Neural Arabic Text Diacritization: State-of-the-Art Results and a Novel Approach for Arabic NLP Downstream Tasks

ALI FADEL, IBRAHEEM TUFFAHA, and MAHMOUD AL-AYYOUB,
Jordan University of Science and Technology

In this work, we present several deep learning models for the automatic diacritization of Arabic text. Our models are built using two main approaches, viz. Feed-Forward Neural Network (FFNN) and Recurrent Neural Network (RNN), with several enhancements such as 100-hot encoding, embeddings, Conditional Random Field (CRF), and Block-Normalized Gradient (BNG). The models are tested on the only freely available benchmark dataset and the results show that our models are either better or on par with other models even those requiring human-crafted language-dependent post-processing steps, unlike ours. Moreover, we show how diacritics in Arabic can be used to enhance the models of downstream NLP tasks such as Machine Translation (MT) and Sentiment Analysis (SA) by proposing novel *Translation over Diacritization* (ToD) and *Sentiment over Diacritization* (SoD) approaches.

CCS Concepts: • **Computing methodologies → Neural networks;**

Additional Key Words and Phrases: Arabic, datasets, diacritization, translation, sentiment analysis

ACM Reference format:
Ali Fadel, Ibraheem Tuffaha, and Mahmoud Al-Ayyoub. 2021. Neural Arabic Text Diacritization: State-of-the-Art Results and a Novel Approach for Arabic NLP Downstream Tasks. *ACM Trans. Asian Low-Resour. Lang. Inf. Process.* 21, 1, Article 20 (December 2021), 25 pages.
https://doi.org/10.1145/3470849

1 INTRODUCTION

In Arabic and many other languages, diacritics are added to the characters of a word (as short vowels) in order to convey certain information about the meaning of the word as a whole and its place within the sentence. **Arabic Text Diacritization** (**ATD**) is an important problem with various applications such as **text to speech** (**TTS**). At the same time, this problem is a very challenging one even to native speakers of Arabic due to the many subtle issues in determining the correct diacritic for each character from the list shown in Table 2 and the lack of practice for many native speakers. Thus, the need to build automatic Arabic text diacritizers is high [38].

We gratefully acknowledge the support of the Deanship of Research at the Jordan University of Science and Technology for supporting this work via Grant #20180193 in addition to NVIDIA Corporation for the donation of the Titan Xp GPU that was used for this research.
Authors' address: A. Fadel, I. Tuffaha, and M. Al-Ayyoub, Jordan University of Science and Technology, Irbid 22110, Jordan; emails: {aliosm1997, bro.t.1996, malayyoub}@gmail.com.

2375-4699/2021/12-ART20 $15.00
https://doi.org/10.1145/3470849

Table 1. The Different Arabic Characters along with Their Variations

ز	ر	ذ	د	خ	ح	ج	ث	ت	ة	ب	ا	ئ	إ	ؤ	أ	آ	ء
ي	ى	و	ه	ن	م	ل	ك	ق	ف	غ	ع	ظ	ط	ض	ص	ش	س

Table 2. The 15 Diacritic Classes Under Consideration[1]

Class Name	Class Shape	Class Name	Class Shape
No Diacritization	ت	Shadda	تّ
Fatha	تَ	Shadda + Fatha	تَّ
Damma	تُ	Shadda + Damma	تُّ
Kasra	تِ	Shadda + Kasra	تِّ
Fathatan	تً	Shadda + Fathatan	تًّ
Dammatan	تٌ	Shadda + Dammatan	تٌّ
Kasratan	تٍ	Shadda + Kasratan	تٍّ
Sukun	تْ		

The meaning of a sentence is greatly influenced by the diacritization, which is determined by the context of the sentence as shown in the following example:

كلم أحمد ...
Buckwalter Transliteration: klm >Hmd ...
Incomplete sentence without diacritization.
كَلَّمَ أَحْمَدُ صَدِيقَهُ
Buckwalter Transliteration: kal~ama >aHomadN Sadiyqahu
Translation: Ahmad talked to his friend.
كَلَمَ أَحْمَدُ عَدُوَّهُ
Buckwalter Transliteration: kalama >aHomadN Eaduw~ahu
Translation: Ahmad wounded his enemy.

The letters كلم "klm" manifests into two different words when given two different diacritizations. As shown in this example, كَلَّمَ "kal~ama" in the first sentence is the verb "talked" in English, while كَلَمَ "kalama" in the second sentence is the verb "wounded" in English.

The ATD problem is formulates in a formal manner as follows. Given a sequence of characters representing an Arabic sentence S (where the full set of characters are shown in Table 1), automatically assign each Arabic character $S_i \in S$ the correct diacritic class from the ones depicted in Table 2. In this work, we consider a version of the ATD problem where each character must be diacritized. This is known as the Full Diacritization problem [29]. Other versions such as Half Diacritization and Partial Diacritization are part of our future plans for this work.

Despite the problem's importance, it received limited attention. One of the reasons for this is the scarcity of freely available resources for this problem. To address this issue, the Tashkeela Corpus[2] [37] has been released to the community. Unfortunately, there are many problems with the use of this corpus for benchmarking purposes. A very recent study [24] discussed in details these issues and provided a cleaned version of the dataset with pre-defined split into training, testing, and

[1]Note that some diacritization conventions do not allow "incomplete or missing" diacritics, which means that the "No Diacritization" and "Shadda" classes should not exist. However, this is not always the case. In fact, there are many texts where partial (or even minimal) diacritization is used. An example of work exclusively focusing on such cases is Alqahtani et al. [7].
[2]https://sourceforge.net/projects/tashkeela.

validation sets. In this work, we use this dataset and provide yet another extension of it with a larger training set and a new testing set in order to provide means of fair comparison with some of the existing systems that have already been trained on the entire Tashkeela Corpus.

According to [24], existing approaches to ATD are split into two groups: traditional rule-based approaches and machine-learning-based approaches. The former was the main approach by many researchers such as [18, 32, 38], while the latter has started to receive attention only recently [2, 12, 14, 31]. Based on the extensive experiments of [24], deep learning approaches (aka neural approaches) are superior to non-neural approaches especially when large training data are available. In this work, we present several neural ATD models and compare their performance with the **state-of-the-art** (**SOTA**) approaches to show that our models are either on par with the SOTA approaches or even better. Finally, we present a novel way to utilize diactritization in order to enhance the accuracy of several models for downstream Arabic NLP tasks such as **Machine Translation** (**MT**) and **Sentiment Analysis** (**SA**) in what we call ***Translation over Diacritization*** (**ToD**) and ***Sentiment over Diacritization*** (**SoD**) approaches.

The rest of the article is organized as follows. The following section presents our survey of the literature. Section 3 discusses the ATD datasets used in this work as well as the evaluation metrics used toe valuate the proposed models. Section 4 discusses our two main approaches for ATD: **Feed-Forward Neural Network** (**FFNN**) and **Recurrent Neural Network** (**RNN**), respectively. Section 5 discusses the results of our experimentation on the proposed models. Moreover, this section provides a brief literature survey and presents a comparison with the SOTA approaches, while Section 6 describes our novel approach to integrate diacritization into downstream tasks. The article is concluded in Section 7 with final remarks and future directions of this work.

2 RELATED WORK

As mentioned in the previous section, although the ATD problem is an important one with various applications, the efforts on building automatic ATD are limited. A recent study [24] surveyed existing approaches and tools for ATD. After discussing the limitations in closed-source tools, they divided existing approaches to ATD into two groups: traditional rule-based approaches [6, 7, 10, 13, 17, 18, 25, 29, 32, 34, 38] and machine-learning-based approaches [2, 3, 5, 8, 12, 14, 30, 31]. The extensive experiments of [24] showed that neural ATD models are superior to their competitors especially when large training data are available. Thus, we limit our attention in this work to such models.

Belinkov and Glass [14] presented a language-agnostic system for Arabic text diacritization. According to [18], this is the first work on ATD that does not employ linguistic features and tools. The authors trained their system on diacritized text extracted from the **Arabic TreeBank** (**ATB**) dataset [19] without relying on additional resources. They used different types of neural networks in addition to letter embeddings to automatically diacritize Arabic text. The network types they considered are FFNN, **Long Short-Term Memory** (**LSTM**), **Bidirectional LSTM** (**BiLSTM**), and stacked BiLSTM. This approach is open-sourced and its results rival those of the SOTA systems that rely on language-specific tools such as the MaxEnt approach of Zitouni and Sarikaya [38].

The work of Abandah et al. [2] (and its extensions [1, 3, 8]) used stacked BiLSTM. Experimenting on ATB, the Tashkeela Corpus and the Holy Quran, Abandah et al. [2] achieved SOTA performance after applying some language-related, post-processing, and error correction techniques.

Al-Thubaity et al. [5] proposed to combine BiLSTM with **Conditional Random Fields** (**CRF**). This BiLSTM-CRF approach [26] has been shown to be very useful for many sequence tagging tasks including ones in Arabic [4]. The authors trained and tested their model on four datasets that include ATB and the Holy Quran in addition to Sahih Al-Bukhary and the **King Abdulaziz**

Table 3. Extra Training Dataset Statistics

	Extra Train
Words Count	22.4 M
Lines Count	533 K
Avg Chars/Word	3.97
Avg Words/Line	42.1
0 Diacritics (%)	17.79
1 Diacritic (%)	77.16
2 Diacritics (%)	5.03
Error Diacritics (%)	0

City for Science and Technology text-to-speech (**KACST TTS**) dataset. The reported results are comparable or better than the SOTA results for the datasets under consideration.

Finally, the open-source project Shakkala was built by Barqawi and Zerrouki [12] for ATD using BiLSTM networks in addition to character embeddings. The model was trained on the Tashkeela Corpus several times while removing the data with negative influence on the training process. The system provides the diacritization service through an interactive web interface, without providing an API.[3] The website allows users to diacritize text containing up to 490 symbols. An error message is given if the input is longer. The code is publicly available on Github.[4] It has three different trained models. The first (and earliest) version is used in the website, while the third (and latest) version provides the best results but is limited to 315 characters at a time.

A one-of-its-kind work was done by Masmoudi et al. [29]. What is unique about this work is that it focuses on the Tunisian dialect. ATD models focus on either the **Classical Arabic** (**CA**) or the **Modern Standard Arabic** (**MSA**). Diacritizing dialectal Arabic text is a new frontier that has not been explored. The authors propose two models for this problem. The first one is based on **Statistical Machine Translation** (**SMT**), while the other is absed on CRF.

