [12] have been proven to be suitable tools for learning the task entirely from data without using manually designed features [13, 14, 15]. Their combination with conditional random field (CRF) and the extension to sequence-to-sequence modeling [16] help the model performance [17, 18]. [19] uses large self-attention models [20] with BERT-like pretraining where the model incorporates additional auto-generated knowledge instances. The idea of a multi-source model is orthogonal to this, and our experiments show that neither a large model nor large data are fundamental for the benchmarks.

Other works have shown the benefits of jointly modeling lexicalized and non-lexicalized morphological features. [21, 22, 23] use additional hand-crafted features likes lemmas and part-of-speech tags to improve diacritization. Being dependent on morphological features, on one hand, these approaches are limited by available resources that provide additional features. On the other hand, such knowledge sources can be inaccurate. [24] tackles the data sparsity problem of Arabic diacritization texts by using additional bilingual texts and employing a word-level multi-task setup to diacritize and translate using large self-attention models. In contrast, our approach requires no additional data and leads to a compact and data-efficient model.

[25] propose an attention-based [26] model with hierarchical encoders. Their architecture combines word- and character-level encoders and an autoregressive decoder. With similar motivations to ours, they suggest applying diacritical hints in the input by forcing the model predictions and feeding them back to the model. Our approach differs from this as the hints af-

 10.21437/Interspeech.2023-903

fect all the internal model representations starting from the first layer, and not only the decoder by forcing output predictions.

Other works address the diacritization task by selectively restoring a group of diacritics required for specific purposes [27, 4]. Such studies limit the number of needed diacritics by definition of half, partial, or minimal diacritization on the output. These studies are orthogonal to ours, as we are interested in partial diacritics in input rather than as task objective. A survey of Arabic text diacritization methods can be found in [28].

3. Proposed Approach

In order to automatically restore diacritics, we aim to have small yet powerful models, so to minimize their computational overhead in processing pipelines. We treat the diacritization problem as a classification task and use both bidirectional LSTM (BiLSTM) and self-attention networks [20].

3.1. Baseline

The input to the model is a sequence of Arabic characters, and the model prediction is diacritics per position. Formally, let us assume an input sequence of Arabic non-diacriticized characters $c_1^N = c_1, \ldots, c_N$ and an output sequence of diacritic marks $y_1^N = y_1, \ldots, y_N$ of the same length. Each y_i can represent 0, 1, or 2 diacritic marks assigned to a character. The model is non-autoregressive, which assumes the output predictions are independent of each other:

$$y_i = \text{argmax}_{j \in C}\{P(\tilde{y}_{ij}|c_1, \ldots, c_N)\}$$

where C is a set of diacritic classes of size 15 as in [17] and $\tilde{y}_{ij}$ is the candidate diacritic j for position i.

The character sequence is converted in an embedding tensor and then fed to stacked BiLSTM or self-attention layers, followed by a linear layer and softmax to compute a probability distribution over the 15 output classes.

3.2. Autoregressive

The independence assumption made for the baseline can result in subpar predictions. We therefore experiment with a network architecture that adds a 1-layer autoregressive decoder that takes as input the output from the previous layer and the diacritic embedding of the last prediction. The diacritics embedding is a new additional matrix. The model is now

$$y_i = \text{SEARCH}_{j \in C}\{P(\tilde{y}_{ij}|y_{i-1}; c_1, \ldots, c_N)\}$$

enabling the use of beam search. When adding autoregression, we want to keep the network comparable in size with the baseline, thus our autoregressive layer is an extension of the existing output linear layer, which now takes as input the concatenation of the two aforementioned input tensors and produces the unnormalized probabilities for the output classes.

3.3. Partially-Diacritized Input

We design our model, which we call **2SDiac**, to leverage diacritics from the input text for influencing the model's internal representations and their consequent outputs. An additional design objective is to keep the input length equal to the number of characters, thus having a 1-to-1 mapping between the input and output sequence positions. The second design goal prevents us from adding the diacritics in the character input sequence. Thus, we propose a multi-source model with characters and diacritics as two source sequences of the same length. Let $c_1^N = c_1, \ldots, c_N$ be the character sequence without diacritics as in §3.1, and $d_1^N = d_1, \ldots, d_N$ be a sequence of diacritics that include a <blank> symbol which represents the absence of a diacritic symbol and has not to be confused with the Arabic diacritic symbol *Sukun*. Then, 2SDiac is trained to predict $y_1^N = y_1, \ldots, y_N$, where $y_i = d_i$ for $d_i \neq$ <blank>.

