IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received August 5, 2020, accepted August 19, 2020, date of publication August 24, 2020, date of current version September 3, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3018885

Arabic Diacritization Using Bidirectional Long Short-Term Memory Neural Networks With Conditional Random Fields

ABDULMOHSEN AL-THUBAITY, ATHEER ALKHALIFA, ABDULRAHMAN ALMUHAREB, AND WALEED ALSANIE

National Center for Data Analytics and Artificial Intelligence, KACST, Riyadh 11442, Saudi Arabia

Corresponding author: Abdulmohsen Al-Thubaity (aalthubaity@kacst.edu.sa)

This work was supported by the King Abdulaziz City for Science and Technology (KACST).

ABSTRACT Arabic diacritics play a significant role in distinguishing words with the same orthography but different meanings, pronunciations, and syntactic functions. The presence of Arabic diacritics can be useful in many natural language processing applications, such as text-to-speech tasks, machine translation, and part-of-speech tagging. This article discusses the use of bidirectional long short-term memory neural networks with conditional random fields for Arabic diacritization. This approach requires no morphological analyzers, dictionary, or feature engineering, but rather uses a sequence-to-sequence schema. The input is a sequence of characters that constitute the sentence, and the output consists of the corresponding diacritic(s) for each character in that sentence. The performance of the proposed approach was examined using four datasets with different sizes and genres, namely, the King Abdulaziz City for Science and Technology text-to-speech (KACST TTS) dataset, the Holy Quran, Sahih Al-Bukhary, and the Penn Arabic Treebank (ATB). For training, 60% of the sentences were randomly selected from each dataset, 20% were selected for validation, and 20% were selected for testing. The trained models achieved diacritic error rates of 3.41%, 1.34%, 1.57%, and 2.13% and word error rates of 14.46%, 4.92%, 5.65%, and 8.43% on the KACST TTS, Holy Quran, Sahih Al-Bukhary, and ATB datasets, respectively. Comparison of the proposed method with those used in other studies and existing systems revealed that its results are comparable to or better than those of the state-of-the-art methods.

INDEX TERMS Arabic diacritic restoration, bi-directional long short-term memory, computational linguistics, conditional random fields, deep learning, neural network.

I. INTRODUCTION

Diacritics are marks written above or below words or letters in several languages such as Arabic [1], Turkish [2], and Romanian [3]. Arabic texts are usually written without diacritics, and readers can infer the meanings and correct pronunciations of the words from their contexts. However, such inference is not always easy. In Arabic, diacritics are used to clarify the correct pronunciations, meanings, and syntactic roles of words, specifically in the last letters of words (see Section III for more information). The word علم ''Elm,'' for instance, has several meanings: عَلَمْ ''Ealamo'' (flag: noun), عِلْمْ ''Eil∼omo'' (science: noun), عَلِمَ ''Ealima'' (he knows: verb), عُلِّمَ ''Eul∼ima'' (he has been taught: verb), عَلَّمَ ''EalFma'' (he taught: verb), and عُلِمَ ''Eulima'' (it was known: verb). Without diacritics, it is difficult to infer the meaning and pronounce the word correctly.

Furthermore, the diacritic of the last letter in the word reveals its syntactic role. Consider the noun عَلَمْ ''Ealamo'' (flag) in the following two sentences:

رفرف عَلَمُ البلاد (The country flag flapped.)
رفع الجندي عَلَمَ البلاد (The soldier raised the country's flag.)

In the first sentence, the word is a subject, so the diacritic of the last letter م ''m'' is Damma ''ُ'' because it is a singular noun. In the second sentence, the word is an object; hence, the diacritic of the last letter م ''m'' is Fatha ''َ'' because it is a singular noun. Note that the diacritics of the first and second

The associate editor coordinating the review of this manuscript and approving it for publication was Aysegul Ucar.

letters of the word do not change; otherwise, the meaning would also change.

The objective of any diacritization system is to restore the missing diacritics, that is, to assign the correct diacritics to the letters in the words of a given sentence either fully or partially as required. Diacritic restoration is not exclusive to classical Arabic [1], modern standard Arabic [4], or Arabic dialects [5], [6]. Several efforts are also being made in other languages such as Turkish [2], Romanian [3], and Yorùbá [7].

Restoring Arabic diacritics can be useful in many natural language processing and computational linguistics tasks. These tasks include text-to-speech applications [8], automatic speech recognition [9], homograph disambiguation [10], part-of-speech (POS) tagging [11], identifying the syntactic role of the words in sentences [12], and Arabic machine translation [13].

The existing systems for Arabic diacritic restoration typically consider the problem either within morphological disambiguation or as a standalone problem. In the latter case, most proposed systems are based on dictionaries and rules, language resources, or feature engineering approaches that employ linguistic information. In contrast, few efforts have been made based on data alone (see Section II).

In this study, we investigated the performance of a supervised deep learning approach, specifically, bidirectional long short-term memory (BiLSTM) with conditional random fields (CRFs) [14] for Arabic diacritization. The BiLSTM-CRF approach has been shown to be suitable for many natural language processing and text mining tasks that depend on sequence tagging, such as name entity recognition [15], POS tagging [16], and sentiment detection [17].

The contribution of this study is twofold. First, the proposed method of Arabic diacritic restoration does not employ any type of morphological analyzer, a dictionary, rules, or any kind of feature engineering. It is solely based on data and is distinct from other Arabic diacritic restoration efforts that employ long short-term memory (LSTM) networks by using the sequence of characters that constitute the sentence as input and their corresponding diacritics as output. The proposed approach does not use moving windows or a hot vector as input. Second, we investigated the performance of the proposed method using four datasets of different sizes covering different genres, namely, the King Abdulaziz City for Science and Technology text-to-speech (KACST TTS) dataset, the Holy Quran, Sahih Al-Bukhary, and the Penn Arabic Treebank (ATB). To the best of our knowledge, no previous researchers have used such diverse datasets to evaluate their models.

The remainder of this article is organized as follows. Section II presents the related literature. Section III provides background information about Arabic diacritics, BiLSTM and CRF. Sections IV and V describe the datasets used in the experiments and the proposed method. Section VI presents the experimental results. Section VII compares the proposed system with other Arabic diacritization systems. Finally, Section VIII summarizes the conclusions.

II. RELATED WORK

This section discusses some of the recent studies examining full Arabic diacritization. Partial Arabic diacritization, for example, on case ending [12] and semantic disambiguation [10], is not addressed because it is outside the scope of this study.

The existing Arabic diacritization algorithms have used three approaches to represent data. The first approach mainly relies on morphological analysis. Habash *et al.*, for example, used the Backwalter and Standard Arabic morphological analyzers to list all possible analyses for each word in the given sentences, including diacritization. To predict the correct morphological analysis from the generated analysis lists, they used different algorithms such as support vector machine (SVM) methods [18]–[21], the J48 decision tree classifier with manually crafted rules [22], and LSTM [23]. Hussein *et al.* [24] also used the Backwalter Arabic morphological analyzer with hidden Markov models (HMMs) for Arabic diacritization.

Other researchers have employed different morphological analyzers to solve the problem of Arabic diacritic restoration. Chennoufi and Mazroui [25] used the Alkhalil Arabic morphological analyzer with a set of rules and an HMM. In addition, Said *et al.* [26] developed a statistical- and rule-based morphological analyzer and rule-based algorithm.

The second approach to represent the data in Arabic diacritization relies on feature engineering. The objective of this approach is to supply the learning algorithm with the most representative features of the data to help the algorithm learn more effectively.

The most commonly used features in Arabic diacritization are linguistic features. For instance, Zitouni *et al.* [27] utilized lexical, segment-based, and POS features with a statistical approach based on maximum entropy. To train deep neural networks, Rashwan *et al.* [28] employed linguistic features, namely, POS tags, prefixes, suffixes, roots, and patterns, alongside other features related to word context, such as previous character, last character, previous word, and last word.

Based on language models that were built on the surface form of the words, morphologically segmented words, and character level, Mubarak *et al.* [6] used a Viterbi search to choose the most probable diacritization for an input word in a sentence.

Similarly, Darwish *et al.* [29] utilized a Viterbi decoder at the word level with word stems, morphological patterns, and transliteration for the core word diacritization and SVM-based ranking with morphological patterns and linguistic rules to determine the proper case ending of the word.

