IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received May 4, 2018, accepted June 12, 2018, date of publication July 3, 2018, date of current version July 25, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2852619

Diacritizing Arabic Text Using a Single Hidden Markov Model

MOHAMMAD S. KHORSHEED
Computer Laboratory, University of Cambridge, Cambridge CB3 0FD, U.K.
National Center of Robotics and Intelligent Systems, King Abdulaziz City for Science and Technology, Riyadh 11442, Saudi Arabia
e-mail: mkhorshd@kacst.edu.sa

This work was supported by a Grant from King Abdulaziz City for Science and Technology in Saudi Arabia through the National Plan for Science, Technology, and Innovation.

ABSTRACT The main obstacle in the development of a robust Arabic text-to-speech system is the ability to place diacritics, that is, small marks above and below the raw text that indicate the correct pronunciation. In this paper, we propose a system that can retrieve the diacritics that match a given Arabic text. First, the system injects the raw text into an engine that is based on a single hidden Markov model. The engine then generates an optimal path through its states. The system finally matches the state sequence with its equivalent diacritics and sets them in place within the text. Experiential results on diverse data sets demonstrate the robustness of the proposed system, even with samples that are novel to the system.

INDEX TERMS Computational and artificial intelligence, hidden Markov models, machine intelligence, pattern analysis, text processing.

I. INTRODUCTION

The spelling characters in an Arabic word do not solely determine the pronunciation of that word. This creates a technological challenge when a designing a text-to-speech system for Arabic [1]–[3]. Thus, the orthographic system for Arabic places small marks above and below the main strokes that are called diacritics. Diacritics are essential for the correct pronunciation of a word and to determine its meaning, accordingly. The accuracy of text diacritization directly affects speech quality [4], [5]. Arabic script readers are accustomed to inferring the meaning of undiacritized text. However, some sentences may require even native Arabic speakers to read them repeatedly before determining the correct meaning. Modern Arabic writing omits these diacritics, which leads to considerable ambiguity among words that have identical consonants but different meanings [6]. Some researchers have measured the average ambiguities for a token in Arabic as 19.2, which is more than eight times that for most other languages [7]. Some words can have even a higher number of analyses [8].

The sound system in Arabic consists of three main groups: short vowels, nunation (tanween), and syllabification (shadda and sukun) [9]. The first two groups include six basic diacritics that may also be combined with the diacritic shadda. The diacritic sukun always appears alone. This results in 13 diacritics, as illustrated in Table 1. Consider the word in Table 2 which consists of the character sequence /*d*, /*r* and /*s*. There are five potential pronunciations, each with a different meaning based on the set of diacritics that accompany the letters.

Some researchers have developed systems to restore diacritics based on the morphological, syntactic, and semantic

TABLE 1. Arabic diacritics.

Group	Diacritic	Position	Label	Example
Short vowels	Fatha	above	MFA	عَدَل
	Dhamma	above	MDA	عُدُل
	Kasra	below	MKA	عِدِل
Short vowels + Shadda	Fatha + Shadda	above	MFS	عَدَّل
	Dhamma + Shadda	above	MDS	عُدُّ
	Kasra + Shadda	above	MKS	عِدِّل
Nunation (Tanween)	Double Fatha	above	MDF	عدلاً
	Double Dhamma	above	MDD	عدلٌ
	Double Kasra	below	MDK	عدلٍ
Nunation (Tanween) + Shadda	Double Fatha + Shadda	above	DSF	علّيًا
	Double Dhamma + Shadda	above	DSD	علّيٌ
	Double Kasra + Shadda	above	DSK	علّيٍ
Syllabic	Sukun	above	MSU	عدْل

TABLE 2. Sample word with five sets of diacritics, and their translations and transliterations.

Diacritized word	Translation	Transliteration
دَرَسَ	Studied	/darasa
دَرْسَ	Lesson	/dars
دَرَّسَ	Edify	/darrasa
دُرِسَ	Been studied	/doresa
دُرِّسَ	Been taught	/dorresa

rules of Arabic [10], [11]. These rules are systematic, but complex [12], [13]. Morphological rules decompose the undiacritized word into its morphological entities: the class of the word, prefix, root, form, and suffix. Syntactic rules determine which diacritics accompany the last character of the word. Semantic rules resolve ambiguous cases. Rule-based diacritization is vocabulary independent; however, it is complex, time-consuming, and difficult to regularly update the rules or extend them to other Arabic dialects [14].