The two input sequences are used to index two different embedding matrices. The two embedding sequences are then summed element-wise to produce a final embedding sequence that is input to the BiLSTM or self-attention model. With this formulation, the diacritic source can model any sequence of diacritics, or no diacritics, in input. When no diacritics are provided, the diacritics sequence consists only of <blank>, and the model is conceptually equivalent to the baseline, except for a fixed bias tensor.

Given the additional source sequence in 2SDiac, it has to be trained with diacritical hints in input, whereas the baseline is trained only with plain text. We propose a training scheme, called **Guided Learning**, which is inspired by noisy autoencoders and masked language model [29], extended to our multi-source case. It consists in copying the reference diacritics into the input sequence and then masking some of the provided diacritics with a *masking factor* $\lambda \in [0, 1]$ to randomly replace a percentage of diacritics with the <blank> symbol, which in practice removes the diacritic hints. E.g., $\lambda = 0.3$ means 30% of diacritics are removed from the input. Analogously, $\lambda = 1.0$ has no diacritics at all, and $\lambda = 0.0$ represents an input with all reference diacritics also provided as a hint to the system.

We precompute different versions of the input with $\lambda = 0.0, 0.1, \ldots, 1.0$ and shuffle all of them during training. We do this for the training and the dev set. The model is in practice trained on tasks with different difficulties, ranging from diacritic restoration of raw text only, to diacritics copy where the characters are supposedly ignored, and all the tasks in between. In our experiments, we set $\lambda = 1$ during testing for a fair comparison with others, where not otherwise specified.

4. Experiments

4.1. Datasets

We use two benchmark datasets. The first one is the publicly available Tashkeela corpus [30] consisting of different domains from articles, news, speeches, and school lessons. We use the filtered version of it by [14] where the data has been cleaned, and a portion of it with high coverage of diacritics has been selected. This data has been randomly divided into training (50k sentences), development (2.5k sentences), and test set (2.5k sentences) and is considered as a standard benchmark for the Arabic text diacritization task used in numerous publications. The second benchmark is the LDC Arabic Treebanks (ATB) corpus which consists of newswire stories. We follow the same data split as introduced in [31] and used in other studies. The data consists of ATB part 1 v4.1 (LDC2010T13), part 2 v3.1 (LDC2011T09), and part 3 v3.2 (LDC2010T08) split into the training (15.8k sentences), development (1.9k sentences), and test (1.9k sentences) sets. All text is tokenized using the Moses toolkit [32]. We remove the "dagger Alif" symbol from the text as it rarely occurs in MSA. We also consider the "Alif" with "Maddah" as a letter together without diacritics.

We also evaluate on the freely available WikiNews test set[1] [10], and provide the first results on it with models trained only

[1] https://github.com/kdarwish/Farasa

Table 1: DER [%] *and* WER [%] *of the systems (trained only on the Tashkeela train set) on the Tashkeela test.*

nr.	system	including "no diacritic"				excluding "no diacritic"				# params (M)
		w/ case ending		w/o case ending		w/ case ending		w/o case ending		
		DER	WER	DER	WER	DER	WER	DER	WER	
1	BiLSTM [14]	3.7	11.2	2.9	6.5	4.4	10.9	3.3	6.4	-
2	BiLSTM + CRF [17]	2.6	7.7	2.1	4.6	3.0	7.4	2.4	4.4	-
3	self-attention + Bert-like [19]	1.9	5.5	1.6	3.6	-	-	-	-	847
4	hierarchical BiLSTM (D3) [25]	**1.8**	**5.3**	**1.5**	**3.1**	**2.1**	**5.1**	**1.7**	**3.0**	13.4+
5	BiLSTM	3.2	9.3	2.7	5.9	3.7	9.0	3.1	5.7	4.0
6	+ autoregressive	3.1	8.8	2.6	5.5	3.6	8.5	3.0	5.4	4.3
7	BiLSTM 2SDiac	2.0	5.9	1.7	3.6	2.4	5.7	1.9	3.5	4.0
8	+ autoregressive	2.0	5.8	1.8	3.5	2.3	5.5	1.9	3.4	4.3
9	self-attention	2.9	8.4	2.3	4.8	3.3	8.1	2.6	4.7	4.8
10	+ autoregressive	2.4	7.0	1.9	4.1	2.7	6.8	2.2	4.0	4.9
11	self-attention 2SDiac	2.0	5.7	1.6	3.3	2.3	5.5	**1.8**	3.3	4.8
12	+ autoregressive	**1.9**	**5.6**	**1.5**	**3.2**	**2.2**	**5.3**	**1.8**	**3.2**	4.9