The third approach relies solely on data, specifically, on characters and words that compose the sentence. This approach is mainly used with deep learning algorithms, especially recurrent neural networks (RNNs) and their variants, LSTM, and BiLSTM because of their abilities to handle sequential data such as text. Our work falls into this category.

Abandah *et al.* [30] employed a deep neural network built by stacking a set of BiLSTM layers to work around the problem of the dependency of the diacritization of the current word on both the previous and following words. This architecture allows the decision to be more context-aware. Subsequent error correction techniques were used to improve the results. The network was trained in either a one-to-one manner (in which the input length equals the output length) or a many-to-many manner (in which the output is longer than the input); empirically, the one-to-one architecture outperformed the one-to-many architecture in several tests.

Belinkov and Glass [31] focused on using different types of neural networks to build a language-independent method of restoring missing diacritics. They undertook numerous experiments for various hidden layer types of neural networks, from a single feed-forward neural network (FFNN) layer to BiLSTM layers. Their results suggest that the use of stacked layers of BiLSTM (three layers) yields better results than FFNNs and LSTM.

Mubarak *et al.* [32] readjusted a standard sequence-to-sequence neural machine translation setup based on an LSTM architecture to formulate a unified model for Arabic diacritic restoration. Their model operates by training on a fixed-length sliding window of *n* words, which are expressed in terms of their individual characters, followed by a voting mechanism to select the most suitable diacritized form for a given word.

The performances of two convolutional neural networks, namely, the temporal convolutional neural network (TCN) and acausal temporal convolutional neural network (A-TCN), were compared with two RNNs, namely, LSTM and BiLSTM, for Arabic diacritization [33]. The results showed that BiLSTM outperforms LSTM and A-TCN outperforms TCN, and the best model among the four is BiLSTM.

Using the same data representation approach, Khorsheed [34] investigated the use of an HMM in conjunction with a character-based 4-gram model to facilitate the selection of the most suitable diacritization for a word. Every individual diacritic has a separate HMM model that concatenates the output models alongside the input character sequence to determine the final diacritized sentence.

ATB part 3 using the split of Zitouni *et al.* [27] for training and testing (cf. [6], [26], [27], [30], [31]) and the three parts of ATB (parts 1, 2, and 3) using the split of Diab *et al.* [35] for training, validation, and testing (cf. [21]–[23], [28], [33]) are the most commonly used benchmarking datasets for Arabic diacritization. Because the ATB dataset is not free, other datasets are utilized for Arabic diacritization with no standard split for training and testing, such as the Holy Quran [30], [34] and various collections of old manuscripts from religious books [1], [25], [30].

The state-of-the-art results for ATB part 3 are a diacritic error rate (DER) and word error rate (WER) of 1.6% [6] and 9.07% [30], respectively. For the three parts of ATB, the state-of-the-art results are a DER and WER of 2.8% and 8.2%, respectively [33]. The method of Abandah *et al.* [30] yielded the best results for the Holy Quran, achieving a DER and WER of 3.04% and 8.7%, respectively. Mubarak *et al.* [32] reported a WER of 4.49% using a 9.7 million word manually diacritized and revised corpus provided by a commercial vendor.

TABLE 1. Fifteen Arabic diacritics along with their Buckwalter transliterations when used with the letter م "m".

	Diacritic	Shape	Buckwalter transliteration
1	Fatha	مَ	ma
2	Damma	مُ	mu
3	Kasra	مِ	mi
4	Sukun	مْ	mo
5	Tanwin Fatha	مً	mF
6	Tanwin Damma	مٌ	mN
7	Tanwin Kasra	مٍ	mK
8	Shadda	مّ	m~
9	Shadda + Fatha	مَّ	m~a
10	Shadda + Damma	مُّ	m~u
11	Shadda + Kasra	مِّ	m~i
12	Shadda + Sukun	مْ	m~o
13	Shadda + Tanwin Fatha	مًّ	m~F
14	Shadda + Tanwin Damma	مٌّ	m~N
15	Shadda + Tanwin Kasra	مٍّ	m~K

Using Arabic diacritization systems that rely on the first and second approaches, that is, morphological analyzers and feature engineering, entail additional costs in real life. This situation necessitates the availability of morphological analyzers and/or feature extraction tools that are used to build these systems, which in most cases are available with fees or are not available at all to the public research community. Using the third approach, which relies solely on data, requires the availability of diacritized texts that are freely available along with many powerful machine learning packages that are also freely available.

III. BACKGROUND

A. ARABIC DIACRITICS

Arabic diacritics can be classified into four groups. The first consists of the short vowels, namely, Fatha "◌َ," Damma "◌ُ," and Kasra "◌ِ." The second group consists of Sukun "◌ْ," which indicates that the consonant is not followed by a vowel. The third group consists of the Tanwin diacritics: Tanwin Fatha "◌ً," Tanwin Damma "◌ٌ," and Tanwin Kasra "◌ٍ." Tanwin diacritics are used only with the last letter of a word if needed. They indicate that the vowel is followed by the consonant ن "n," which is pronounced but not written in that particular case. Finally, the fourth group consists of Shadda "◌ّ," which indicates germination, i.e., the doubling of a consonant letter. Shadda can be utilized with all short vowels, Sukun, or Tanwin diacritics. These four groups include 15 diacritics. Table 1 summarizes these diacritics along with their Buckwalter transliterations when used with the letter م "m."

In fully diacritized texts, some diacritics such as the following are often neglected:

1) The diacritics for "lam shamsiah," the second letter in the definitive article ال when it is written but not pronounced, are ignored. A well-known example is the word الشمس "Al$ms" (the sun), in which this letter is not pronounced. Hence, no diacritic is assigned.
2) Similar to case 1, the diacritics for "lam qamariah," which is the second letter in the definitive article ال and is written and pronounced, are not written, such as in the word القمر "Alqmr" (the moon). However, its correct diacritic "ْ" is occasionally assigned.
3) The diacritics for Alif maqsourah ى "Y" and Alif madd آ "=" are ignored.
4) The diacritics for the letter ا "A" are not added when it comes in the middle of the word.
5) Diacritics that are a part of the long vowel و "w" are not added.
6) Diacritics that are a part of the long vowel ي "y" are not added.
7) Very occasionally, the diacritic for the letter إ "<" is neglected because it must be Kasra.

The omission of diacritics is not consistent in all diacritized texts. This inconsistency does not affect the ability of readers to understand such texts because all neglected diacritics can easily be inferred. However, the case is different for machines, where inconsistent diacritization often causes models that have been trained on such data to perform poorly.

B. BiLSTM NETWORKS

The function of any classification algorithm is to transfer the input data X_t to a certain output Y_t. One of the limitations of FFNNs is that the transformation process from X_t to Y_t does not have access to the previous output Y_{t-1}. Recurrent neural networks (RNN), where the output of each cell y_t depends not only on its input x_t but also on the output of the previous cell y_{t-1}, partially overcome this limitation. In the literature, the output y_t is usually denoted by h_t.

The ability to remember the previous output only (short-term memory) has proven to be efficient for several tasks, such as predicting the last word in a sentence. However, in some cases, it is necessary to remember the previous outputs (long-term memory) to produce h_t. LSTM networks solve this problem because they are able to learn the long-term dependencies between the inputs and outputs.

An LSTM cell receives three inputs, namely, the output of the previous cell h_{t-1}, cell state of the previous cell C_{t-1}, and input into the cell from the previous layer x_t. It then produces two outputs: the cell output h_t and the cell state C_t [37], [38]. Figure 1 illustrates the architecture of an LSTM cell that can control the following information:

a. Which information the cell passes to the next cell: This information is controlled using a "forget gate." The output of the forget gate f_t is given by

$$f_t = \alpha\left(W_f . [h_{t-1}, x_t] + b_f\right). \tag{1}$$

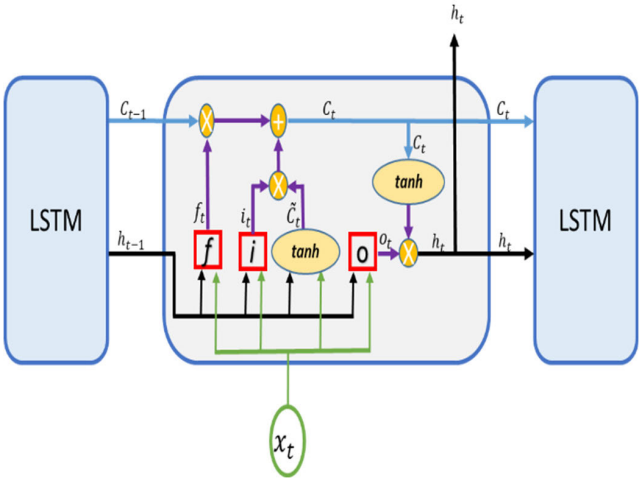

FIGURE 1. LSTM architecture.