3 ARABIC TEXT DIACRITIZATION DATASETS AND EVALUATION METRICS

This section starts with a discussion on ATD datasets followed by a discussion on evaluation metrics.

3.1 Datasets

The first ATD dataset we use in this work is the one prepared by [24]. It is an adaptation of the Tashkeela Corpus and consists of about 2.3 M words spread over 55 K lines. Basic statistics about this dataset size, content, and diacritics usage are given in Table 4. More details about this dataset can be found at [24].

As part of our work, we use the well-known RNN architecture to build several ATD models. However, RNN models usually need huge data to train on and learn high-level linguistic abstractions. Thus, we prepare an external training dataset following the guidelines of [24]. In our experiments, we mention explicitly when a model uses this extra dataset.

The extra training dataset is extracted from the CA part of the Tashkeela Corpus and the Holy Quran. We exclude the lines that already exist in the previously mentioned dataset. Table 3 gives the statistics for the extra training dataset.

[3] https://ahmadai.com/shakkala.
[4] https://github.com/Barqawiz/Shakkala.

Table 4. Statistics about the Size, Content, and Diacritics Usage of [24]'s Dataset

	Train	Valid	Test
Words Count	2,103 K	102 K	107 K
Lines Count	50 K	2.5 K	2.5 K
Avg Chars/Word	3.97	3.97	3.97
Avg Words/Line	42.06	40.97	42.89
0 Diacritics (%)	17.78	17.75	17.80
1 Diacritic (%)	77.17	77.19	77.22
2 Diacritics (%)	5.03	5.05	4.97
Error Diacritics (%)	0	0	0

The lines in the dataset are split using the following 14 punctuations (".", ",", "،", ":", ";", "؛" , "(", ")", "[", "]", "{", "}", "«" and "»"). After that, the lines with length more than 500 characters (without counting diacritics) are split into lines of length no more than 500. This step is necessary for the training phase to limit the memory usage within a single batch. Note that the splitting procedure is omitted within the prediction phase, e.g., when calculating DER/WER on the validation and test sets. Moreover, four special tokens ("<SOS>", "<EOS>", "<UNK>" , and "<PAD>") are used to prepare the input data before feeding it to the model. "<SOS>" and "<EOS>" are added to the start and the end of the sequences, respectively. "<UNK>" is used to represent unknown characters not seen in the training dataset. Finally, "<PAD>" is appended to pad the sequences within the same batch. Four equivalent special tokens are used as an output in the target sequences.

3.2 Evaluation Metrics

Among the resources provided with [24]'s dataset are [24]'s new definitions of the **Diacritic Error Rate** (**DER**), which is "the percentage of misclassified Arabic characters regardless of whether the character has 0, 1, or 2 diacritics", and the **Word Error Rate** (**WER**), which is "the percentage of Arabic words, which have at least one misclassified Arabic character".[5] The redefinition of these metrics in [24] is to exclude counting irrelevant characters, such as numbers and punctuations, which were included in [38]'s original definitions of DER and WER. Thus, the revised definitions provide more realistic metrics of accuracy for ATD models. The interested reader is referred to [24] for more details and examples related to this issue.

It is worth mentioning that DER/WER are computed in four different ways in the literature depending on whether the last character of each word (referred to as *case ending*) is counted or not and whether the characters with no diacritizations are counted or not.

4 PROPOSED DIACRITIZATION MODELS

In this section, we describe our models in details starting with the ones built using FFNN followed by the ones built using RNN.

4.1 The Feed-Forward Neural Network (FFNN) Approach

This is our first approach and we present three models based on it.[6] In this approach, we consider diacritizing each character as an independent classification problem where the model is fed

[5]DER/WER are computed with diacritization_stat.py.

[6]For each one of our models (in this section and the following ones), its architecture and hyperparameters' values are selected based on a guided grid search process that we perform to obtain the best results. The details of this process are left out of this article to reduce cluttering and save some space.

the character to be diactrized along with its context, which is simply a 100 window centered at the character under consideration. To be more specific, the context window include the 50 non-diacritized characters preceding the current one and 49 non-diacritized characters following it. Thus, the model takes a 100-dimensional vector as an input.

For example, the sentence "ذَهَبَ عَلِي", the vector related to the character "ب" is as shown as follows:

$$x = [< \text{PAD} >, \ldots, < \text{PAD} >, ذ, ه, ب, \text{‘ ’}, ع, ل, ي, \text{PAD} >, \ldots, < \text{PAD} >]$$

In this representation, there are two characters before the character "ب" and four after it (including the whitespace). The special token "<PAD>" is used as a filler when there are no characters to feed. Note that this dataset has 75 unique characters and each character is mapped to a unique integer value after sorting them based on their Unicode representations including the special padding and unknown ("<UNK>") tokens.

Each example (character) is labeled with one of the 15 classes under consideration (shown in Table 2). The model outputs probabilities for each class. Using a Softmax output unit, the class with the maximum probability is selected as the output. The number of training, validation, and testing examples from converting the dataset into examples as described earlier are 9,017 K, 488 K, and 488 K, respectively.

Basic Model. After a massive exploration for finding the best hyperparameters to structure the model (like the number of layers, number of neurons in each layer, and the activation function), the resulting basic model is constructed using 17 hidden layers of different sizes. The activation function used in all layers is **Rectified Linear Unit** (**ReLU**) and the number of trainable parameters is about 1.5 M. The final structure is given in Table 16.

100-Hot Model. Starting from the basic model structure (Table 16), and by tuning the hyperparameters and using extra techniques, such as applying Dropout regularization, the model is able to learn better using what we call 100-hot representations.

To better explain this, consider the example depicted in Figure 1. In this example, the character under consideration is "ص". The input to the model consists of this character along with its context of size 100. For each character in this string, a 75-dimensional 1-hot vector is created.[7] This process creates 75-dimensional vector for each of the 100 characters in the context of the input character. Concatenating these vectors results in a 7,500-dimensional vector. Figure 1 explains the encoding process in details.

Based on empirical exploration, the model is structured to have five hidden layers with dropout as shown in Table 17. It has close to 2 M trainable parameters.

Embeddings Model. Using a very similar structure as the 100-hot model structure (Table 17), except for the 100-hot layer is replaced with an embeddings layer to learn feature vectors for each character through the training process, this model is structured as shown in Table 18.

4.2 The Recurrent Neural Network (RNN) Approach

In this subsection, we describe our RNN models. Like what we do with FFNN models, we start with the most basic model before discussing how we enhance it in different ways.

Basic Model. As mentioned in Section 3, we prepare an extra training dataset to cater for RNN models' need for massive training datasets. However, before discussing the models' performance with this extra training dataset, we discuss their performance on the original dataset. Several

[7]The 75 is the number of different characters in the dataset and, as can be seen in Figure 1, the 1-hot encoding assigns 1 to the entry in this vector corresponding to the character under consideration and 0s to the remaining entries.

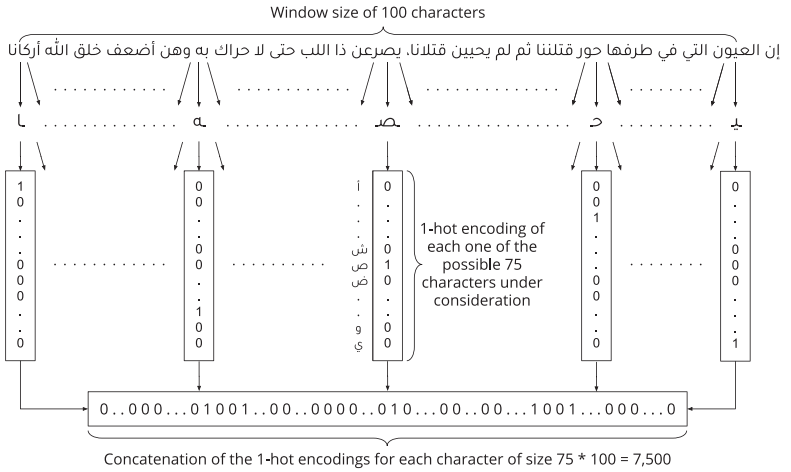

Fig. 1. 100-Hot model encoding process.

model architectures are trained without the extra training dataset. After thorough exploration, the best model architecture is chosen to experiment with different techniques as described in details throughout this section.

The exploration is done to tune different hyperparameters and find the structure that gives the best DER, which, in most cases, leads to better WER. Because the neural network size have a great impact on performance, we primarily experiment with the number of **Bidirectional CuDNN Long Short-Term Memory** (**BiCuDNNLSTM**) [9] layers and their hidden units. The error significantly decreases going from using one layer to using two layers. However, the results show slight improvement (if any) when going from using two layers to using three layers while increasing the training time. Thus, we decide to use two BiCuDNNLSTM layers in the rest of the experiments with 256 hidden units per layer since using a smaller number of units increases the error rate, while using a larger number of units does not significantly improve it. We also experiment with the size and depth of the fully connected feed-forward network. The results show that the depth is not as important as the size of each layer. The best results are produced with the model using two layers with 512 hidden units each. All experiments are done using the Adam optimization algorithm, because other optimizers like Stochastic Gradient Descent, Adagrad, and Adadelta do not converge to the optimal minimal fast enough, while RMSprop, Nadam, and Adamax give the same or slightly worse results. The number of character features to learn in the embedding layer that gives the best results is 25, where more features lead to little improvement and more overfitting, and less features make the training harder for the network. This is probably due to the input vocabulary being limited to 87 different characters. We also experiment with training the models for more than 50 epochs, but either the improvement is very small or the learning becomes unstable eventually causing exploding gradients, which leaves the network with useless predictions, unable to learn anymore. The best model is structured as shown in Figure 2.

In order to explore the impact of the dataset size on the training phase for the diacritization problem, the training is done twice: with and without the extra training dataset. This has led to reduced overfitting. A weights averaging technique over the last few epochs is applied to partially overcome the overfitting issue and obtain better generalizations.