on public data (Tashkeela). We clean WikiNews by removing blank lines, hyphens, and Latin quotation marks.

4.2. Evaluation

For Tashkeela and WikiNews evaluation, we follow the work by [14] and the provided evaluation script and report diacritization error rate (DER) and word error rate (WER) excluding numbers and punctuation. Following the work by [33], we also report the results with/without case ending, and with/without the "no diacritic" class. For comparison with previous works on ATB, we use a different scheme to compute WER and DER where numbers and punctuation are also taken into account [33, 19].

4.3. Training Setups

The embedding size for all models is 128. The BiLSTM networks have 3 layers with 256 units per direction on top of character embeddings. The final linear projection maps from size 512 to 15 output classes. The BiLSTM models are trained for 25 epochs using Adam [34] with a learning rate of 0.001, decay factor of 0.7, dropout of 0.2. The batch size is 120k characters for the non-autoregressive models and 100k for the autoregressive models. The self-attention networks follow the Transformer encoder [20] and have 6 layers of size 256, 4 attention heads per layer, and feed-forward layers consisting of 2 projections separated by ReLu with the hidden size 1024. They are trained for 50 epochs using a decay factor of 0.9 where we wait for 3 epochs to reduce the learning rate. The softmax layer applies a dropout of 0.3 while all other layers use 0.1. The batch size is set to 10k. The beam size is 5 for the autoregressive models. The dev sets are used for hyperparameter tuning and checkpoint selection. The BiLSTM and self-attention 2SDiac models are trained in about 14h and 22h on a single RTX 2080 GPU, respectively. The code and the configurations of our setups using RETURNN [35] are available online[2].

5. Results & Analysis

Table 1 shows the performance of our models on the Tashkeela benchmark. By comparing lines 5 and 9, we see that the self-attention baseline outperforms slightly but consistently the BiLSTM baseline, and it has 0.8M parameters more. Lines 6 and 10 show that the autoregressive models are consistently better than their non-autoregressive counterparts, with a larger gain on the self-attention case. However, they are far behind the state-of-the-art models in lines 3-4. The BiLSTM and self-attention 2SDiac models outperform the respective baselines by a large margin in all metrics, e.g. by 36.6% and 32.1% relative WER in the "no diacritics" with case ending evaluation. Autoregressive 2SDiac is only slightly better, with improvements of 0.0-0.2%. Since the autoregressive decoder is simply a single linear layer, the slight improvements are not obvious and we consider the autoregressive modeling more accurate for this task.

Comparing the results with the state of the art listed in the first block of the table, the best 2SDiac model is mostly on par with [19] that uses a large version of n-gram-aware BERT model, called ZEN 2.0 [36], pretrained on a huge amount of monolingual data and has more than 800M parameters. The best results published for Tashkeela, to the best of our knowledge, are from the D3 model proposed by [25]. It is an autoregressive sequence-to-sequence model with hierarchical encoder trained in 2 stages with ad-hoc losses. The parameter count of 13.4M is provided in the paper only for the D2 model, which lacks the decoder side, so the actual size is more than that. Despite the size difference and the simplicity of our training, the best 2SDiac is only 0.1 absolute WER and DER worse in most metrics, and 0.2 WER in excluding "no diacritics".

Table 2 shows the performance of our self-attention 2SDiac autoregressive model on the ATB task, along with the most recent work from the literature. 2SDiac achieves 3.3% WER on the test set when no case ending is included, a 31% relative improvement over the previous best result from [24], which trained a multitask model for diacritization and translation with a large quantity of bitext Ar↔En data. Their model is based on a big transformer encoder-decoder architecture of 6 layers and large layer sizes, resulting in hundreds of millions of parameters. This contrasts with our architecture with less than 5M parameters which is trained fast on a single GPU, requires low computational resources during inference and has low latency, which is a critical factor for real-time applications. Compared to [22] that utilizes additional morphological features, 2SDiac achieves slightly better results.