b. The information that is stored in the cell state: This information is based on the output of the input gate i_t and the previous state of the cell $\tilde{C}_t$, which can be expressed as follows:

$$i_t = \alpha\left(W_i . [h_{t-1}, x_t] + b_i\right) \tag{2}$$

and

$$\tilde{C}_t = tanh\left(W_C . [h_{t-1}, x_t] + b_C\right). \tag{3}$$

The cell state C_t, which is passed to the next cell, is given by

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t. \tag{4}$$

c. The information produced as an output of the cell: The cell output h_t is based on the output of the output gate o_t and the current cell state C_t, where

$$o_t = \alpha\left(W_o . [h_{t-1}, x_t] + b_o\right) \tag{5}$$

and

$$h_t = o_t * \tanh(C_t). \tag{6}$$

Like the current cell state C_t, the cell output h_t is passed to the next cell. In addition, W_f, W_i, W_C, and W_o are weights; b_f, b_i, b_C, and b_o are biases; and α is the sigmoid function.

In a BiLSTM network, each BiLSTM cell consists of two LSTM cells. One LSTM cell processes the input data from right to left and the other processes the input data from left to right. The outputs of both LSTM cells $\vec{h_t}$ and $\overleftarrow{h_t}$ are concatenated to produce the output of the BiLSTM cell. This architecture allows the network to benefit from left and right inputs to produce the output of the current instance. Figure 2 shows the architecture of the BiLSTM network.

C. CRF

CRFs constitute a class of probabilistic undirected graphical models. A CRF represents a conditional density of a set of class labels Y_t given a set of observations X_t, and the density is factorized according to the graph [39]. One of the

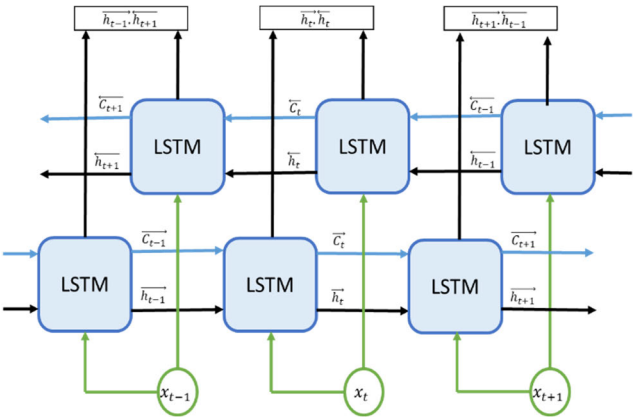

FIGURE 2. BiLSTM network architecture.

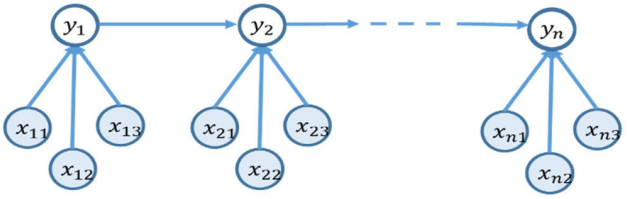

FIGURE 3. Example of an L-CRF.

most widely adopted instances of CRFs is the linear chain conditional random field (L-CRF), which is considered in this article. The density of an L-CRF, shown in Fig. 3, is defined over a sequence of class labels $Y_t = \{y_t\}_{t=0}^{n}$ (empty ovals) given a sequence of observations $X_t = \{x_t\}_{t=0}^{n}$ (filled ovals) as follows:

$$p(y|X = Z\,(X)^{-1} \prod_{t=1}^{n} \Psi_t\,(y_t, y_{t-1}, x_t), \qquad (7)$$

where $Z\,(.)$ is a normalizing factor. As shown in (7), the density models the dependency of the class labels not only on the observations, but also on the other class labels [40]. This property enables relational data to be processed. Therefore, assigning a class label y_t depends on the features x_t and context $y_{j\neq t}$.

L-CRFs are widely adopted in natural language processing, where tokens, whether they are letters, words, morphemes, phonemes, or diacritics, appear as sequences, and each is mostly dependent on its context. It has been used successfully in many language processing tasks such as POS tagging [41] and named entity recognition [42].

IV. DATA

To investigate the performance of the BiLSTM-CRF for automatic Arabic diacritization, we used four datasets covering various genres of different sizes, which were collected from KACST TTS, the Holy Quran, Sahih Al-Bukhary, and the ATB.

KACST TTS [43] is the smallest corpus among our datasets. It consists of randomly selected sentences from different genres such as news, finance, customer service, and literature and was manually diacritized and revised.

The Holy Quran is the most analyzed text in Arabic because it is, as Muslims believe, the revelation from Allah to the Prophet Mohammad. Sahih Al-Bukhary is one of two major collections of Hadith, or traditions relating to the Prophet and his companions in Islam. All official releases of the Holy Quran and Sahih Al-Bukhary are either printed or in an electronic format, fully diacritized, and thoroughly revised to facilitate reading and to avoid ambiguity in meaning. Thus, they are ideal datasets for Arabic diacritization. We downloaded the Holy Quran and Sahih Al-Bukhary from the Tanzil[1] and Shamela[2] websites, respectively.

TABLE 2. Basic dataset statistics.

Dataset	No. of sentences	No. of diacritized words	No. of characters
KACST TTS	4,400	51,687	295,396
Holy Quran	6,227	84,915	397,547
Sahih Al-Bukhary	11,496	617,932	2,990,215
ATB	24,274	622,739	3,349,477

ATB [44] is the largest corpus among our datasets. Three parts of the ATB were used in this study: part 1 (v 3.0), part 2 (v 2.0), and part 3 (v 2.0). The ATB consists of modern standard Arabic texts from the news domain and has been used in numerous studies on Arabic computational linguistics. Several cases of inconsistent diacritization have been identified in the ATB, and numerous reports have noted such observations [6], [29].

To unify the structures of our datasets because they came from different sources, we performed a few preprocessing steps:

a. We divided each dataset into a set of sentences, where ".," "?," and "!" were used as delimiters. However, for the Holy Quran, each verse was treated as a sentence.
b. Extra spaces were removed from each sentence.
c. The positions of the diacritics were unified, where each diacritic was inserted directly after the corresponding letter.
d. When Shaddah and other diacritics were used, the sequence was unified, where the Shaddah came first, followed by the other diacritics.

Table 2 summarizes the basic statistics for the datasets used in this study. We counted the number of sentences based on our sentence segmenter. The following sentences are examples from KACST TTS, Holy Quran, Sahih Al-Bukhary, and ATB:

تَمَّ تَعْدِيلُ خَطِّ السَّيْرِ بِنَجَاحْ .. مِنْ فَضْلِكَ، تَوَجَّهْ إِلَى أَقْرَبِ مَكْتَبٍ لِاسْتِبْدَالِ التَّذْكَرَةْ.

ذَلِكَ الْكِتَابُ لَا رَيْبَ فِيهِ هُدًى لِلْمُتَّقِينَ

قَالَ حَدَّثَنَا أَبِي قَالَ حَدَّثَنَا أَبُو بُرْدَةَ بْنُ عَبْدِ اللَّهِ بْنِ أَبِي بُرْدَةَ عَنْ أَبِي بُرْدَةَ عَنْ أَبِي مُوسَى رَضِيَ اللَّهُ عَنْهُ قَالَ قَالُوا يَا رَسُولَ اللَّهِ أَيُّ الْإِسْلَامِ أَفْضَلُ قَالَ مَنْ سَلِمَ الْمُسْلِمُونَ مِنْ لِسَانِهِ وَيَدِهِ

وَأَفادَت مَصادِرُ مُطَّلِعَةٌ عَلَى التَحْقِيقِ أَنَّهُ يَجْرِي التَوَسُّعُ فِيهِ مَعَ ثَلاثَةِ عَناصِرَ مِن قُوَى الأَمْنِ الداخِلِيِّ أُحِيلُوا عَلَى مَفْرَزَةِ التَحَرِّي لِاسْتِيضاحِهِم طَبِيعَةَ العَلاقَةِ الَّتِي تَرْبِطُهُم بِالمُخْتَفِي

[1]http://tanzil.net/download
[2]http://www.shamela.ws

TABLE 3. Percentages of Arabic letters with and without diacritics in each corpus in our dataset.