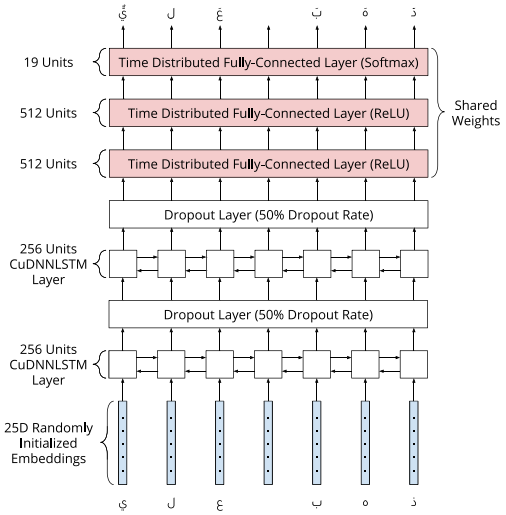

Fig. 2. RNN basic model structure.

The models in all of the following experiments are trained on Google Colab[8] [16] environment for 50 epochs using an Nvidia Tesla T4 GPU, Adam optimization algorithm with 0.001 learning rate, 0.9 beta1, 0.999 beta2, 10^{-7} epsilon, 256 batch size, and categorical cross-entropy loss function.

Conditional Random Field (CRF) Model. A CRF classifier is used in this model instead of the Softmax layer to predict the network output. CRF is usually more powerful than Softmax in terms of sequence dependencies in the output layer, which exist in the diacritization problem. It is worth mentioning that CRF is considered to be one of the "best practices" in sequence labeling problems. However, in this particular problem, the results show that CRF performs worse than Softmax in most cases except for WER results when training without the extra dataset, which indicates that even with worse DER results, CRF is able to make more consistent predictions within the same word.

Block-Normalized Gradient (BNG) Model. In this model, Yu et al. [36]'s BNG method is applied to normalize gradients within each batch. This can help accelerate the training process. According to [36], this method performs better in RNN when using optimizers with adaptive step sizes, such as Adam. It can also lead to solutions with better generalization. This coincides with our results.

5 EXPERIMENTS' RESULTS AND DISCUSSION

In this section we discuss the results of each one of our models and compare the best results we obtain with the SOTA models.

5.1 Results of the FFNN Models

For the FFNN model, the basic model results in 92.87%, 90.72%, and 90.67% accuracy for training, validation, and testing datasets, respectively. It is trained for 300 epochs on an Nvidia GeForce GTX 970 M GPU for about 16 hours using AdaGrad optimization algorithm [20] with 0.01 learning rate, 512 batch size, and categorical cross-entropy loss function. Figure 15 shows the loss and accuracy

[8] http://colab.research.google.com.

Table 5. DER/WER Comparison of the Different FFNN Models on the Test Set

DER/WER	w/case ending	w/o case ending	w/case ending	w/o case ending
	Including "no diacritic"		Excluding "no diacritic"	
Basic model	9.33%/25.93%	6.58%/13.89%	10.85%/25.39%	7.51%/13.53%
100-Hot model	6.57%/20.21%	4.83%/11.14%	7.75%/19.83%	5.62%/10.93%
Embeddings model	**5.52%/17.12%**	**4.06%/9.38%**	**6.44%/16.63%**	**4.67%/9.10%**

The best results in these tables are shown in bold font.

values on the training and validation datasets while training. The model is still able to slightly learn as well as generalize even after 300 epochs with no signs of overfitting.

The second FFNN model is the 100-hot model. This model results in 94.25%, 93.49%, and 93.45% accuracy for training, validation, and testing datasets, respectively. This is an improvement of 2.78% on the test set accuracy compared to the basic model. Figure 16 shows the loss and accuracy values on the training and validation datasets while training. The model is trained for 50 epochs on an Nvidia GeForce GTX 970 M GPU for about 3 hours using Adam optimization algorithm [27] with 0.001 learning rate, 0.9 beta1, 0.999 beta 2, 512 batch size, and categorical cross-entropy loss function.

The third and final FFNN model we have is the embeddings model, which achieves the best results compared to the basic and 100-hot models with 94.88%, 94.53%, and 94.49% accuracy for training, validation, and testing datasets, respectively. This model improves the accuracy by 1.04% on the test set compared to the 100-hot model while using 728 K trainable parameters, which represent size reductions of 51.46% and 62.66% compared to the basic and 100-hot models, respectively. Figure 17 shows the loss and accuracy values on the training and validation datasets while training. The model is trained with the same configurations as the 100-hot model and the training time is about 2.5 hours only.

Although the idea of diacritizing each character independently might seem counter-intuitive, the results of the FFNN models on the test set (shown in Table 5) are very promising with the embeddings model having an obvious advantage over the basic and 100-hot models and performing much better than the best rule-based diacritization system Mishkal[9] among the systems reviewed by [24] (Mishakl DER: 13.78% vs FFNN Embeddings model DER: 4.06%). However, these models are still imperfect.

Figure 3 shows the best diacritization examples diacritized using each FFNN model, while Figure 4 shows the worst diacritization examples. It is worth mentioning that the worst examples (listed in Figure 4) are from old Arabic poetry, which is very hard to diacritize flawlessly even for native speakers.

5.2 Results of the RNN Models

The results of the RNN models on the test set (shown in Table 6) are much better than the FFNN models by about 67%. To show the effect of the weights averaging technique, Table 7 reports the DER/WER statistics related to the BNG model after averaging its weights over the last 1, 5, 10, and 20 epochs. Studying the confusion matrices for all the models suggests that the Shadda class and the composite classes (i.e., Shadda + another diacritic) are harder to learn for the network compared to other classes. However, with the extra training dataset, the network is able to find significantly better results compared to the results without the extra training dataset, especially for the Shadda class.

[9] https://tahadz.com/mishkal.

Model		Best Line	File
FFNN Basic	Correct Diacritization	464 - حَدَّثَنِي يَحْيَى عَنْ مَالِك عَنْ هِشَامِ بْنِ عُرْوَةَ عَنْ أَبِيهِ Buckwalter Transliteration: 464 - Had~avaniy yaHoyaY Eano maAlik Eano hi$aAmi boni Eurowapa Eano >abiyhi	موطأ مالك
	Model's Output	464 - حَدَّثَنِي يَحْيَى عَنْ مَالِك عَنْ هِشَامِ بْنِ عُرْوَةَ عَنْ أَبِيهِ Buckwalter Transliteration: 464 - Had~avaniy yaHoyaY Eano maAlik Eano hi$aAmi boni Eurowapa Eano >abiyhi	
FFNN 100-Hot	Correct Diacritization	فَإِذَا ادَّعَى شَخْصٌ عَلَى غَيْرِهِ بِأَنَّهُ رَقِيقٌ فَعَلَيْهِ الْبَيَانُ . Buckwalter Transliteration: fa<i*aA Ad~aEaY $axoSN EalaY gayorihi bi>an~ahu raqiyqN faEalayohi AlobayaAnu .	الروض الأنف
	Model's Output	فَإِذَا ادَّعَى شَخْصٌ عَلَى غَيْرِهِ بِأَنَّهُ رَقِيقٌ فَعَلَيْهِ الْبَيَانُ . Buckwalter Transliteration: fa<i*aA Ad~aEaY $axoSN EalaY gayorihi bi>an~ahu raqiyqN faEalayohi AlobayaAnu .	
FFNN Embeddings	Correct Diacritization	وَالْهَاوَنُ مِثَالٌ ، فَمِثْلُهُ كُلُّ مَا يَتَعَذَّرُ كَسْرُهُ عَلَى رَأْسِهَا . Buckwalter Transliteration: waAlohaAwanu mivaAlN ، famivoluhu kul~u maA yataEa*~aru kasoruhu EalaY ra>osihaA .	مبعث النبي صلى الله عليه وسلم
	Model's Output	وَالْهَاوَنُ مِثَالٌ ، فَمِثْلُهُ كُلُّ مَا يَتَعَذَّرُ كَسْرُهُ عَلَى رَأْسِهَا . Buckwalter Transliteration: waAlohaAwanu mivaAlN ، famivoluhu kul~u maA yataEa*~aru kasoruhu EalaY ra>osihaA .	

Fig. 3. FFNN models good diacritization examples.

Model		Worst Line	File
FFNN Basic	Correct Diacritization	أَنْصَابِ مَكَّةَ عَامِدِينَ لِيَثْرِبِ ... فِي ذِي غَيَاطِلَ جَحْفَلٍ جَبْجَابِ Buckwalter Transliteration: >anoSaAbi mak~pa EaAmidiyna liyavoribi ... fiy *iy gayaATila jaHofaIK jabojaAbi	سيرة ابن هشام
	Model's Output	أَنْصَابَ مَكَّةَ عَامِدِيْنَ لِيُثْرِبُ ... فِي ذِي غَيَاطُلَ جَحْفَلُ جَبْجَابِ Buckwalter Transliteration: >noSaAba mak~apa EaAmid~iyoni liyuviribu ... fiy *iy giyaATulu jaHafalu jabojaAabK	
FFNN 100-Hot	Correct Diacritization	لَوْلَا جَرِيرٌ هَلَكَتْ بَجِيلَهْ ... نِعْمَ الْفَتَى ، وَبِئْسَ الْقَبِيلَهْ Buckwalter Transliteration: lawolaA jariyrN halakato bajiylaho ... niEoma AlofataY ، wabi}osa Aloqabiylaho	الروض الأنف
	Model's Output	لَوْلَا جَرِيرٌ هَلَكَتْ بَجِيلَهْ ... نَعَمْ الْفَتَى ، وَبِئْسَ الْقَبِيلَهْ Buckwalter Transliteration: lawalaA jariyra halakato bijayolihi ... naEamo AlofataY ، wabi}isa Aloqabiyliho	
FFNN Embeddings	Correct Diacritization	فَكُلّ صَدِيقٍ وَابْنِ أُخْتٍ نَعُدّهُ لَعَمْرِي وَجَدْنَا غِبّهُ غَيْرَ طَائِلِ Buckwalter Transliteration: fakul~ SadiyqK waAboni >uxotK naEud~hu laEamoriy wajadonaA gib~hu gayora TaA}ili	مبعث النبي صلى الله عليه وسلم
	Model's Output	فَكُلُّ صَدِيقٍ وَابْنُ أُخْتٍ نَعُدُّهُ لِعُمْرِي وَجَدْنَا غِبَّهُ غَيْرُ طَائِلٍ Buckwalter Transliteration: fakul~u SadiyqK waAbonu >uxoti naEoduhu liEumoriy wajadonaA gabohi gayoru TaA}ilK	

Fig. 4. FFNN models bad diacritization examples.