Another widespread approach is to use a machine learning technique, such as a recurrent neural network, to build high-level linguistic abstractions of text [15]. Other researchers have implemented hidden Markov models (HMMs) at the word level to obtain the right vowelization of words based mainly on their context. The HMMs observe the undiacritized text, and the output is a sequence of possible diacritized words [16]–[19]. Dynamic programming has also been applied to efficiently restore the optimal sequence of diacritics based on monogram [20], bigram [21], trigram [22], or n-gram models [23]. The maximum entropy framework is another statistical model that integrates different sources of knowledge to enhance the performance of diacritization [24], [25].

Combining linguistic knowledge with stochastic techniques has resulted in a hybrid system that determines the most likely diacritics for a given undiacritized text, then factorizes each Arabic word into all its possible morphological constituents, and finally shortlists the sequence of morphemes that is the most likely diacritization [1], [26].

II. PROBLEM STATEMENT

In this study, we inject Arabic script into the diacritization system as a sequence of ASCII codes of the text characters. This sequence represents the observation set for the diacritization system that aims to restore the full diacritics of the given script. Markov models are restrictive in the diacritization process because they match each observation to a distinct state within the model, whereas, in reality, each state (diacritic) has a probabilistic function that generates the possible observations (ASCII codes of the script letters). In this section, we extend the implementation of the diacritization process using HMMs.

A. CONCEPTUAL UNDERSTANDING

Assume that a model is observing a set of random numbers that represent the ASCII codes of undiacritized text. It then attempts to generate an optimal output sequence that best matches the diacritic set that accompanies that text. The set of states/diacritics is hidden; however, it can be observed through the stochastic processes that generate the observations/characters. At any instant of time, the model observes the ASCII code of one character from an undiacritized text line, and then steps to one of its states that represents the equivalent diacritic.

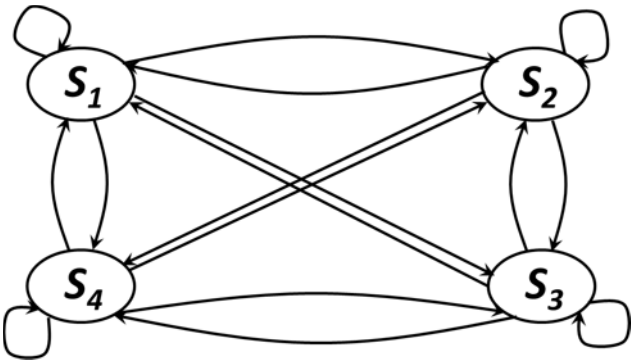


FIGURE 1. Ergodic HMM with four states.

The proposed system is an ergodic HMM. In such a model, a transition from one state to another, including itself, takes place in a single step, as shown in Fig. 1. The model is described at any time ($t = 1, 2, \cdots, T$) as being in one of the states $S_1, S_2, \cdots, S_N$, where N denotes the number of states. At each time, the model either transitions to another state or remains in its current state. This is interpreted as the model transitioning from state q_t at time t to state q_{t+1} at time $t + 1$. The transition from one state to another occurs according to a probability set that is associated with each state. These activities may be described using the current state and its predecessors:

$$P[q_t = j|q_{t-1} = i, q_{t-2} = k, \cdots, q_2 = y, q_1 = z] \quad (1)$$

where $t = 1, 2, \cdots$. The model under consideration only looks at the immediate preceding state:

$$P[q_t = j|q_{t-1} = i, q_{t-2} = k, \cdots, q_2 = y, q_1 = z] = P[q_t = j|q_{t-1} = i] \quad (2)$$

Moreover, only the processes that are independent of time are considered. This leads to a set of state transition probabilities:

$$a_{ij} = P[q_t = S_j|q_{t-1} = S_i] \quad a_{ij} \geq 0 \quad (3)$$

The observation is a probabilistic function of the state. This means that the states are not directly observable. The model has 15 distinct states, that is $N = 15$. Table 3 lists the 13 possible diacritics, in addition to the start state (*STR*) and non-diacritic sign state (*NON*). The model always commences

TABLE 3. Numbering and labeling the model states.