Dataset	With diacritics (%)	Without diacritics (%)
KACST TTS	77.25	22.75
Holy Quran	77.99	22.01
Sahih Al-Bukhary	79.67	20.33
ATB	61.18	38.82

To prepare our data for training, we assigned a tag to each character in each sentence. The tag was either one diacritic (a short vowel or Sukun, Shadda, or Tanwin diacritic), two diacritics (Shadda with short vowels, Sukun, or Tanwin diacritic), or one of two special tags "_" and "!." The special tag "_" was used to indicate that the diacritic was missing, while the special tag "!" was utilized to indicate that the character was not an Arabic letter and hence that no diacritic could be assigned to it. We split each dataset randomly into three parts: training (60%), validation (20%), and testing (20%).

Table 3 lists the percentages of Arabic letters with and without diacritics for each corpus in our dataset. The data show that about 74% of all the letters in the overall dataset have diacritics. About 80% of the letters are diacritized in the Al-Bukhary text. In contrast, only 61% of the letters are diacritized in the ATB. After reviewing the datasets, we found that about 40% of the non-diacritized letters are associated with the definite article ال "Al." If the second letter ل "l" is lam shamsiah, no diacritic is assigned because this ل is not pronounced. If the second letter ل is lam qamariah, the diacritics must be Sukun; however, this diacritic is neglected, specifically in the ATB corpus. One of the clear cases of neglected diacritics in the ATB is the diacritic of a letter in the middle of a word, which must be Fatha and followed by "ا." In this case, the diacritics for both letters were neglected. The neglect of diacritics negatively affects the accuracy of the diacritization model, especially when the diacritics are not neglected consistently.

Table 4 summarizes the distributions of all the diacritics in the datasets. The most frequent diacritic in all datasets is Fatha, followed by Kasra, which together represent approximately 63% of the total. Sukun, Damma, and Shadda with Fatha come next, representing approximately 32% of all diacritics. The remaining 10 diacritics all occur infrequently in the dataset, with less than 2% frequency for each tag. The unbalanced distribution of diacritics will probably affect the diacritization accuracy; that is, the system will be able to restore frequent diacritics more accurately than infrequent diacritics.

The diacritic distributions reveal that the KACST TTS corpus is the only corpus that contains the diacritization Shadda with Sukun "ّْ." All of these cases (47 cases) occur at the ends of sentences and on the letter ج. The reason for the occurrence of "ّْ" in the KACST TTS corpus is that the corpus was diacritized specifically for the TTS application. In Arabic, there is a rule stating that "Arabic speech does not start with a consonant and does not pause with a vowel." This rule may explain why the Sukun diacritic percentage (20%)

TABLE 4. Diacritic distribution in each corpus.

Diacritic	KACST TTS (%)	Holy Quran (%)	Sahih Al-Bukhary (%)	ATB (%)	ALL (%)
ً	39.13	41.49	45.99	35.66	41.11
ٌ	9.17	13.94	11.79	11.50	11.69
ٍ	22.07	16.77	14.58	30.73	21.74
َّ	20.63	17.14	15.62	11.43	14.22
ُّ	0.87	1.46	0.64	1.19	0.93
ِّ	0.35	0.91	0.70	0.41	0.58
ّ	0.80	0.97	1.39	1.58	1.42
ْ	0.30	0.11	0.03	0.71	0.33
ًّ	4.65	5.52	7.73	4.40	6.07
ٌّ	0.77	0.57	0.64	0.53	0.60
ٍّ	1.12	1.03	0.80	1.64	1.18
ًْ	0.03	-	-	-	0.001
ٌْ	0.04	-	0.002	-	0.003
ّْ	0.03	0.05	0.03	0.06	0.05
ٍْ	0.03	0.03	0.05	0.15	0.09

*Dashes indicate no occurrences of the corresponding diacritics.

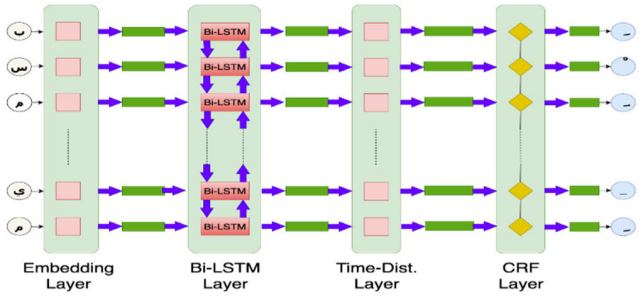

FIGURE 4. Proposed model architecture.

is higher in the KACST TTS corpus than in the other corpora. Additionally, Table 4 shows that the KACST TTS and Sahih Al-Bukhary corpora are the only corpora that contain the diacritic Shadda with Tanwin Fatha.

In Arabic, the diacritic of the last letter of a word usually has a special property: it is not always fixed. This property originates from the fact that, in most cases, the same word can have different diacritics for the last letter based on its syntactic role in the sentence. Exceptions occur in a set of words known as "Mabniah," whose phonological structures are fixed regardless of their syntactic roles. In the former case, if the word is a singular noun, for instance, and it is the subject in a verbal sentence, the last diacritic should be Damma. If the same word is the object, the last diacritic should be Fatha. In the data, the most frequent diacritics for the last letter are Kasra (20%), Fatha (18%), and Damma (13%).

V. METHOD

Our model is based on the BiLSTM-CRF. It consists of six consecutive layers: an input layer, a character embedding layer, a BiLSTM layer, a time-distributed layer, a CRF layer, and an output layer. Figure 4 illustrates the architecture of the proposed model.

1) **Input layer:** The model accepts its input (i.e., a sentence) as a sequence of characters through the input

layer. Because the length of the input varies from sentence to sentence, the length of the input given to the model must be fixed to a predefined value (max_len). When the length of the input sentence is greater than the value of max_len, any character beyond this limit is truncated. In the opposite scenario, that is, when the length of the input sentence is less than the value of max_len, the input sequence is padded with ENDPAD for each input character exceeding the actual length of the sentence. Each character in the training data has a unique numerical representation. The output of this layer is fed to the next layer: the character embedding layer.

2) **Character embedding layer:** Arabic is a morphologically rich language, and word prefixes such as the letters ب "b" and ل "l," for example, affect the diacritic of the last letter in the word. Several studies have shown that character embedding is useful for many natural language processing tasks, and it can deal well with the out-of-vocabulary problem and morphologically rich languages [45]–[47]. The character embedding layer accepts the input via the input layer and produces a vector representation with a predefined length for each character in the training data. We used a vector of length 128 to represent each character in the training dataset.
3) **BiLSTM layer:** The input into this layer is a set of embedded characters. We set the number of units to 128, where the layer is set to return all sequences; that is, the BiLSTM representation of each character in the input sentence (h_0 to h_n). This representation has been proven effective for making independent tagging decisions [44]. The encoded sequence is passed to a CRF through the next layer (the time-distributed layer). The recurrent dropout is set to 0.4 to avoid over-fitting. The hyperbolic tangent (tanh) serves as an activation function for this layer. Instead of a single BiLSTM layer, several BiLSTM layers can be stacked. However, this configuration complicates the model and slows the training process.
4) **Time distributed layer:** The time distributed layer enables the application of one layer to every element of the BiLSTM sequence output independently and control of the dimension of the data input into the CRF. The output dimension of this layer was set to 128. As for the BiLSTM layer, tanh serves as an activation function for this layer.
5) **CRF layer:** The CRF algorithm is well known for its performance in sequence labeling. We chose to use the L-CRF architecture in this layer to complement the BiLSTM for the classification layer. The CRF layer increases the overall complexity of the model, specifically, its time complexity, which is largely affected by the input sequence length and output sequence length. The time complexity for the linear chain CRF is $O(l.|y|^2)$, where $|y|$ is the length of the output sequence and l is the length of the input sequence.

TABLE 5. Diacritization performance results for each dataset.

Dataset	Maximum input length	DER (%)	WER (%)	SER (%)
KACST TTS	250	3.41	14.46	5.46
Holy Quran	250	1.34	4.92	2.19
Sahih Al-Bukhary	650	1.57	5.65	2.97
ATB	550	2.13	8.43	5.53
Q+BUKH+ATB	550	2.63	9.78	5.61

6) **Output layer:** The target of our model is a sequence of diacritics corresponding to the input sequence, where each diacritic in the training data is represented as a one-hot vector. The length of this vector is equal to the number of diacritics in the training dataset plus two special tags used to indicate that the character is not an Arabic letter (and hence cannot be diacritized) or that the diacritic for the corresponding Arabic letter is missing from the dataset.