The comparison method for calculating DER/WER without case ending skips comparing the diacritization on the end of each word. This skip improves the best DER to 1.34% (vs 1.69%) and best WER to 2.91% (vs 5.09%), which is a 26% improvement in DER and 43% improvement in WER. This is because the diacritic of the last character of the word usually depends on the part of speech

Table 6. DER/WER Comparison of the Different RNN Models on the Test Set

DER/WER	w/case ending	w/o case ending	w/case ending	w/o case ending
	Including 'no diacritic'		Excluding 'no diacritic'	
Without Extra Train Dataset				
Basic model	2.68%/7.91%	2.19%/4.79%	3.09%/7.61%	2.51%/4.66%
CRF model	2.67%/7.73%	2.19%/4.69%	3.08%/7.46%	2.52%/4.60%
BNG model	**2.60%/7.69%**	**2.11%/4.57%**	**3.00%/7.39%**	**2.42%/4.44%**
With Extra Train Dataset				
Basic model	1.72%/5.16%	1.37%/2.98%	1.99%/4.96%	1.59%/2.92%
CRF model	1.84%/5.42%	1.47%/3.17%	2.13%/5.22%	1.69%/3.09%
BNG model	**1.69%/5.09%**	**1.34%/2.91%**	**1.95%/4.89%**	**1.54%/2.83%**

The best results in these tables are shown in bold font.

Table 7. DER/WER Comparison Showing the Effect of the Weights Averaging Technique on BNG Model

DER/WER	Averaged Epochs	w/case ending	w/o case ending	w/case ending	w/o case ending
		Including 'no diacritic'		Excluding 'no diacritic'	
Without extra train dataset	1	2.73%/8.08%	2.21%/4.80%	3.16%/7.79%	2.54%/4.68%
	5	2.64%/7.80%	2.14%/4.64%	3.04%/7.49%	2.46%/4.52%
	10	**2.60%/7.69%**	**2.11%**/4.57%	**3.00%/ 7.39%**	**2.42%/4.44%**
	20	2.61%/7.73%	**2.11%/4.56%**	3.01%/7.42%	**2.42%**/7.42%
With extra train dataset	1	1.97%/5.85%	1.61%/3.55%	2.20%/5.61%	1.82%/3.45%
	5	1.73%/5.20%	1.38%/3.02%	1.98%/4.98%	1.58%/2.92%
	10	1.70%/5.13%	1.35%/2.94%	1.96%/4.92%	1.55%/2.85%
	20	**1.69%/5.09%**	**1.34%/2.91%**	**1.95%/4.89%**	**1.54%/2.83%**

The best results in these tables are shown in bold font.

Arabic sentence	هذا هو كتابُه	قرأ كتابَه	مكتوب في كتابِه
English translation	This is his book	He read his book	It's written in his book
Part of speech tag	Subject	Object	Genitive

Fig. 5. Case ending different diacritization with different part of speech tag.

tag making it harder to diacritize. However, we note that the actual last character of the word may come before the end of the word if the word has some suffix appended to it.

Consider the example shown in Figure 5. The word "كتابه" means "his book" where the last character "ـه" is the suffix representing the pronoun "his", and the letter before it may take three different diacritics depending on its part of speech tagging.

Figure 6 shows the validation DER of each model every five epochs of training. This figure clearly shows the importance of the dataset size as all models trained with the extra train dataset have significantly better DER compared with all models trained without it.

Moreover, to explore the embeddings learnt by our best model, the weights vectors from the embeddings layer are extracted and reduced to two dimensions instead of 25 using t-SNE dimensionality reduction algorithm [28]. Then, they are plotted in 2D space as shown in Figure 7. The embeddings are able to capture meaningful information where digits appear together at the bottom-left, the majority of the punctuations appear at the middle and the top-left side, and finally, the Arabic letters appear at the right side.

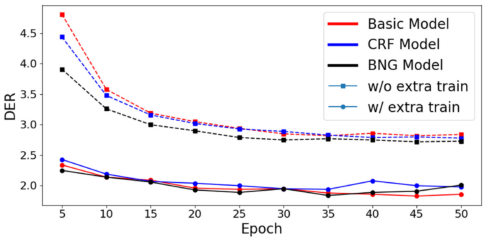

Fig. 6. RNN models validation DER while training.

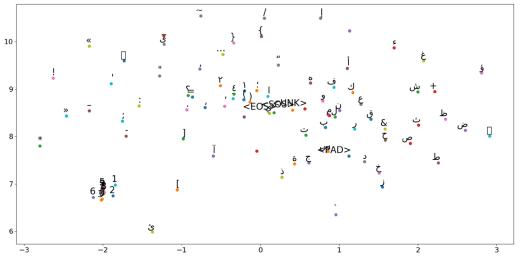

Fig. 7. Embeddings plotted in 2D space.

Model		Best Line	File
RNN Basic	Correct Diacritization	قَوْلُهُ : (وَبَحَثَ الرَّافِعِيُّ صِحَّتَهَا) وَإِنْ قَصَدَ تَمْلِيكَ الْمَسْجِدِ وَهُوَ الْمُعْتَمَدُ Buckwalter Transliteration: qawoluhu : (wabaHava Alr~aAfiEiy~u SiH~atahaA) wa<ino qaSada tamoliyka Alomasojidi wahuwa AlomuEotamadu	حاشية البجيرمي على الخطيب
	Model's Output	قَوْلُهُ : (وَبَحَثَ الرَّافِعِيُّ صِحَّتَهَا) وَإِنْ قَصَدَ تَمْلِيكَ الْمَسْجِدِ وَهُوَ الْمُعْتَمَدُ Buckwalter Transliteration: lilomurotahini qabola Huluwli >ajali Ald~ayoni bi>ano qay~adahaA bizamanK >awo EamalK yanoqaDiy qabolahu	
RNN CRF	Correct Diacritization	لِلْمُرْتَهِنِ قَبْلَ حُلُولِ أَجَلِ الدَّيْنِ بِأَنْ قَيَّدَهَا بِزَمَنٍ أَوْ عَمَلٍ يَنْقَضِي قَبْلَهُ Buckwalter Transliteration: lilomurotahini qabola Huluwli >ajali Ald~ayoni bi>ano qay~adahaA bizamanK >awo EamalK yanoqaDiy qabolahu	منح الجليل شرح مختصر خليل
	Model's Output	لِلْمُرْتَهِنِ قَبْلَ حُلُولِ أَجَلِ الدَّيْنِ بِأَنْ قَيَّدَهَا بِزَمَنٍ أَوْ عَمَلٍ يَنْقَضِي قَبْلَهُ Buckwalter Transliteration: qawoluhu : (wabaHava Alr~aAfiEiy~u SiH~atahaA) wa<ino qaSada tamoliyka Alomasojidi wahuwa AlomuEotamadu	
RNN BNG	Correct Diacritization	وَلِذَا قَالَ وَلَوْ لَمْ يَقُلْ أَيْ الْمُوَثِّقُ إنْ فَعَلَ شَيْئًا مِنْهَا بِأَنْ قَالَ Buckwalter Transliteration: wali*aA qaAla walawo lamo yaqulo >ayo Alomuwav~iqu <no faEala $ayo}FA minohaA bi>ano qaAla	شرح مختصر خليل للخرشي
	Model's Output	وَلِذَا قَالَ وَلَوْ لَمْ يَقُلْ أَيْ الْمُوَثِّقُ إنْ فَعَلَ شَيْئًا مِنْهَا بِأَنْ قَالَ Buckwalter Transliteration: wali*aA qaAla walawo lamo yaqulo >ayo Alomuwav~iqu <no faEala $ayo}FA minohaA bi>ano qaAla	

Fig. 8. RNN models good diacritization examples.

Figures 8 and 9 show both best and worst examples of diacritization using each RNN model. An important note is that the old Arabic poetry lines no longer represent the majority of the worst examples, in contrast to the FFNN models.

Finally, Figures 10 and 11 show the confusion matrices related to our best model when trained without and with the extra train dataset, respectively. By comparing them, it is easy to see that the Shadda class is the worst one in both cases. However, the case with the extra train dataset shows dramatic improvement for this class, as well as other classes like Shadda + another diacritic and the Dammatan. A justification for this improvement is that there are a larger number of examples in the extra train dataset related to these classes as shown in Table 8. Another insight can be concluded from the confusion matrices is that the model usually misclassifies the Shadda class as Shadda + another diacritic class due to different diacritization conventions, which in many cases would be a grammatically correct guess.

Model		Worst Line	File
RNN Basic	Correct Diacriti-zation	وَبَكَّيهِ لِلْأَيْتَامِ وَالرِّيحُ زَفْزَةٌ ... وَتَشْبِيبُ قِدْرٍ طَالَمَا أَزْبَدَتْ تَغْلِي Buckwalter Transliteration: wabak~yhi lilo>ayotaAmi waAlr~yHu zafozapN ... wata$obiybu qidorK TaAlamaA >azobadato tagoliy	سيرة ابن هشام
	Model's Output	وَبَكَيْهُ لِلْأَيْتَامِ وَالرِّيحِ زَفْزَةٌ ... وَتَشْبِيبُ قَدْرٍ طَالَمَا أَزْبَدَتْ تَغْلِي Buckwalter Transliteration: wabakayohu lilo>ayotaAmi waAlr~iyHi zafozapu ... wata$obiybu qadorK TaAlimFA >azobadoto tagoliy	
RNN CRF	Correct Diacriti-zation	وَنَقَلَ فِي مَوْضِعٍ آخَرَ تَعْلِيلَهُ بِأَنَّهُ لَا يَقْلَعُ النَّجَسَ لِلزُّوجَتِهِ . Buckwalter Transliteration: wanaqala fiy mawoDEK |xara taEoliylahu bi>an~ahu laA yuqolEu Aln~ajasa liluzuwjatihi .	شرح البهجة الوردية
	Model's Output	وَنُقِلَ فِي مَوْضِعٍ آخَرَ تَعْلِيلُهُ بِأَنَّهُ لَا يَقْلَعُ النَّجِسَ لِلزَّوْجَتِهِ . Buckwalter Transliteration: wanuqila fiy mawoDEK |xara taEoliyluhu bi>an~ahu laA yaqolaEu Aln~ajisa lilz~awojatahi .	
RNN BNG	Correct Diacriti-zation	وَنَقَلَ فِي مَوْضِعٍ آخَرَ تَعْلِيلَهُ بِأَنَّهُ لَا يَقْلَعُ النَّجَسَ لِلزُّوجَتِهِ . Buckwalter Transliteration: wanaqala fiy mawoDEK |xara taEoliylahu bi>an~ahu laA yuqolEu Aln~ajasa liluzuwjatihi .	شرح البهجة الوردية
	Model's Output	وَنُقِلَ فِي مَوْضِعٍ آخَرَ تَعْلِيلُهُ بِأَنَّهُ لَا يَقْلَعُ النَّجِسَ لِلزَّوْجَتِهِ . Buckwalter Transliteration: wanuqila fiy mawoDEK |xara taEoliyluhu bi>an~ahu laA yaqolaEu Aln~ajisa lilz~awojataho .	