Number	Label	Number	Label	Number	Label
1	STR	6	MDS	11	DSF
2	MFA	7	MKS	12	DSD
3	MDA	8	MDF	13	DSK
4	MKA	9	MDD	14	MSU
5	MFS	10	MDK	15	NON

from the start state (STR) at $i = 1$. It then transitions from one state (diacritic) either to another state (diacritic) or to itself. The latter occurs when the same diacritic appears twice consecutively within the same word. This scheme affects the initial state probability π as follows:

$$\pi_i = \begin{cases} 1 & i = 1 \\ \epsilon & i \neq 1, \quad \epsilon \approx 0 \end{cases} \tag{4}$$

To calculate the state transition coefficients, we first need to derive the corpus statistics that include the zero-order and first-order diacritic transition probability:

$$P_0(\alpha) = \frac{\text{No. of times diacritic } \alpha \text{ appears at the beginning}}{\text{Total no. of lines in the corpus}} \tag{5}$$

$$P_1(\alpha \rightarrow \beta) = \frac{\text{No. of transitions from diacritic } \alpha \text{ to diacritic } \beta}{\text{No. of transitions from diacritic } \alpha} \tag{6}$$

We mentioned previously that the model is ergodic, however because of the nature of the problem, there are two constraints that apply to state transitions. The first constraint prevents any state transition to STR because it only appears at the beginning of the state sequence. Therefore, no transition from any state, including STR, to STR is permitted. We assign ϵ, where $\epsilon \approx 0$, to all state transition coefficients a_{ij} when $j = 1$.

The second constraint is based on the syntactic rule that nunation appears at the end of a word; hence, only a space or the end of the line may occur. This forces the model, when residing in any of the nunation states, MDF, MDD, MDK, DSF, DSD, DSK, to transition only to NON, or when $i \in \{8, 9, 10, 11, 12, 13\} \Rightarrow j = 15$. The state transition probability is set to ϵ otherwise.

Apart from the previous two cases, the state transition probability is calculated as

$$a_{ij} = \begin{cases} P_0(j) & i = 1, j \neq 1 \\ P_1(i, j) & otherwise \end{cases} \tag{7}$$

There are M distinct observation symbols per state, where $M = 110$, which correspond to the actual output of the system being modeled. For Arabic text diacritization, the observation symbols are the ASCII codes of individual characters that represent the undiacritized text. They are denoted by $V = v_1, v_2, \cdots, v_M$. The probability of observing symbol v_k while the model is in state i at time t is referred to as the observation symbol probability:

$$b_j(k) = P[o_t = v_k | q_t = i] \quad 1 \leqslant k \leqslant M \tag{8}$$

When $j = 1$ this refers to the probability of a letter appearing at the beginning of a character sequence: $b_1(k) = P_{initial}(v_k)$, where

$$P_{initial}(v_k) = \frac{\text{No. of character sequences starting with the letter } v_k}{\text{Total no. of lines in the corpus}} \tag{9}$$

Not all the 110 observations are eligible to start a word, and hence, a sentence or character sequence. Fig. 2 illustrates some letters that cannot be at the initial position. For those letters, $b_1(v_k) = \epsilon$, where $\epsilon \approx 0$.

FIGURE 2. Arabic letters not eligible to initiate a sentence.

Other than the initial letter, $j \neq 1$, the symbol probability, $b_j(v_k)$, is calculated as follows:

$$P_{S_j}(v_k) = \frac{\text{No. of times letter } v_k \text{ appears with diacritic } S_j}{\text{Total no.of times diacritic } S_j \text{ appears in the corpus}} \tag{10}$$

Specifying the two model parameters N and M, the observation symbols, and three sets of probability measures A, B and π is essential to providing a complete specification of an HMM. The model is denoted by $\lambda = (A, B, \pi)$

B. COMPUTING THE LETTER SEQUENCE LIKELIHOOD

The model generates the output sequence using the following parameters: λ, M, N, observation sequence, and state sequence. The model always starts from the initial state: $q_1 = 1$ or STR at $t = 1$. While it resides at STR, the model observes the first sample, which symbolizes the ASCII code of the first character within the undiacritized text. The model then transitions to q_2, which represents the diacritic of the first sample previously observed. Next, the model enters an iterative loop in which it observes one sample from the undiacritized text each iteration, and transitions to new state $q_{t+1} = j$ according to a_{ij}. The state label reflects the diacritic of the character sample observed in the previous step. The iterative process ends when there are no more observations, that is, when it finishes diacritizing the entire text. Typically, the character sequence ends with an end-of-line character. This has a null observation.