We implemented the proposed model with Python 3.7 using the Keras library (with a TensorFlow backend) version 2.2.4 and Keras_contrib library version 0.0.2. To train our model, we used Adam as an optimization function with the default parameters provided by the Keras library (learning rate = 0.001, β_1 = 0.9, β_2 = 0.999, ε = 1e-07, amsgrad = False) and the crf.loss_function from Keras_contrib as a loss function. The batch size was 32, and the number of epochs was set to 200 with early stopping if the loss function did not improve within 40 epochs. For the other training parameters, we used the default settings provided by the aforementioned libraries. For example, the kernel initializer was "glorot_uniform," and the bias initializer was "zeros." We ran our experiments on a PC with an Intel Core i7-8750H CPU (2.2 GHz) and 16 GB RAM. We did not use a GPU.

To measure the performances of the resulting models, we used three measures, namely, the DER, WER, and SER. The DER is the percentage of all the characters that are incorrectly discretized. The WER is the percentage of all the words that are incorrectly discretized. An entire word was considered incorrectly diacritized if one character of the word was incorrectly diacritized. The SER focuses on the last letter of the word and is the percentage of all the words for which the last letter is incorrectly diacritized.

VI. RESULTS

We conducted 20 experiments on each dataset to test the performance of the proposed method. Firstly, we examined the effect of input length, where we increased the length of the input sequence in intervals of 50, starting from 250 characters and ending with 700 characters. For each input, we set the dropout value to 0.3 and 0.4. The results suggest that using 0.4 as the dropout yields marginally better results. Table 5 illustrates the DER, WER, and SER for all datasets and the combined Holy Quran, Sahih Al-Bukhary, and ATB dataset (Q + BUKH + ATB) along with the input length that yielded the best results.

TABLE 6. Percentages of sentences with one or more word errors among all erroneous sentences for each test dataset.

Number of errors in sentence	KACST TTS (%)	Holy Quran (%)	Sahih Al-Bukhary (%)	ATB (%)	Q+BUKH+ATB (%)
1	38.5	45.2	32.0	30.4	25.5
2	17.8	24.8	20.2	22.6	20.5
3	11.5	13.9	12.2	16.8	16.3
4	7.9	5.0	9.0	11.3	10.5
>4	24.3	11.1	26.6	18.9	27.3

The results show that the best performance is obtained when the second smallest dataset, the Holy Quran dataset, is used, followed by the third smallest dataset, the Sahih Al-Bukhary dataset. These results suggest that the proposed method is considerably affected by the data quality. The Holy Quran is the most revered, revised, and studied text in Arabic, so the diacritization quality of the Holy Quran is the best, followed by that of Sahih Al-Bukhary.

Although the ATB dataset is larger than the Holy Quran and Sahih Al-Bukhary datasets, the diacritization performance is worse for this dataset than for the Holy Quran and Sahih Al-Bukhary datasets. This phenomenon may be due to the quality of ATB diacritization (see Section IV).

The method produced worse DER, WER, and SER results when the combined dataset (Q + BUKH + ATB) was employed than when the individual datasets were used. This low performance may be due to the inconsistency between the diacritization approaches in the three datasets and the differences between the genres of the datasets. The proposed method performed the worst on the KACST TTS dataset because of the small size of this dataset and the diversity of its topics.

Noting the use of different splits for training and testing, the DER and WER results for the Holy Quran are superior to those achieved by Abandah *et al.* [30] (3.04% for DER and 8.7% for WER) and the DER results for KACST TTS are better than those of Khorsheed [34] (26.87%).

We analyzed the errors produced by the models at the sentence level, where we counted the numbers of words that the models failed to predict correctly. We found that 50.9%, 74.3%, 29.7%, 28.0%, and 21.3% of the sentences from the testing data were diacritized with no error by the KACST TTS, Holy Quran, Sahih Al-Bukhary, ATB, and Q + BUKH + ATB models, respectively. Although the KACST TTS model performed worse than the other models in terms of DER and WER, it outperformed the other models in terms of the percentage of sentences that had no errors in diacritization prediction. The fact that the KACST TTS model performed the worst in terms of DER and WER may be due to the small sizes of the testing and training datasets. In particular, DER and WER are largely affected by the size of the testing data.

Table 6 lists the percentages of sentences with one or more word errors among all erroneous sentences for eachtest dataset. The data show that most of the erroneous sentences have only one word error, followed by sentences with two, three, four, and more than four word errors.

TABLE 7. Percentages of errors for each diacritic and each testing dataset.

Diacritic	KACST TTS (%)	Holy Quran (%)	Sahih Al-Bukhary (%)	ATB (%)	Q+BUKH+ATB (%)
◌َ	4.41	1.67	1.83	2.85	3.15
◌ُ	9.01	3.85	4.45	6.70	7.57
◌ِ	3.68	1.63	2.86	1.99	2.98
◌ّ	3.82	1.06	1.67	2.24	2.62
◌ْ	7.14	2.60	5.18	5.74	6.18
◌ٌ	25.53	6.14	12.50	27.04	28.62
◌ٍ	0.00	4.65	2.77	9.26	6.50
◌ً	9.09	1.72	7.02	7.00	12.02
◌ّ◌َ	5.13	1.63	1.80	3.75	2.95
◌ّ◌ُ	9.02	6.27	6.50	14.57	15.21
◌ّ◌ِ	11.57	5.34	6.82	5.28	8.66
◌ّ◌ٌ	90.91	-	-	-	-
◌ّ◌ٍ	0.00	-	0.00	-	5.56
◌ّ◌ً	33.33	4.76	23.65	29.44	35.83
◌ّ◌ْ	40.00	5.56	10.88	8.09	14.20
_	1.99	0.29	0.28	1.78	1.93
!	0.00	0.00	0.00	0.00	0.00

TABLE 8. Numbers of words with one, two, three, and more than three diacritization errors in the error samples.

Error sample	Number of diacritic errors				Total number of diacritics errors
	1	2	3	>3	
KACST TTS	81	14	5	0	124
Holy Quran	83	14	2	1	121
Sahih Al-Bukhary	85	10	5	0	120
ATB	85	12	2	1	122
Q+BUKH+ATB	86	13	0	1	116

Table 7 presents the percentages of errors among the diacritics for each test dataset. In general, as expected for any classification problem, the data suggest that the diacritics that appear frequently in the corpus have lower error percentages compared with diacritics that appear less frequently (see Table 4). The data show that the models were able to assign no diacritics for non-Arabic letters. Note that we did not use any post-processing rules in this case. Moreover, the lowest error percentage for all testing datasets occurs for the special tag "_," which denotes that the diacritic is missing.

We randomly selected 100 words with diacritization errors from each testing dataset to find the common error types. Table 8 shows the numbers of words with one, two, three, and more than three diacritization errors. The data suggest that all of the trained models behave similarly regarding the number of diacritic errors per word, where most words (84 on average) have one diacritic error, followed by the words that have two diacritic errors (12.6 on average). Words with three or more errors are very rare (3.4 on average), and there were four diacritic errors for one word each in the Holy Quran and Q + BUKH + ATB samples and seven diacritic errors for one word in the ATB sample. These observations indicate the consistency of the model performance regardless of the type of data.

TABLE 9. Positions of diacritization errors in error samples.

Error sample	Beginning	Middle	End
KACST TTS	15	49	51
Holy Quran	26	52	35
Sahih Al-Bukhary	18	45	47
ATB	15	30	62
Q+BUKH+ATB	14	39	56

TABLE 10. Distributions of diacritization errors across the main arabic pos categories in the error samples.

Error sample	Nouns	Verbs	Particles
KACST TTS	89	11	0
Holy Quran	51	45	4
Sahih Al-Bukhary	64	31	5
ATB	82	16	2
Q+BUKH+ATB	87	13	0

In the error samples, diacritization errors occurred at the beginning, middle, or end of the words; concurrently at all of these places; or at two of them. As Table 9 illustrates, the trained models are sensitive to the syntactic roles of the words because most of the errors occurred at the ends of the words (50.2 on average), except in the case of the Holy Quran, where most of the errors occurred in the middle of the words. The high accuracy for the Holy Quran is probably due to the diacritization quality and short length of the Holy Quran sentences compared to the length of the sentences in the other datasets.