Fig. 9. RNN models bad diacritization examples.

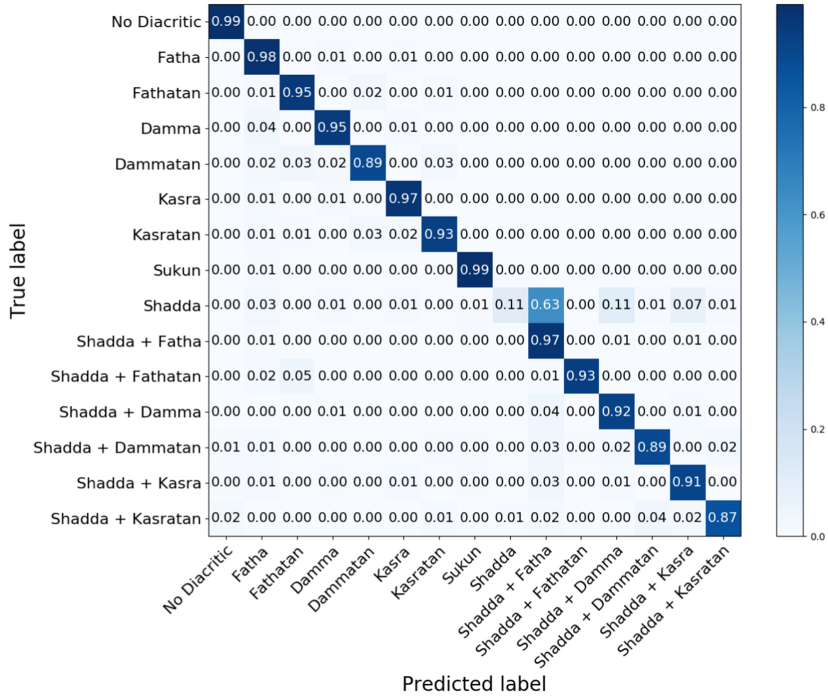

Fig. 10. Without extra train confusion matrix for the best BNG model.

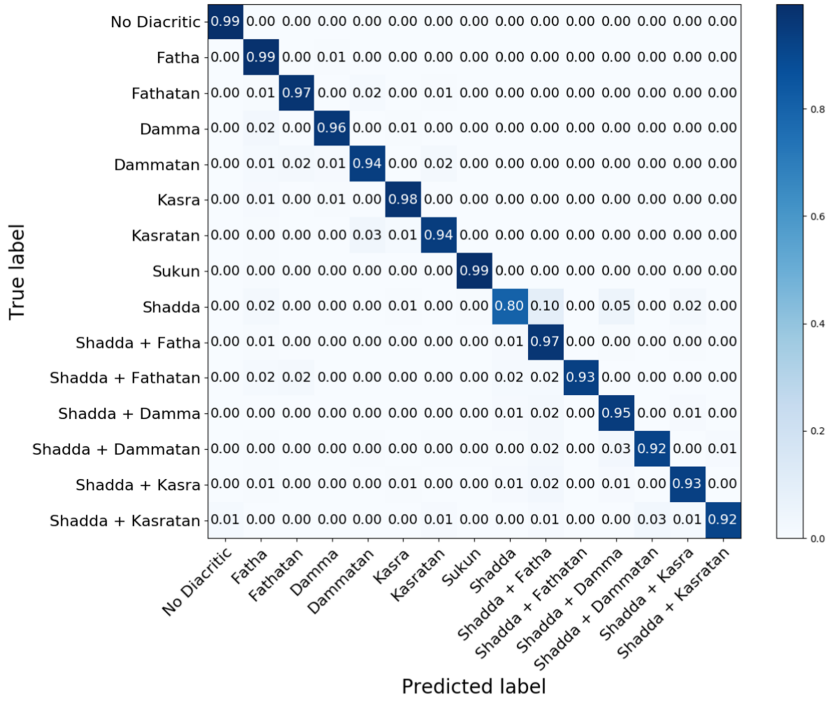

Fig. 11. With extra train confusion matrix for the best BNG model.

Table 8. Number of Examples for Each Class

	Train	Valid	Test	Extra Train	Total	%
No Diacritic	4,366 K	213 K	222 K	46,647 K	51,449 K	38.87
Fatha	2,932 K	144 K	150 K	31,287 K	34,514 K	26.07
Fathatan	58 K	3 K	3 K	626 K	691 K	00.52
Damma	812 K	39 K	41 K	8,648 K	9,539 K	07.20
Dammatan	58 K	3 K	3 K	622 K	686 K	00.51
Kasra	1,265 K	62 K	64 K	13,533 K	14,924 K	11.27
Kasratan	88 K	4 K	4 K	941 K	1,037 K	00.78
Sukun	1,230 K	60 K	63 K	13,135 K	14,487 K	10.94
Shaddah	6 K	254	471	66 K	73 K	00.05
Shaddah + Fatha	300 K	15 K	15 K	3,202 K	3,532 K	02.66
Shaddah + Fathatan	3 K	189	132	36 K	40 K	00.03
Shaddah + Damma	43 K	2 K	2 K	463 K	511 K	00.38
Shaddah + Dammatan	5 K	238	222	51 K	56 K	00.04
Shaddah + Kasra	64 K	3 K	3 K	679 K	749 K	00.56
Shaddah + Kasratan	6 K	298	273	63 K	69 K	00.05

Table 9. Comparing the BNG Model with [12] in Terms of DER/WER on the Test Set

DER/WER	w/case ending	w/o case ending	w/case ending	w/o case ending
	Including "no diacritic"		Excluding "no diacritic"	
[24] Testing Dataset Results				
Our best model	**1.78%/5.38%**	**1.39%/3.04%**	**2.05%/5.17%**	**1.60%/2.96%**
Shakkala model [12]	3.73%/11.19%	2.88%/6.53%	4.36%/10.89%	3.33%/6.37%
Auxiliary Testing Dataset Results				
Our best model	**5.98%/15.72%**	5.21%/11.07%	**5.54%/13.21%**	**4.85%/9.02%**
Shakkala model [12]	6.41%/17.52%	**5.12%/10.91%**	6.82%/15.92%	5.32%/9.65%

The best results in these tables are shown in bold font.

Furthermore, an Encoder–Decoder structure **sequence-to-sequence** (**seq2seq**) was built using BiCuDNNLSTMs to encode a sequence of characters and generate a sequence of diacritics, but the model was not able to successfully learn the alignment between inputted characters and outputted diacritics. Other attempts include trying to encode the sentences as sequences of words and generate sequences of diacritics performed terribly and failed to learn the correct diacritization.

The BNG model performs the best compared with other models described so far. So, it is used for comparison with other systems in the following section.

5.3 Comparison with Existing Systems

As mentioned in Section 2, we limit our attention in this work to neural ATD models. According to [24], the Shakkala system [12] performs the best compared to other existing systems using the test set and the evaluation metrics proposed in [24]. Considering our best model's results mentioned previously, it is clear that our model outperforms Shakkala on the testing set after splitting the lines to be at most 315 characters long (Shakkala system limit), which causes a slight drop in our best model's results. However, since Shakkala was also trained on Tashkeela Corpus, we develop an auxiliary test set extracted from three books from Al-Shamela Library[10] : "تاج العروس من جواهر القاموس", "الفتاوى الكبرى لابن تيمية", and "فتح الباري شرح صحيح البخاري" using the same extraction and cleaning method proposed by [24] while keeping only lines with more than 80% "diacritics to Arabic characters" ratio. The extracted lines are each split into lines of lengths no more than 315 characters (without counting diacritics), which is the input limit of the Shakkala system. This produces a test set consisting of 443 K words. Table 9 gives the results comparison with Shakkala.

A comparison with the pre-trained model of [14] is also done using the test set and the evaluation metrics of [24] while splitting the lines into lines of lengths no more than 125 characters (without counting diacritics) since any input with length more than that causes an error in [14]'s system. The results show that [14]'s system performs poorly. However, we note that [14]'s system was trained and tested on the ATB dataset, which consists of text in MSA. So, to make a fair comparison with [14]'s system, an auxiliary dataset is built from the MSA part of the Tashkeela Corpus using the same extraction and cleaning method proposed by [24] keeping only lines with more than 80% "diacritics to Arabic characters" ratio. This test set consists of 111 K words. The results are reported in Table 10. In addition to the poor results of [14]'s system, its output has a large number of special characters inserted randomly. These characters are removed manually to make the evaluation of the system possible.

Finally, we compare our model with [2]'s model which, to our best knowledge, is the most recent deep learning work announcing the best results so far. To do so, we employ a similar comparison method to [17]'s by using the 10 books from the Tashkeela Corpus and the Holy Quran that were

[10] http://shamela.ws.

Table 10. Comparing the BNG Model with [14] in Terms of DER/WER on the Test Set

DER/WER	w/case ending	w/o case ending	w/case ending	w/o case ending
	Including 'no diacritic'		Excluding 'no diacritic'	
Classical Arabic Testing Dataset Results				
Our best model	**1.99%/6.10%**	**1.48%/3.25%**	**2.30%/5.88%**	**1.70%/3.17%**
Belinkov, 2015	31.26%/75.29%	29.66%/59.46%	35.78%/74.37%	33.67%/57.66%
Modern Standard Arabic Testing Dataset Results				
Our best model	**8.05%/23.56%**	**6.85%/16.12%**	**8.29%/21.10%**	**7.16%/14.41%**
Belinkov, 2015	31.77%/75.02%	29.21%/59.40%	37.13%/73.93%	33.82%/58.03%

The best results in these tables are shown in bold font.