We also evaluate our 2SDiac from line 12 in Table 1 on the unseen test set of WikiNews (see Table 3), with a different domain. Since this is the first work evaluating on WikiNews with a model trained only on public training data, no fair comparison with previous works is possible. Despite the high WER numbers due to cross-domain testing, we observe again that 2SDiac

[2]https://github.com/apptek/ArabicDiacritizationInterspeech2023

Table 2: DER [%] *and* WER [%] *of the systems (trained only on the ATB train set) on the ATB test.*

system	including "no diacritic"			
	w/ case ending		w/o case ending	
	DER	WER	DER	WER
multi-task + features [22]	2.5	7.5	-	-
multi-task + bitext [24]	3.6	-	1.7	4.8
2SDiac	**2.3**	**7.2**	**1.6**	**3.3**

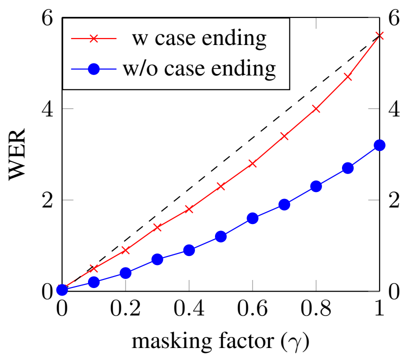

Figure 1: WER [%] *(including "no diacritic") of partially-diacritized input with different* λ*, ranged from 0.0 to 1.0.*

is much better than the single-source model under all metrics.

5.1. Partially-Diacritized Text

Figure 1 shows the results in WER including "no diacritic" against different values of λ from 0.0 to 1.0. In this experiment, the model (nr. 12 in Table 1) trained using the 11 masking factors from 0.0 to 1.0, is evaluated on different levels of masking factors applied to the test set. According to the figure, as the masking factor increases from 0.0 (i.e. keeping all diacritics in the input) to 1.0 (i.e., removing all diacritics), the WER results go up from almost 0 to the values indicated in Table 1. In particular, the error rates go up from 0.06% to 5.6% with the case ending and from 0.03% to 3.2% without case ending, respectively, when changing λ in the $[0.0, 1.0]$ interval. The dashed line illustrates the theoretical improvement when the provided hints and the model errors are both i.i.d., which would have a linear improvement proportional to the masking factor. The red curve representing 2SDiac performance is clearly below the dashed line, showing that the model can use the hints to make better predictions for other positions.

5.2. User Experience

One important factor in production systems is user experience. The model has been designed and trained to copy the provided hints. We then expect it to keep this behavior during test and "trust" the input information by generating output accordingly. This is coherent with the principle of least surprise[3]. However, when user-provided hints are not coherent with the training vocabulary, the model may ignore them to produce instead an in-vocabulary word. In the next experiment, we want to measure the percentage of copy when the model is provided with random hints in input, and its robustness to introduced errors by measuring the corresponding DER. We prepare the Tashkeela test set by randomly adding hints in the input from the 8 main Arabic diacritics. The level of inserted noise can vary in the $[0.0, 1.0]$ interval. We compute the document-level copy percentage as a ratio of the number of copied diacritics over the total number of randomly inserted diacritics. We observe copy percentages of 78.4% to 84.7% when increasing the noise from 10% to 40%. The DER changes rapidly from 11.5% to 40.7%, following the noise ratio more than the copy ratio, further proving the model sensitivity to the user-provided hints.

[3]https://en.wikipedia.org/wiki/Principle_of_least_astonishment

Table 3: DER [%] *and* WER [%] *of the systems (trained only on Tashkeela due to its data quality) on the WikiNews test.*

system	including "no diacritic"			
	w/ case ending		w/o case ending	
	DER	WER	DER	WER
self-attention	13.2	38.6	11.0	26.3
2SDiac	**11.6**	**33.2**	**9.9**	**22.7**
system	**excluding "no diacritic"**			
	w/ case ending		w/o case ending	
	DER	WER	DER	WER
self-attention	15.5	37.9	12.8	26.0
2SDiac	**13.2**	**32.2**	**11.1**	**22.1**

5.3. Discussion

The results in this study are mainly limited by the size and domains of the available datasets. Yet, within the limits of the study, training with diacritics in input resulted to be surprisingly effective for improving models' quality. 2SDiac without Guided Learning theoretically and practically identical to the baseline (not shown), but together they produce a state-of-the-art model. According to our observation, Guided Learning also helps with limited masking range, but it is recommended to add all variants for better generalization. The learning curves show better training and dev metrics throughout the entire training process, suggesting that it represents a better modeling for the data and not only better regularization. The generality of this approach on larger and more diverse data should be validated in future work, but the present results show that simple models can be very competitive in the diacritization task.