Table 10 shows the distributions of diacritization errors across the three main Arabic POS categories: nouns, verbs, and particles. The data indicate that the most errors occurred on nouns followed by verbs for each sample. This tendency is as expected because nouns occur more frequently than verbs in Arabic. However, the difference between nouns and verbs varies largely among the error samples. The maximum difference occurs for the KACST TTS sample and the minimum for the Holy Quran sample. Diacritization errors on particles are rare or even absent because, regardless of their frequent usage in the language, they have fixed diacritization and the errors that may occur are due to orthographic similarity with other words in Arabic, such as مِنْ ''mino'' (from) and مَنْ ''mano''(who).

The main error pattern for verbs is confusion between passive and active verbs. This error type constitutes 31%, 44%, 43%, and 38% of the verb errors in the Holy Quran, Sahih Al-Bukhary, ATB, and Q + BUKH + ATB error samples. For KACST TTS, the main error pattern is confusion between present and past tense verbs. Only one instance of confusion between passive and active verbs was found in the KACST TTS error samples, possibly because passive verbs rarely occur in the KACST TTS dataset.

We investigated the error patterns for each error sample and found that the error samples generally share the same error patterns. The main error patterns for nouns in the error samples include confusion between different morphological patterns, syntactic confusion regarding the second word of the genitive construction, confusion between verb subject and verb object, named entities, and not assigning diacritics to the letter ا ''A'' at the beginning of a definitive noun. We found the last error pattern in the KACST TTS error sample only. Additionally, we did not find any errors regarding named entities in the Holy Quran error sample. Adding syntactic features may help eliminate such errors. Table 11 summarizes the distributions of these noun error patterns in the error samples.

TABLE 11. Distributions of error patterns among nouns in the error samples.

Pattern	Error sample				
	KACST TTS (%)	Holy Quran (%)	Sahih Al-Bukhary (%)	ATB (%)	Q+BUKH +ATB (%)
Morphological pattern	38%	59%	34%	24%	33%
Genitive construction	13%	8%	8%	33%	22%
Verb subject and verb object	15%	25%	27%	6%	2%
Named entities	8%	0%	20%	17%	14%
Not assigning diacritics to the letter ا ''A''	12%	0%	0%	0%	0%

TABLE 12. Numbers of instances of correct prediction with examples in the error samples.

Errors samples	Number of instances	Original sentence	Predicted diacritiz-ation
KACST TTS	10	أَلْوَاحِدَةُ وَالنِّصفُ إِلَّا دَقِيقَةً وَاحِدَةً فَجراً	وَالنِّصْفُ
ATB	3	.... فِي شَأْنِ ' الحالَةِ بَيْنَ العِراقِ وَالكُوَيْتِ....	وَالكُوَيْتِ
Q+BUKH+ATB	11	وَحَذَّرَ ممازِحا من أَنَّهُ إِذا تَكَرَّرَ الأَمْرُ في ماليزيا.....	مُمازِحاً

Additionally, for the KACST TTS, ATB, and Q + BUKH + ATB error samples, we noticed instances in which the models predicted the correct diacritization of letters when the diacritization was missing from the original sentence and the predicted diacritization was possible in Arabic. Table 12 lists the numbers of such instances in each error sample along with corresponding examples.

VII. COMPARISON

We employed two approaches to compare the proposed method with other available methods. The first approach involved comparing the performance results of the proposed method with those of other methods using the same data and data splits that have been utilized in previous studies.

We used the three parts of the ATB (parts 1, 2, and 3) and the same split as Diab *et al.* [35] to compare the performance of the proposed method with those of two methods that, to the best of our knowledge, achieved state-of-the-art published results for Arabic diacritization, namely, the methods of Zalmout and Habash [23] and Alqahtani *et al.* [33].

TABLE 13. Comparison of our proposed method with two state-of-the-art methods using ATB parts 1, 2, and 3.

Model	DER (%)	WER (%)	SER (%)
Zalmout and Habash [23]	-	8.3	-
Alqahtani et al. [33]	2.8	8.2	-
BiLSTM-CRF	2.34	9.85	5.02

Table 13 lists the results. Our proposed method achieved the best DER (2.34%) but yielded the worst WER among the considered methods. However, when we used random splits for training, validation, and testing (see Section VI), the proposed method produced a better WER (8.43%) and an even better DER (2.13%).

In the second comparison approach, we compared the resulting models (see Section VI) with three well-known and publicly available Arabic diacritization systems, namely, MADAMIRA[3] (morphological analysis-based), Farasa[4] (feature engineering-based), and the Belinkov and Glass model[5] (data-based). We based our comparison on three manually diacritized texts from three different genres with various writing styles and sentence structures. These texts were a Friday sermon[6] (1,854 words), a children's story[7] (927 words), and a classical Arabic poem[8] by the 10th-century poet Al-Mutanabbi (399 words).

Each text was divided into sentences, as described in Section IV. However, for the sentences that exceeded the maximum length of the model input, extra splits were performed to ensure that every sentence was within the length accepted by the model. To compare the performances of the models, a non-diacritic version of the testing dataset was also used.

We faced the following issues with the output of the three diacritization systems:

A. FARASA

The insertion of additional letters into the middle of words such as للتي and لِألّتي, the removal of some punctuation, and letter substitutions; for instance, the letter ا could be changed to ا and ة could be changed to ه.

B. MADAMIRA

Letter substitutions; the conversion of punctuation into letters, such as { } being changed to ؟ ئ; the insertion of question marks into the middle of words such as كَذ؟لِكَ and لـ؟كِنَّ; and the insertion of spaces between punctuation and letters.

C. BELINKOV AND GLASS MODEL

The insertion of question marks into the middle of words such as ذا؟ه and ك؟رب, diacritizing punctuation, the insertion of extra characters and new lines, and the replacement of some characters with others, such as changing ≫ to ؟ً؟ and { } to ؟ ئ.

[3] https://camel.abudhabi.nyu.edu/madamira/
[4] http://qatsdemo.cloudapp.net/farasa/demo.html
[5] https://github.com/boknilev/diacritization
[6] www.alminbar.net
[7] http://haybinyakzhan.blogspot.com/
[8] https://she3r.net

TABLE 14. Performances of all models when applied to the Friday sermon text.

Model	DER (%)	WER (%)	SER (%)
MADAMIRA	32.90	79.41	46.80
Farasa	23.13	66.56	**14.19**
Belinkov and Glass	32.90	79.41	46.80
KACST TTS-based model	17.93	48.47	27.88
Holy Quran-based model	13.39	37.12	22.31
Sahih Al-Bukhary-based model	10.35	29.99	17.39
ATB-based model	26.85	72.97	32.44
Q+BUKH+ATB-based model	**9.02**	**27.65**	15.11

TABLE 15. Performances of all models when applied to the Children's story text.

Model	DER (%)	WER (%)	SER (%)
MADAMIRA	36.19	81.44	48.44
Farasa	29.79	73.06	28.88
Belinkov and Glass	32.26	79.89	49.56
KACST TTS-based model	27.12	**61.88**	35.64
Holy Quran-based model	**24.84**	62.00	**31.89**
Sahih Al-Bukhary-based model	28.04	63.17	37.26
ATB-based model	36.95	82.07	45.68
Q+BUKH+ATB-based model	25.81	62.31	34.34

To compare the performances of our models with those of the other systems fairly, the resulting outputs should be standardized in terms of character order, and the sentence division must be identical to that of the original testing text. Therefore, we manually edited the diacritized text of each model by removing unnecessary insertions of letters and diacritics. Additionally, we only considered the Arabic characters in our DER, WER, and SER calculations, discarding non-Arabic letters and non-alphabetic characters such as numbers and punctuation.

Tables 14, 15, and 16 present the performances of all models when applied to a Friday sermon text, children's story, and classical Arabic poem, respectively. As expected, the error rates are high for our models and the other systems because the data used for comparison were not from the same discourses on which the models were trained.

The results indicate that our models outperformed the other systems when applied to the Friday sermon in terms of the DER and WER, where the Q + BUKH + ATB-based model achieved the best DER and WER of 9.02% and 27.65%, respectively, followed by the Sahih Al-Bukhary-based model. In addition to the prediction capabilities of the deep learning approaches, this performance can be ascribed to the genre similarity between the Friday sermon and the data from the Q + BUKH + ATB- and Sahih Al-Bukhary-based models, which were trained on religious texts. Farasa achieved the best SER (14.9%), perhaps because it was trained to

TABLE 16. Performances of all models when applied to the classical Arabic poem.