Table 11. Comparing the BNG Model with [2] in Terms of DER/WER on the Test Set

	DER		WER	
	w/case ending	w/o case ending	w/case ending	w/o case ending
Our best model	2.18%	1.76%	**4.44%**	**2.66%**
Abandah, 2015	**2.09%**	**1.28%**	5.82%	3.54%

The best results in these tables are shown in bold font.

excluded from [2]'s test set. The sentences used for testing our best model are all sentences that are not included in the training dataset of [24] or extra training dataset on which our model is trained. To make the comparison fair, we use the same evaluation metric as [2], which is [38]'s. Moreover, the characters with no diacritics in the original text are skipped similarly to [2]. The results are shown in Table 11. It is worth mentioning that the results of [2] include post-processing techniques, which improved DER by 23.8% as reported in [2]. It can be easily shown that, without this step, our model's results are actually superior.

All codes related to the diacritization work are publicly available on GitHub,[11] and are also implemented into a web application[12] for testing purposes.

6 USING DIACRITIZATION TO ENHANCE DOWNSTREAM TASKS

In this section, we describe our novel approach to integrate diacritization into two different downstream tasks for Arabic NLP. Specifically, these tasks are MT (Section 6.1) and SA (Section 6.2.).

6.1 Translation Over Diacritization (ToD)

Word's diacritics can carry various types of information about the word itself, like its part of speech tag, the semantic meaning and the pronunciation. Intuitively, providing such extra features in NLP tasks has the potential to improve the results of any system. In this section, we show how we benefit from the integration of diacritics into Arabic-English (Ar-En) **Neural Machine Translation** (**NMT**) creating what we call ToD.

Dataset Extraction and Preparation. Due to the scarcity of free standardized benchmark parallel corpora for sentence-level Ar-En MT, we create a mid-size dataset using the following corpora: GlobalVoices v2017q3, MultiUN v1, News-Commentary v11, Tatoeba v2, TED2013 v1.1, Ubuntu v14.10, and Wikipedia v1.0 [35] downloaded from the OPUS[13] project. The dataset contains 1 M Ar-En sentence pairs split into 990 K pairs for training and 10 K pairs for testing. The extracted

[11] https://github.com/AliOsm/shakkelha.
[12] https://shakkelha.herokuapp.com.
[13] http://opus.nlpl.eu.

Table 12. Vocab Size for All Sequences Types before and after BPE Step

Language	Vocab Size	
	Before BPE	After BPE
English	113 K	31 K
Original Arabic	224 K	32 K
Diacritized Arabic	402 K	186 K
Diacritics Forms	41 K	15 K

1 M pairs follow these conventions: (i) The maximum length for each sentence in the pair is 50 tokens, (ii) Arabic sentences contain Arabic letters only, (iii) English sentences contain English letters only, and (iv) the sentences do not contain any URLs.

The Arabic sentences in the training and testing datasets are diacritized using the best BNG model. After that, **Byte Pair Encoding** (**BPE**)[14] [33] is applied separately on both English and original (undiacritized) Arabic sequences to segment the words into subwords with 32 K merge operations. This step overcomes the **Out Of Vocabulary** (**OOV**) problem and reduces the vocabulary size. Then, diacritics are added to Arabic subwords to create the diacritized version. Table 12 shows the number of tokens before and after BPE step for English, Original Arabic, and Diacritized Arabic as well as the Diacritics forms when removing the Arabic characters.

Model Structure. The model used in the experiments is a basic Encoder–Decoder seq2seq model that consists of a BiCuDNNLSTM layer for encoding and a CuDNNLSTM layer for decoding with 512 units each (256 per direction for the encoder) while applying additive attention [11] on the outputs of the encoder. As for the embeddings layer, a single randomly initialized embeddings layer with vector size 64 is used to represent the subwords when training without diacritics. Another layer with the same configuration is used to represent subwords' diacritics, which is concatenated with the subwords embeddings when training with diacritics. The model structure shown in Figure 12.

Results and Discussion. To explore the effect of the Arabic diacritization on the NMT task, we experiment with training both with and without diacritics. The models are trained for 50 epochs using an Nvidia Titan Xp GPU, Adam optimization algorithm with 0.001 learning rate, 0.9 beta1, 0.999 beta2, 10^{-7} epsilon, and 256 batch size.

The structure for training the model with diacritics may vary. We experiment with two variations where the first one uses the diacritized version of the sequences, while the other one uses the original sequences and the diacritics sequences in parallel. When merging diacritics with their sequences, we get more variations of each word depending on its different forms of diacritization, therefore expanding the vocabulary size. On the other hand, when separating diacritics from their sequences, the vocab size stays the same, and diacritics are added separately as extra input.

The results in Table 13 show that training the model with diacritization compared to without diacritization improves marginally by 0.31 BLEU score[15] when using the "with diacritics (merged)" data and improves even more when using the "with diacritics (separated)" data by 1.33 BLEU score. Moreover, the training time and model size increases by about 20.6% and 41.4%, respectively, for using the "with diacritics (merged)" data, while they only increase by about 3.4% and 4.5%, respectively, for using the "with diacritics (separated)" data. By observing Figure 13, which reports the BLEU score on all three models every five epochs, it is clear that, although the "with diacritics

[14] https://github.com/rsennrich/subword-nmt.

[15] BLEU scores are computed with multi-bleu.perl.

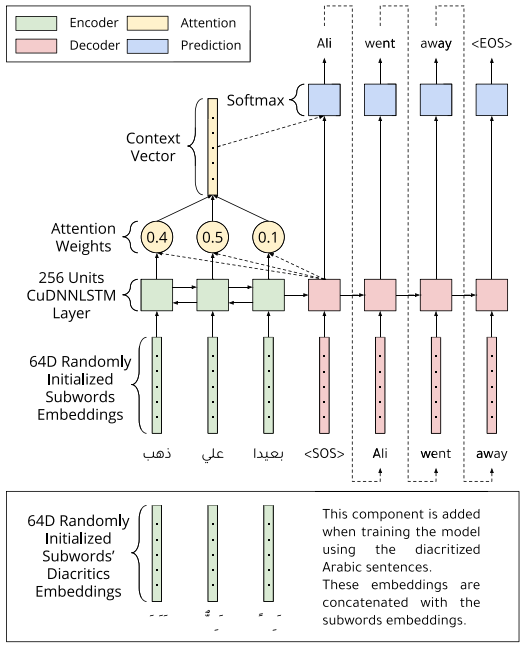

Fig. 12. ToD model structure.

(merged)" model converges better at the start of the training, it starts diverging after 15 epochs, which might be due to the huge vocab size and the training data size.

By analysing Figure 13, we find that BLUE score converges faster when training with diacritics (merged) compared to the other two approaches. However, it starts diverging later on due to vocabulary sparsity. As for with diacritics (separated), the BLUE score has higher convergence compared to without diacritics while also maintaining stability compared to with diacritics (merged). This is because separating diacritics solves the vocabulary sparsity issue while also providing the information needed to disambiguate homonym words.

We note that, concurrently to our work, another work on utilizing diacritization for MT has recently appeared. Ref. [7] used diacritics with text in three downstream tasks, namely, **Semantic Text Similarity** (**STS**), NMT, and **Part of Speech** (**POS**) tagging, to boost the performance of their systems. They applied different techniques to disambiguate homonym words through diacritization. They achieved 27.1 and 27.3 BLUE scores without and with diacritics, respectively, using their best disambiguation technique. This is a very small improvement of 0.74% compared to our noticeable improvement of 4.03%. Moreover, our approach is simpler and it does not require to drop any diacritical information.

All codes related to the ToD work are publicly available on GitHub.[16]

6.2 Sentiment Over Diacritization (SoD)

Similarly to ToD (Section 6.1), diacritics may carry sentimental information about words by—for example—disambiguating neutral words like "شَره" which means "gluttonous" and words that may imply certain emotions like "شَرُّهُ", which means "his evil".

[16] https://github.com/AliOsm/translation-over-diacritization.

Table 13. ToD Results on the Test Set

Model	Training Time	Model Size	Best BLEU Score
Without	**29 Hours**	**285 MB**	33.01
Merged	35 Hours	403 MB	33.32
Separated	30 Hours	298 MB	**34.34**

The best results in these tables are shown in bold font.

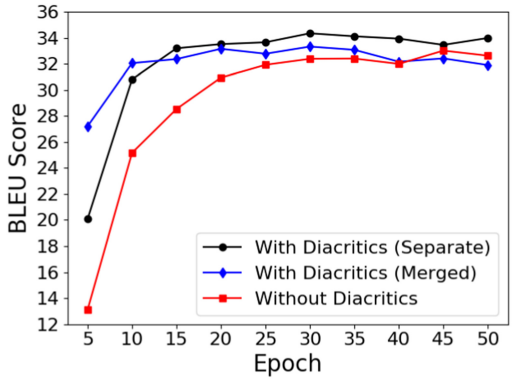

Fig. 13. Testing dataset BLEU score while training.

Dataset Extraction and Preparation. To evaluate our approach, we used eight datasets from [21–23]. The datasets can be divided into three categories: (i) Small datasets: which have less that 1,000 testing examples, (ii) Medium datasets: which have less than 10,000 testing examples, and (iii) Large datasets: which have more than 10,000 testing examples. To prepare the datasets, existing diacritics were removed, duplicated consecutive characters with count more than two were reduced to two characters only, examples with more than 256 tokens were truncated to 256 tokens only, and examples that contain any English characters were excluded. After that, we diacritized the datasets using the BNG model (Section 4.2) to obtain a diacritized version of the datasets. Each dataset was split based on the classes, where 90% and 10% of each class goes to training and testing datasets, respectively. Table 14 shows the statistics of these datasets.

Model Structure. The model used in the experiments is a basic RNN model that consists of two BiCuDNNLSTM layers with 256 units each (128 per direction) followed by two fully connected layers of size 256 and 128, respectively. The input words were converted to feature vectors using Arabic FastText [15] pretrained model.[17]

To merge the diacritical information for each word into its representation, we introduced the merging component. It is a single BiCuDNNLSTM layer of size 100 units that takes the diacritics for each word separately and builds a vector representation for the sequence of diacritics. Then, concatenate it to the word representation extracted from FastText model. Figure 14 shows the architecture of this component.