6. Conclusions

In this work, we have presented a new model that supports partially-diacritized text as input motivated by the practical reason of leveraging existing diacritic marks in texts as hints for better model prediction. The proposed 2SDiac model provides a large performance gain over single-sourced recurrent and self-attention models (relative improvement of about 36% in WER), as well as achieving state-of-the-art results on the Tashkeela and ATB test sets, while having a fraction of the parameters of the compared models and a simple design. Our approach is orthogonal to other methods in the literature, and as a future work we are interested in how it combines with other state-of-the-art approaches and other Arabic NLP tasks.

7. Acknowledgement

This work was partially supported by NeuroSys which, as part of the initiative "Clusters4Future", is funded by the Federal Ministry of Education and Research BMBF (03ZU1106DA).

8. References

[1] D. Vergyri and K. Kirchhoff, "Automatic diacritization of Arabic for acoustic modeling in speech recognition," in *Workshop on Computational Approaches to Arabic Script-based Languages*, Aug. 2004.

[2] Drago, Burileanu, V. Popescu, C. Negrescu, and A. Dervi, "Automatic diacritic restoration for a tts-based e-mail reader application," *UPB Scientific Bulletin*, 2008.

[3] M. Di Gangi, N. Rossenbach, A. Pérez, P. Bahar, E. Beck, P. Wilken, and E. Matusov, "Automatic video dubbing at AppTek," in *Conference of the European Association for Machine Translation (EAMT)*, Jun. 2022.

[4] E. Hermena, D. Drieghe, S. Hellmuth, and S. Liversedge, "Processing of arabic diacritical marks: Phonological-syntactic disambiguation of homographic verbs and visual crowding effects," *Journal of Experimental Psychology: Human Perception and Performance*, vol. 41, 2015.

[5] N. Habash, O. Rambow, and R. Roth, "Mada+ tokan: A toolkit for arabic tokenization, diacritization, morphological disambiguation, pos tagging, stemming and lemmatization," in *International Conference on Arabic Language Resources and Tools (MEDAR)*, Apr. 2009.

[6] A. Pasha, M. Al-Badrashiny, M. Diab, A. El Kholy, R. Eskander, N. Habash, M. Pooleery, O. Rambow, and R. Roth, "MADAMIRA: A fast, comprehensive tool for morphological analysis and disambiguation of Arabic," in *International Conference on Language Resources and Evaluation (LREC)*, May 2014.

[7] M. Elshafei, H. Al-Muhtaseb, and M. M. Alghamdi, "Statistical methods for automatic diacritization of arabic text," in *The Saudi National Computer Conference*, vol. 18, 2006.

[8] Y. Hifny, "Restoration of arabic diacritics using dynamic programming," in *Conference on Computer Engineering and Systems (ICCES)*, 2013.

[9] I. Zitouni, J. S. Sorensen, and R. Sarikaya, "Maximum entropy based restoration of arabic diacritics," in *Annual Meeting of the Association for Computational Linguistics (ACL)*, Jul. 2006.

[10] K. Darwish, H. Mubarak, and A. Abdelali, "Arabic diacritization: Stats, rules, and hacks," in *Arabic Natural Language Processing Workshop (WANLP)*, Apr. 2017.

[11] M. S. Khorsheed, "Diacritizing arabic text using a single hidden markov model," *Institute of Electrical and Electronics Engineers (IEEE)*, vol. 6, 2018.

[12] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, Nov. 1997.

[13] Y. Belinkov and J. R. Glass, "Arabic diacritization with recurrent neural networks," in *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Sep. 2015.

[14] A. Fadel, I. Tuffaha, B. Al-Jawarneh, and M. Al-Ayyoub, "Arabic text diacritization using deep neural networks," *ArXiv*, May 2019.