Model	DER (%)	WER (%)	SER (%)
MADAMIRA	42.07	87.22	55.64
Farasa	**34.47**	**74.87**	36.18
Belinkov and Glass	41.97	86.43	44.47
KACST TTS-based model	48.33	90.45	43.47
Holy Quran-based model	45.12	88.94	37.69
Sahih Al-Bukhary-based model	42.28	86.18	**32.66**
ATB-based model	40.08	83.92	42.46
Q+BUKH+ATB-based model	39.89	85.68	33.42

		Model 1	
		Correct	Wrong
Model 2	Correct	a	b
	Wrong	c	d

determine the proper word case endings. However, Farasa was followed by the Q + BUKH + ATB-based model with a marginal difference (15.11%).

For the children's story text, the Holy Quran-based model achieved the best DER (24.84%) and SER (31.89%) but came in second in terms of the WER (62.00%), with a marginal difference from the KACST TTS-based model (61.88%). Knowing that the lengths of the sentences in children's stories are generally short, the performances of these two models can be ascribed to the short length of the sentences in the training data.

For the classical Arabic poem, the performances of all models are not encouraging. The poor results may be due to the tendency of Arabic poetry not to repeat words more than once in the same poem and the extensive use of metaphors. These two characteristics would have caused the models to face two problems: out-of-vocabulary situations and new contexts for words not seen before. Any model may face such problems; however, these issues are largely present in poetry texts. Farasa achieved the best results for the classical Arabic poem in terms of the DER (34.47%) and WER (74.87%), followed by the Q + BUKH + ATB-based model (39.89% and 85.68%, respectively), whereas the Sahih Al-Bukhary-based model achieved the best SER (32.66%) followed by the Q + BUKH + ATB-based model (33.42%). The good performance of Farasa may have resulted from the large amount of training data (9.7 million tokens) and their diversity [32].

We used the McNemar test [48], [49], also known as the within-subjects chi-squared test, to check whether the differences in DER between the best performing model and other models were statistically significant. We selected the DER because our approach is based on character diacritization.

Suppose we have the following 2 × 2 confusion matrix:

The McNemar test statistic can be computed using the following formula:

$$McNemar\ test\ statistic = \frac{(b-c)^2}{(b+c)}. \tag{8}$$

TABLE 17. McNemar test statistics for best performing models for the test texts.

Model	Friday sermon	Children's story	Arabic poem
	Q+BUKH+ATB-based model	Holy Quran-based mode	Farasa
MADAMIRA	1556.49	55.28	39.07
Farasa	658.21	32.51	-
Belinkov and Glass	1718.49	136.12	37.12
KACST TTS-based model	240.11	9.60	99.68
Holy Quran-based model	35.68	-	59.43
Sahih Al-Bukhary-based model	**12.71**	12.11	23.98
ATB-based model	1157.50	50.24	22.15
Q+BUKH+ATB-based model	-	**8.57**	**16.85**

The minimum accepted p value for the test is 0.05, which corresponds to a 95% confidence interval; that is, the McNemar test statistic should be equal to or greater than 3.841. Table 17 illustrates the McNemar test statistic values for the best performing model and other models for each testing set.

Considering the smallest McNemar test statistic (12.7), which occurs for the Friday sermon text, the performance difference between the Q + BUKH + ATB-based model and the other models is significant with a minimum p value of 0.0003 (99.97% confidence interval). The difference in performance between the Holy Quran-based model and the other models on the children's story is significant, with a minimum p value of 0.003 (99.9% confidence interval). For Farasa and the other models, the difference is also significant for the Arabic poems, with a minimum p value of 0.00004 (99.996% confidence interval).

VIII. CONCLUSION

In this study, we examined sequence-to-sequence tagging using a BiLSTM neural network with CRF for Arabic diacritization. Our approach does not involve a morphological analyzer, dictionary, or feature engineering. The results suggest that the approach is affected by the quality of diacritization and then by the size of the data.

Comparing the performance of the proposed method with those of existing models using benchmarking data and data splits for training, validation, and testing, the proposed method achieved, to the best of our knowledge, state-of-the-art DER results (2.37%) for ATB parts 1, 2, and 3.

Additionally, we compared the proposed approach with three well-known and publicly available Arabic diacritization systems using three texts from different genres. The results revealed that our models outperform all models except for Farasa, which yielded the best results for the classical Arabic poem.

Our proposed method generally achieved the best DERs because it is based on characters, whereas other Arabic

diacritization systems are based on words, in addition to its prediction capabilities resulting from combining deep learning networks with CRF. This performance is also reflected in the WER in general. Like any other deep learning approach, our Arabic diacritization approach requires larger amounts of memory, computational resources, and time than other machine learning approaches. However, deep learning approaches can extract useful features solely from data and produce better results than other machine learning algorithms, specifically when combined with CRF.

Our experiments and comparisons with other systems demonstrate that a robust Arabic diacritization system can be learned from data if the training set is large and fully and consistently diacritized and covers a large range of topics.

REFERENCES

[1] Y. Hifny, "Open vocabulary Arabic diacritics restoration," *IEEE Signal Process. Lett.*, vol. 26, no. 10, pp. 1421–1425, Oct. 2019.

[2] Z. Ozer, I. Ozer, and O. Findik, "Diacritic restoration of turkish tweets with word2vec," *Eng. Sci. Technol., Int. J.*, vol. 21, no. 6, pp. 1120–1127, Dec. 2018.

[3] M. Nutu, B. Lorincz, and A. Stan, "Deep learning for automatic diacritics restoration in romanian," in *Proc. IEEE 15th Int. Conf. Intell. Comput. Commun. Process. (ICCP)*, Sep. 2019, pp. 235–240.

[4] M. Al-Badrashiny, A. Hawwari, and M. Diab, "A layered language model based hybrid approach to automatic full diacritization of Arabic," in *Proc. 3rd Arabic Natural Lang. Process. Workshop*, 2017, pp. 177–184.

[5] A. Masmoudi, S. Mdhaffar, R. Sellami, and L. H. Belguith, "Automatic diacritics restoration for tunisian dialect," *ACM Trans. Asian Low-Resource Lang. Inf. Process.*, vol. 18, no. 3, pp. 1–18, Jul. 2019.

[6] H. Mubarak, A. Abdelali, K. Darwish, M. Eldesouki, Y. Samih, and H. Sajjad, "A system for diacritizing four varieties of Arabic," in *Proc. Conf. Empirical Methods Natural Lang. Process. 9th Int. Joint Conf. Natural Lang. Process. (EMNLP-IJCNLP), Syst. Demonstration*, Nov. 2019, pp. 217–222.

[7] I. Orife, "Attentive sequence-to-sequence learning for diacritic restoration of Yorùbá language text," in *Proc. Interspeech*, Sep. 2018, pp. 2848–2852.

[8] I. H. Ali, Z. Mnasri, and Z. Laachri, "Gemination prediction using DNN for Arabic text-to-speech synthesis," in *Proc. 16th Int. Multi-Conf. Syst., Signals Devices (SSD)*, Mar. 2019, pp. 366–370.

[9] S. Abed, M. Alshayeji, and S. Sultan, "Diacritics effect on Arabic speech recognition," *Arabian J. Sci. Eng.*, vol. 44, no. 11, pp. 9043–9056, Nov. 2019.

[10] S. Alqahtani, H. Aldarmaki, and M. Diab, "Homograph disambiguation through selective diacritic restoration," in *Proc. 4th Arabic Natural Lang. Process. Workshop*, 2019, p. 49.

[11] A. Kadim and A. Lazrek, "Parallel HMM-based approach for Arabic part of speech tagging," *Int. Arab J. Inf. Technol.*, vol. 15, no. 2, pp. 341–351, 2018.

[12] Y. Hifny, "Hybrid LSTM/MaxEnt networks for Arabic syntactic diacritics restoration," *IEEE Signal Process. Lett.*, vol. 25, no. 10, pp. 1515–1519, Oct. 2018.

[13] 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 *Proc. 6th Workshop Asian Transl.*, 2019, pp. 215–225.

[14] Z. Huang, W. Xu, and K. Yu, "Bidirectional LSTM-CRF models for sequence tagging," 2015, *arXiv:1508.01991*. [Online]. Available: http://arxiv.org/abs/1508.01991

[15] V. Yadav and S. Bethard, "A survey on recent advances in named entity recognition from deep learning models," in *Proc. 27th Int. Conf. Comput. Linguistics*, Aug. 2018, pp. 2145–2158.