Results and Discussion. To explore the effect of diacritics on this task, we trained the model described above twice for each dataset. The first training was done without the merging component.

[17] https://fasttext.cc/docs/en/crawl-vectors.html.

Table 14. SoD Datasets Statistics

Dataset Name	Size	#Train	#Test	#Classes	Avg. Words	Class 1	Class 2	Class 3	Class 4	Class 5
Attraction Reviews TripAdvisor	Small	1,845	204	2	38.78	03.75%	96.25%	N/A	N/A	N/A
Product Reviews Souq	Small	3,403	376	3	12.17	20.53%	07.12%	72.35%	N/A	N/A
Hotel Reviews TripAdvisor	Medium	12,510	1,389	3	93.28	17.39%	13.90%	68.71%	N/A	N/A
Restaurant Reviews TripAdvisor & Qaym	Medium	9,042	1,003	3	36.41	24.58%	02.50%	72.93%	N/A	N/A
Balanced Book Reviews GoodReads	Large	140,856	15,650	2	74.05	50.08%	49.92%	N/A	N/A	N/A
Unbalanced Book Reviews GoodReads	Large	434,405	48,265	5	73.32	06.11%	09.20%	20.86%	30.98%	32.84%
Balanced Hotel Reviews Booking	Large	92,690	10,297	2	23.46	49.95%	50.05%	N/A	N/A	N/A
Unbalanced Hotel Reviews Booking	Large	359,665	39,960	5	21.21	03.50%	09.37%	19.63%	32.32%	35.18%

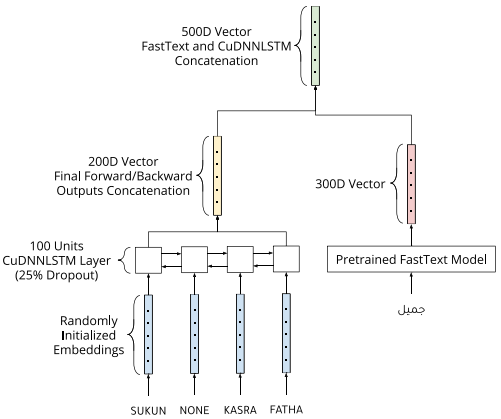

Fig. 14. SoD diacritics merging component.

That is, we converted each word to its FastText representation and fed it to the model as an input. In the second training, we used the merging component to build an enriched word representation for each word by concatenating FastText representation with the word's diacritics representation. The model was trained for 10, 25, and 50 epochs for the small, medium, and large datasets, respectively, on Google Colab using an Nvidia Tesla K80 GPU, Adam optimization algorithm with 0.001 learning rate, 0.9 beta1, and 0.999 beta2, 10^{-7} epsilon and a batch size of 32 for both the small and medium datasets, while using a batch size of 64 with the large datasets.

Table 15 shows the training time, evaluation time, accuracy, and F1-score results for each one of the datasets' test set without and with the merging diacritics component. The small datasets results did not change by a significant margin, for the accuracy it did not change at all, while the change in the F1-score results were less than 0.1. The similar results for the small datasets could be due to the small size of the training and testing datasets. So, the merging diacritics component might not have learnt a lot of information from the small training data thus, not able to build useful diacratical information representations.

Table 15. SoD Results on the Test Sets

Dataset Name	Size	Training Time		Evaluation Time		Accuracy		F1-score	
		w/o diacs	w/diacs	w/o diacs	w/diacs	w/o diacs	w/diacs	w/o diacs	w/diacs
Attraction Reviews TripAdvisor	Small	30 second	50 second	7 second	9 second	97.06%	97.06%	98.46%	**98.48%**
Product Reviews Souq	Small	30 second	40 second	6 second	8 second	82.45%	82.45%	**79.14%**	79.08%
Hotel Reviews TripAdvisor	Medium	13.3 meter	25.8 meter	50 second	58 second	83.80%	**84.88%**	82.91%	**83.65%**
Restaurant Reviews TripAdvisor and Qaym	Medium	5 meter	9.6 meter	19 second	25 second	90.13%	**90.33%**	89.01%	**89.51%**
Balanced Book Reviews GoodReads	Large	2.2 hour	5.6 hour	303 second	377 second	**86.8%**	86.37%	**86.9%**	86.56%
Unbalanced Book Reviews GoodReads	Large	6.9 hour	18.2 hour	895 second	1122 second	52.50%	**53.25%**	51.61%	**52.27%**
Balanced Hotel Reviews Booking	Large	0.7 hour	1.7 hour	97 second	151 second	95.04%	**95.29%**	95.04%	**95.30%**
Unbalanced Hotel Reviews Booking	Large	2.7 hour	6.5 hour	341 second	516 second	78.99%	**79.32%**	78.87%	**79.21%**

The best results in these tables are shown in bold font.

For the medium size datasets, the results were improved consistently with an average improvement of 0.64 on accuracy and 0.62 on F1-score on both medium datasets. Finally, with the four large datasets, the results improved on three of them and decreased only on one. The averaged improvement on the three datasets was 0.44 on accuracy and 0.42 on F1-score. The higher improvements on the medium datasets compared to the large ones could be caused by the diacritical information becoming less effective as the data grow bigger and the models are able to learn to extract enough information to make a correct classification without the need of the diacritics representations.

Regarding to the training time, the model that was trained without merging diacritics component was faster than the model that was trained with merging diacritics component by an average of 15 seconds, 8.6 minutes, and 4.9 hours for the small, medium, and large datasets, respectively. While in the evaluation phase, the with merging diacritics component model was almost as fast as the without merging diacritics component model in evaluating the small and medium datasets. But, in evaluating the large datasets, it was slower by an average of 2 minutes.

To test the improvements statistically, we conducted an McNemar–Bowker test on the predictions from the "Unbalanced Hotel Reviews Booking" dataset. The test showed that the results are statistically highly significant with a p value below the 0.05 threshold.

All codes related to the SoD work are publicly available on GitHub.[18]

7 CONCLUSION

In this work, we explored the ATD problem. Our models, which follow two main approaches: FFNN and RNN, proved to be very effective as they performed on par with or better than SOTA approaches. In the future, we plan on investigating the seq2seq models such as RNN Seq2seq, Conv Seq2seq, and Transformer. Moreover, we plan on exploring different ways to leveraging additional information, such as morphological information, to improve the performance of neural ATD models.

In another contribution of this work, we showed that diacritics can be integrated into other systems to attain enhanced versions in NLP tasks. We used MT and SA as a case study and showed how our ideas of ToD and SoD improved the results of the models in the two tasks. Exploring the benefits of diacritization for other NLP tasks is a subject of future research.

[18] https://github.com/AliOsm/sentiment-over-diacritization.

APPENDIX

A DETAILED STRUCTURES OF PROPOSED MODELS

Table 16. FFNN Basic Model Structure

Layer Name	Neurons	Activation Func
Hidden 1	200	ReLU
Hidden 2	500	ReLU
Hidden 3	500	ReLU
Hidden 4	450	ReLU
Hidden 5	400	ReLU
Hidden 6	400	ReLU
Hidden 7	350	ReLU
Hidden 8	300	ReLU
Hidden 9	300	ReLU
Hidden 10	250	ReLU
Hidden 11	200	ReLU
Hidden 12	200	ReLU
Hidden 13	150	ReLU
Hidden 14	100	ReLU
Hidden 15	100	ReLU
Hidden 16	50	ReLU
Hidden 17	25	ReLU
Output	15	Softmax
Trainable Parameters: 1,501,115		

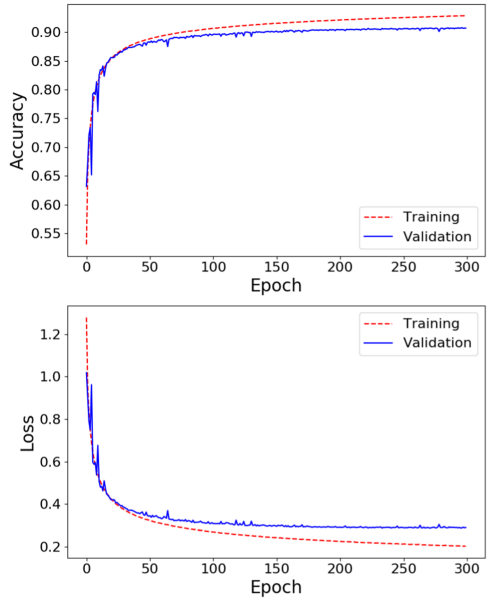

Fig. 15. FFNN basic model training and validation accuracy and loss.

Table 17. FFNN 100-Hot Model Structure

Layer Name	Neurons	Activation Func
One Hot	N/A	N/A
Flatten	N/A	N/A
Dropout 1 (2.5%)	N/A	N/A
Hidden 1	250	ReLU
Dropout 2 (2.5%)	N/A	N/A
Hidden 2	200	ReLU
Dropout 3 (2.5%)	N/A	N/A
Hidden 3	150	ReLU
Dropout 4 (2.5%)	N/A	N/A
Hidden 4	100	ReLU
Dropout 5 (2.5%)	N/A	N/A
Hidden 5	50	ReLU
Dropout 6 (2.5%)	N/A	N/A
Output	15	Softmax
Trainable Parameters: 1,951,515		

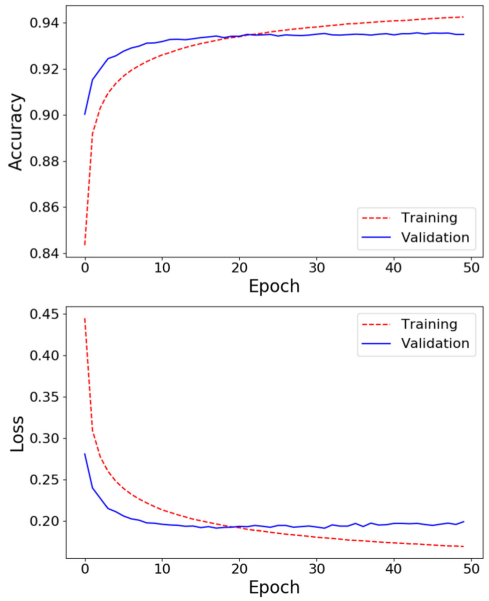

Fig. 16. FFNN 100-Hot model training and validation accuracy and loss.