[15] K. Darwish, A. Abdelali, H. Mubarak, and M. Eldesouki, "Arabic diacritic recovery using a feature-rich bilstm model," *ACM Transactions on Asian Low Resource Language Information Processing*, vol. 20, 2021.

[16] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Conference on Neural Information Processing Systems (NeurIPS)*, Dec.ember 2014.

[17] A. Fadel, I. Tuffaha, B. Al-Jawarneh, and M. Al-Ayyoub, "Neural Arabic text diacritization: State of the art results and a novel approach for machine translation," in *Workshop on Asian Translation*, Nov. 2019.

[18] H. Mubarak, A. Abdelali, H. Sajjad, Y. Samih, and K. Darwish, "Highly effective arabic diacritization using sequence to sequence modeling," in *Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, Jun. 2019.

[19] H. Qin, G. Chen, Y. Tian, and Y. Song, "Improving arabic diacritization with regularized decoding and adversarial training," in *Annual Meeting of the Association for Computational Linguistics (ACL)*, Aug. 2021.

[20] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Conference on Neural Information Processing Systems (NeurIPS)*, Dec. 2017.

[21] N. Zalmout and N. Habash, "Adversarial multitask learning for joint multi-feature and multi-dialect morphological modeling," in *Annual Meeting of the Association for Computational Linguistics (ACL)*, Jul. 2019.

[22] S. Alqahtani, A. Mishra, and M. Diab, "A multitask learning approach for diacritic restoration," in *Annual Meeting of the Association for Computational Linguistics (ACL)*, Jul. 2020.

[23] N. Zalmout and N. Habash, "Joint diacritization, lemmatization, normalization, and fine-grained morphological tagging," in *Annual Meeting of the Association for Computational Linguistics (ACL)*, Jul. 2020.

[24] B. Thompson and A. Alshehri, "Improving Arabic diacritization by learning to diacritize and translate," in *International Workshop on Spoken Language Translation (IWSLT)*, May 2022.

[25] B. AlKhamissi, M. N. ElNokrashy, and M. Gabr, "Deep diacritization: Efficient hierarchical recurrence for improved arabic diacritization," in *Arabic Natural Language Processing Workshop (WANLP)*, Dec. 2020.

[26] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *International Conference on Learning Representations (ICLR)*, May 2015.

[27] A. M. Azmi and R. S. Almajed, "A survey of automatic arabic diacritization techniques," *Natural Language Engineering*, vol. 21, 2015.

[28] M. M. Almanea, "Automatic methods and neural networks in arabic texts diacritization: A comprehensive survey," *Institute of Electrical and Electronics Engineers (IEEE)*, vol. 9, 2021.

[29] J. K. Devlin, M.-W. Chang, and L. K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, Jun. 2019.

[30] T. Zerrouki and A. Balla, "Tashkeela: Novel corpus of arabic vocalized texts, data for auto-diacritization systems," *Data in Brief*, vol. 11, 2017.

[31] M. T. Diab, N. Habash, O. Rambow, and R. Roth, "LDC arabic treebanks and associated corpora: Data divisions manual," *ArXiv*, Sep. 2013.

[32] P. Koehn, H. Hoang, A. Birch, C. Callison-Burch, M. Federico, N. Bertoldi, B. Cowan, W. Shen, C. Moran, R. Zens, C. Dyer, O. Bojar, A. Constantin, and E. Herbst, "Moses: Open source toolkit for statistical machine translation," in *Annual Meeting of the Association for Computational Linguistics (ACL)*, Jun. 2007.

[33] I. Zitouni and R. Sarikaya, "Arabic diacritic restoration approach based on maximum entropy models," *Comput. Speech Lang.*, vol. 23, 2009.

[34] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *International Conference on Learning Representations (ICLR)*, May 2015.

[35] A. Zeyer, T. Alkhouli, and H. Ney, "Returnn as a generic flexible neural toolkit with application to translation and speech recognition," in *Annual Meeting of the Association for Computational Linguistics (ACL)*, Jul. 2018.

[36] Y. Song, T. Zhang, Y. Wang, and K. Lee, "ZEN 2.0: Continue training and adaption for n-gram enhanced text encoders," *ArXiv*, Feb. 2021.