[16] H. Kim, S. Yang, and Y. Ko, "How to utilize syllable distribution patterns as the input of LSTM for korean morphological analysis," *Pattern Recognit. Lett.*, vol. 120, pp. 39–45, Apr. 2019.

[17] P. Gupta, K. Saxena, U. Yaseen, T. Runkler, and H. Schütze, "Neural architectures for fine-grained propaganda detection in news," in *Proc. 2nd Workshop Natural Lang. Process. Internet Freedom, Censorship, Disinformation, Propaganda*, 2019, p. 92.

[18] N. Habash and O. Rambow, "Arabic diacritization through full morphological tagging," in *Proc. Hum. Lang. Technol., Conf. North Amer. Chapter Assoc. Comput. Linguistics; Companion Volume, Short Papers (NAACL)*, 2007, pp. 53–56.

[19] N. Habash, O. Rambow, and R. Roth, "MADA+ TOKAN: A toolkit for Arabic tokenization, diacritization, morphological disambiguation, POS tagging, stemming and lemmatization," in *Proc. 2nd Int. Conf. Arabic Lang. Resour. Tools*, vol. 41, Cairo, Egypt, Apr. 2009, p. 62.

[20] R. Roth, O. Rambow, N. Habash, M. Diab, and C. Rudin, "Arabic morphological tagging, diacritization, and lemmatization using lexeme models and feature ranking," in *Proc. 46th Annu. Meeting Assoc. Comput. Linguistics Hum. Lang. Technol. Short Papers (HLT)*, 2008, pp. 117–120.

[21] A. Pasha, M. Al-Badrashiny, M. Diab, A. El Kholy, R. Eskander, N. Habash, and R. M. Roth, "MADAMIRA: A fast, comprehensive tool for morphological analysis and disambiguation of Arabic," in *Proc. 9th Int. Conf. Lang. Resour. Eval., Eur. Lang. Resour. Assoc.*, Jan. 2014, pp. 1094–1101.

[22] A. Shahrour, S. Khalifa, and N. Habash, "Improving Arabic diacritization through syntactic analysis," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, Sep. 2015, pp. 1309–1315.

[23] N. Zalmout and N. Habash, "Don't throw those morphological analyzers away just yet: Neural morphological disambiguation for Arabic," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2017, pp. 704–713.

[24] A. S. Hussein, M. A. Rashwan, and A. F. Atiya, "Arabic full text diacritization using light layered approach," *Artif. Intell. Mach. Learn. J.*, vol. 16, no. 1, pp. 1687–4846, 2016.

[25] A. Chennoufi and A. Mazroui, "Morphological, syntactic and diacritics rules for automatic diacritization of Arabic sentences," *J. King Saud Univ.-Comput. Inf. Sci.*, vol. 29, no. 2, pp. 156–163, Apr. 2017.

[26] A. Said, M. El-Sharqwi, A. Chalabi, and E. Kamal, "A hybrid approach for Arabic diacritization," in *Proc. Int. Conf. Appl. Natural Lang. Inf. Syst.* Berlin, Germany: Springer, Jun. 2013, pp. 53–64.

[27] I. Zitouni, J. S. Sorensen, and R. Sarikaya, "Maximum entropy based restoration of Arabic diacritics," in *Proc. 21st Int. Conf. Comput. Linguistics 44th Annu. Meeting (ACL)*, Jul.2006, pp. 577–584.

[28] M. A. A. Rashwan, A. A. Al Sallab, H. M. Raafat, and A. Rafea, "Deep learning framework with confused sub-set resolution architecture for automatic Arabic diacritization," *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 23, no. 3, pp. 505–516, Mar. 2015.

[29] K. Darwish, H. Mubarak, and A. Abdelali, "Arabic diacritization: Stats, rules, and hacks," in *Proc. 3rd Arabic Natural Lang. Process. Workshop*, 2017, pp. 9–17.

[30] G. A. Abandah, A. Graves, B. Al-Shagoor, A. Arabiyat, F. Jamour, and M. Al-Taee, "Automatic diacritization of Arabic text using recurrent neural networks," *Int. J. Document Anal. Recognit.*, vol. 18, no. 2, pp. 183–197, Jun. 2015.

[31] Y. Belinkov and J. Glass, "Arabic diacritization with recurrent neural networks," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2015, pp. 2281–2285.

[32] H. Mubarak, A. Abdelali, H. Sajjad, Y. Samih, and K. Darwish, "Highly effective Arabic diacritization using sequence to sequence modeling," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics, Hum. Lang. Technol.*, vol. 1, Jun. 2019, pp. 2390–2395.

[33] S. Alqahtani, A. Mishra, and M. Diab, "Efficient convolutional neural networks for diacritic restoration," in *Proc. Conf. Empirical Methods Natural Lang. Process. 9th Int. Joint Conf. Natural Lang. Process. (EMNLP-IJCNLP)*, Nov. 2019, pp. 1442–1448.

[34] M. S. Khorsheed, "Diacritizing Arabic text using a single hidden Markov model," *IEEE Access*, vol. 6, pp. 36522–36529, 2018.

[35] M. T. Diab, N. Y. Habash, O. C. Rambow, and R. M. Roth, "LDC Arabic treebanks and associated corpora: Data divisions manual," 2013, *arXiv:1309.5652*. [Online]. Available: https://arxiv.org/abs/1309.5652

[36] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, Oct. 1986.

[37] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Trans. Neural Netw.*, vol. 5, no. 2, pp. 157–166, Mar. 1994.
[38] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.
[39] C. Sutton, K. Rohanimanesh, and A. McCallum, "Dynamic conditional random fields: Factorized probabilistic models for labeling and segmenting sequence data," in *Proc. 21st Int. Conf. Mach. Learn. (ICML)*, 2004, pp. 282–289.
[40] C. Sutton and A. McCallum, "An introduction to conditional random fields for relational learning," in *Introduction to Statistical Relational Learning*, L. Getoor and B. Taskar, Ed. Cambridge, MA, USA: MIT Press, 2007.
[41] C. Sutton, K. Rohanimanesh, and A. McCallum, "Dynamic conditional random fields: Factorized probabilistic models for labeling and segmenting sequence data," in *Proc. 21st Int. Conf. Mach. Learn. (ICML)*, 2004, pp. 693–723.
[42] A. McCallum and W. Li, "Early results for named entity recognition with conditional random fields, feature induction and Web-enhanced lexicons," in *Proc. 7th Conf. Natural Lang. Learn. (HLT-NAACL)*, 2003, pp. 188–191.
[43] I. Almosallam, A. AlKhalifa, M. Alghamdi, M. Alkanhal, and A. Alkhairy, "SASSC: A standard Arabic single speaker corpus," in *Proc. 8th ISCA Workshop Speech Synthesis*, 2013, pp. 249–253.
[44] M. Maamouri, A. Bies, T. Buckwalter, and W. Mekki, "The Penn Arabic treebank: Building a large-scale annotated Arabic corpus," in *Proc. NEMLAR Conf. Arabic Lang. Resour. Tools*, Cairo, Egypt, vol. 27, 2004, pp. 466–467.
[45] W. Ling, C. Dyer, A. W. Black, I. Trancoso, R. Fermandez, S. Amir, L. Marujo, and T. Luis, "Finding function in form: Compositional character models for open vocabulary word representation," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2015, pp. 1520–1530.
[46] P. Ding, X. Zhou, X. Zhang, J. Wang, and Z. Lei, "An attentive neural sequence labeling model for adverse drug reactions mentions extraction," *IEEE Access*, vol. 6, pp. 73305–73315, 2018.
[47] M. Al-Smadi, B. Talafha, M. Al-Ayyoub, and Y. Jararweh, "Using long short-term memory deep neural networks for aspect-based sentiment analysis of Arabic reviews," *Int. J. Mach. Learn. Cybern.*, vol. 10, no. 8, pp. 2163–2175, Aug. 2019.
[48] Q. McNemar, "Note on the sampling error of the difference between correlated proportions or percentages," *Psychometrika*, vol. 12, no. 2, pp. 153–157, Jun. 1947.
[49] T. G. Dietterich, "Approximate statistical tests for comparing supervised classification learning algorithms," *Neural Comput.*, vol. 10, no. 7, pp. 1895–1923, Oct. 1998.