Table 18. FFNN Embeddings Model Structure

Layer Name	Neurons	Activation Func
Embedding (25)	N/A	N/A
Flatten	N/A	N/A
Dropout (10%)	N/A	N/A
Hidden 1	250	ReLU
Hidden 2	200	ReLU
Hidden 3	150	ReLU
Hidden 4	100	ReLU
Hidden 5	50	ReLU
Output	15	Softmax
Trainable Parameters: 728,590		

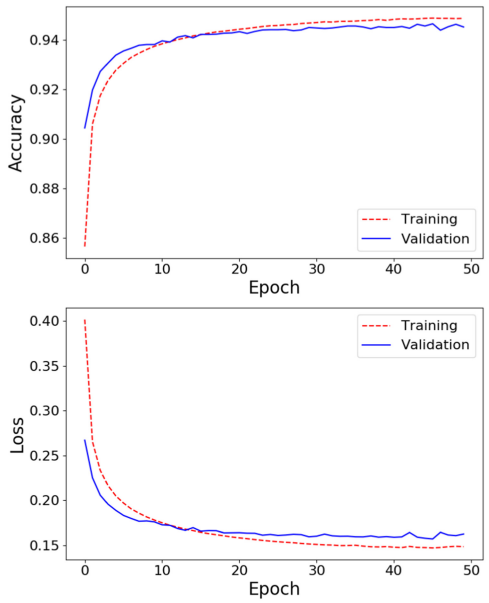

Fig. 17. FFNN Embeddings model training and validation accuracy and loss.

REFERENCES

[1] Gheith Abandah and Asma Abdel-Karim. 2020. Accurate and fast recurrent neural network solution for the automatic diacritization of Arabic text. *Jordanian Journal of Computers and Information Technology* 6, 2 (2020), 103–121.

[2] Gheith A. Abandah, Alex Graves, Balkees Al-Shagoor, Alaa Arabiyat, Fuad Jamour, and Majid Al-Taee. 2015. Automatic diacritization of Arabic text using recurrent neural networks. *International Journal on Document Analysis and Recognition* 18, 2 (2015), 183–197.

[3] Gheith A. Abandah, Mohammed Z. Khedher, Mohammad R. Abdel-Majeed, Hamdi M. Mansour, Salma F. Hulliel, and Lara M. Bisharat. 2020. Classifying and diacritizing Arabic poems using deep recurrent neural networks. *Journal of King Saud University-Computer and Information Sciences* (2020).

[4] Mohammad Al-Smadi, Bashar Talafha, Mahmoud Al-Ayyoub, and Yaser Jararweh. 2019. Using long short-term memory deep neural networks for aspect-based sentiment analysis of Arabic reviews. *International Journal of Machine Learning and Cybernetics* 10, 8 (2019), 2163–2175.

[5] Abdulmohsen Al-Thubaity, Atheer Alkhalifa, Abdulrahman Almuhareb, and Waleed Alsanie. 2020. Arabic diacritization using bidirectional long short-term memory neural networks with conditional random fields. *IEEE Access* 8 (2020), 154984–154996.

[6] Rehab Alnefaie and Aqil M. Azmi. 2017. Automatic minimal diacritization of Arabic texts. *Procedia Computer Science* 117 (2017), 169–174.

[7] Sawsan Alqahtani, Hanan Aldarmaki, and Mona Diab. 2019. Homograph disambiguation through selective diacritic restoration. In *Proceedings of the 4th Arabic Natural Language Processing Workshop*. 49–59.

[8] Saba' Alqudah, Gheith Abandah, and Alaa Arabiyat. 2017. Investigating hybrid approaches for Arabic text diacritization with recurrent neural networks. In *Proceedings of the 2017 IEEE Jordan Conference on Applied Electrical Engineering and Computing Technologies*. IEEE, 1–6.

[9] Jeremy Appleyard, Tomas Kocisky, and Phil Blunsom. 2016. Optimizing performance of recurrent neural networks on gpus. arXiv:1604.01946 (2016). Retrieved from https://arxiv.org/abs/1604.01946.

[10] Aqil M. Azmi and Reham S. Almajed. 2015. A survey of automatic Arabic diacritization techniques. *Natural Language Engineering* 21, 3 (2015), 477–495.

[11] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR'15), San Diego, CA, USA*. http://arxiv.org/abs/1409.0473

[12] Ahmad Barqawi and Taha Zerrouki. 2017. Shakkala, Arabic Text Vocalization. Retrieved February 20, 2021 from https://github.com/Barqawiz/Shakkala.

[13] Mohamed Bebah, Chennoufi Amine, Mazroui Azzeddine, and Lakhouaja Abdelhak. 2014. Hybrid approaches for automatic vowelization of Arabic texts. arXiv:1410.2646. Retrieved from https://arxiv.org/abs/1410.2646.

[14] Yonatan Belinkov and James Glass. 2015. Arabic diacritization with recurrent neural networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. 2281–2285.

[15] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics* 5, 1 (2017), 135–146.

[16] Tiago Carneiro, Raul Victor Medeiros Da Nóbrega, Thiago Nepomuceno, Gui-Bin Bian, Victor Hugo C. De Albuquerque, and Pedro Pedrosa Reboucas Filho. 2018. Performance analysis of Google colaboratory as a tool for accelerating deep learning applications. *IEEE Access* 6 (2018), 61677–61685.

[17] Amine Chennoufi and Azzeddine Mazroui. 2017. Morphological, syntactic and diacritics rules for automatic diacritization of Arabic sentences. *Journal of King Saud University-Computer and Information Sciences* 29, 2 (2017), 156–163.

[18] Kareem Darwish, Hamdy Mubarak, and Ahmed Abdelali. 2017. Arabic diacritization: Stats, rules, and hacks. In *Proceedings of the 3rd Arabic Natural Language Processing Workshop*. 9–17.

[19] Mona Diab, Nizar Habash, Owen Rambow, and Ryan Roth. 2013. LDC Arabic treebanks and associated corpora: Data divisions manual. arXiv:1309.5652 (2013). Retrieved from https://arxiv.org/abs/1309.5652.

[20] John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research* 12, 61 (2011), 2121–2159. http://jmlr.org/papers/v12/duchi11a.html.

[21] Ashraf Elnagar and Omar Einea. 2016. Brad 1.0: Book reviews in arabic dataset. In *Proceedings of the 2016 IEEE/ACS 13th International Conference of Computer Systems and Applications*. IEEE, 1–8.

[22] Ashraf Elnagar, Yasmin S. Khalifa, and Anas Einea. 2018. Hotel Arabic-reviews dataset construction for sentiment analysis applications. In *Proceedings of the Intelligent Natural Language Processing: Trends and Applications*. Springer, 35–52.

[23] Hady ElSahar and Samhaa R. El-Beltagy. 2015. Building large arabic multi-domain resources for sentiment analysis. In *Proceedings of the International Conference on Intelligent Text Processing and Computational Linguistics*. Springer, 23–34.

[24] Ali Fadel, Ibraheem Tuffaha, Bara' Al-Jawarneh, and Mahmoud Al-Ayyoub. 2019. Arabic text diacritization using deep neural networks. In *Proceedings of the 2nd International Conference on Computer Applications & and Amp; Information Security*.

[25] Amany Fashwan and Sameh Alansary. 2017. SHAKKIL: An automatic diacritization system for modern standard Arabic texts. In *Proceedings of the 3rd Arabic Natural Language Processing Workshop*. 84–93.

[26] Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF models for sequence tagging. arXiv:1508.01991. Retrieved from https://arxiv.org/abs/1508.01991.

[27] Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR'15), San Diego, CA, USA*. http://arxiv.org/abs/1412.6980.

[28] Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of Machine Learning Research* 9, 86 (2008), 2579–2605. http://jmlr.org/papers/v9/vandermaaten08a.html.

[29] Abir Masmoudi, Salima Mdhaffar, Rahma Sellami, and Lamia Hadrich Belguith. 2019. Automatic diacritics restoration for Tunisian dialect. *ACM Transactions on Asian and Low-Resource Language Information Processing* 18, 3 (2019), 1–18.

[30] Rajae Moumen, Raddouane Chiheb, Rdouan Faizi, and Abdellatif El Afia. 2018. Evaluation of gated recurrent unit in Arabic diacritization. *International Journal of Advanced Computer Science and Applications* 9, 11 (2018). DOI: 10.14569/IJACSA.2018.091150

[31] Hamdy Mubarak, Ahmed Abdelali, Hassan Sajjad, Younes Samih, and Kareem Darwish. 2019. Highly effective Arabic diacritization using sequence to sequence modeling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. 2390–2395.

[32] Arfath Pasha, Mohamed Al-Badrashiny, Mona T. Diab, Ahmed El Kholy, Ramy Eskander, Nizar Habash, Manoj Pooleery, Owen Rambow, and Ryan Roth. 2014. Madamira: A fast, comprehensive tool for morphological analysis and disambiguation of arabic. In *Proceedings of the 9th International Conference on Language Resources and Evaluation*, Vol. 14. 1094–1101.

[33] Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, 1715–1725. https://aclanthology.org/P16-1162.

[34] Anas Shahrour, Salam Khalifa, and Nizar Habash. 2015. Improving arabic diacritization through syntactic analysis. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. 1309–1315.

[35] Jörg Tiedemann. 2012. Parallel data, tools and interfaces in OPUS. In *Proceedings of the 8th International Conference on Language Resources and Evaluation*, Vol. 2012. 2214–2218.

[36] Adams Wei Yu, Lei Huang, Qihang Lin, Ruslan Salakhutdinov, and Jaime Carbonell. 2017. Block-normalized gradient method: An empirical study for training deep neural network. arXiv:1707.04822. Retrieved from https://arxiv.org/abs/1707.04822.

[37] Taha Zerrouki and Amar Balla. 2017. Tashkeela: Novel corpus of Arabic vocalized texts, data for auto-diacritization systems. *Data in Brief* 11 (2017), 147.

[38] Imed Zitouni and Ruhi Sarikaya. 2009. Arabic diacritic restoration approach based on maximum entropy models. *Computer Speech and Language* 23, 3 (2009), 257–276.

Received August 2020; revised February 2021; accepted June 2021