The character sequence $/d/r/s$ shown in Table 2 represents the observation sequence of the model, where each diacritized sequence has a different likelihood probability. We inject this pattern into our model as $/d\ /r\ /s\ null$. The model starts at state STR, observes letter $/d$, transitions to q_2, observes letter $/r$, transitions to q_3, observes letter $/s$, and finally transitions to q_4 and observes null. The probability of this observation

sequence for a specific state sequence is

$$P(O|Q,\lambda) = \prod_{t=1}^{4} P(o_t|q_t,\lambda) = b_1(/d) \times b_{q_2}(/r) \times b_{q_3}(/s) \times b_{q_4}(null) \quad (11)$$

and the probability of the state sequence is

$$P(Q|\lambda) = a_{1\to q_2} \times a_{q_2\to q_3} \times a_{q_3\to q_4} \quad (12)$$

Therefore, the probability of the observations given the model is

$$P(O|\lambda) = \sum_{Q} P(O|Q,\lambda)P(Q|\lambda) = \sum_{q_1q_2q_3q_4} b_1(/d)a_{1\to q_2} \times b_{q_2}(/r)a_{q_2\to q_3} \times b_{q_3}(/s)a_{q_3\to q_4} \times b_{q_4}(null) \quad (13)$$

This allows the evaluation of the probability of the observation sequence; however, the direct evaluation of this sequence would be exponential in T.

Consider the transliterated character sequences shown in Table 2, where each sequence represents a potential outcome of the model. We calculate the probabilities of the observations */d /r /s null* for various state sequences.

The first combination, */darasa*, is the model outcome of the state sequence *STR*, *MFA*, *MFA*, *MFA*. We calculate the probability of the observation sequence given the first state sequence as

$$P(/d/r/s\ null|STR\quad MFA\quad MFA\quad MFA,\lambda) = b_1(/d)a_{1\to MFA} \times b_{MFA}(/r)a_{MFA\to MFA} \times b_{MFA}(/s)a_{MFA\to MFA} \times b_{MFA}(null) \quad (14)$$

We may substitute state labels for state numbers:

$$P(/d/r/s\ null|1,2,2,2\ ,\lambda) = b_1(/d)a_{1\to 2} \times b_2(/r)a_{2\to 2} \times b_2(/s)a_{2\to 2} \times b_2(null) \quad (15)$$

The second sequence /dars is the model outcome of the state sequence *STR*, *MFA*, *MSU*, *MSU*. The likelihood probability given this state sequence is

$$P(/d/r/s\ null|STR\quad MFA\quad MSU\quad MSU,\lambda) = b_1(/d)a_{1\to MFA} \times b_{MFA}(/r)a_{MFA\to MSU} \times b_{MSU}(/s)a_{MSU\to MSU} \times b_{MSU}(null) \quad (16)$$

Equally,

$$P(/d/r/s\ null|1,2,14,14\ ,\lambda) = !b_1(/d)a_{1\to 2} \times b_2(/r)a_{2\to 14} \times b_{14}(/s)a_{14\to 14} \times b_{14}(null) \quad (17)$$

The third sequence /darrasa is the model outcome of the state sequence *STR*, *MFA*, *MFS*, *MFA*:

$$P(/d/r/s\ null|STR\quad MFA\quad MFS\quad MFA,\lambda) = b_1(/d)a_{1\to MFA} \times b_{MFA}(/r)a_{MFA\to MFS} \times b_{MFS}(/s)a_{MFS\to MFA} \times b_{MFA}(null) \quad (18)$$

or

$$P(/d/r/s\ null|1,2,5,2\ ,\lambda) = b_1(/d)a_{1\to 2} \times b_2(/r)a_{2\to 5} \times b_5(/s)a_{5\to 2} \times b_2(null) \quad (19)$$

We calculate the fourth observation sequence /doresa given the state sequence *STR*, *MDA*, *MKA*, *MFA* as

$$P(/d/r/s\ null|STR\quad MDA\quad MKA\quad MFA,\lambda) = b_1(/d)a_{1\to MDA} \times b_{MDA}(/r)a_{MDA\to MKA} \times b_{MKA}(/s)a_{MKA\to MFA} \times b_{MFA}(null) \quad (20)$$

or using state numbers

$$P(/d/r/s\ null|1,3,4,2\ ,\lambda) = b_1(/d)a_{1\to 3} \times b_3(/r)a_{3\to 4} \times b_4(/s)a_{4\to 2} \times b_2(null) \quad (21)$$

Finally, the likelihood probability of the observation sequence /dorresa given the state sequence *STR*, *MDA*, *MKS*, *MFA* is calculated as

$$P(/d/r/s\ null|STR\quad MDA\quad MKS\quad MFA,\lambda) = b_1(/d)a_{1\to MDA} \times b_{MDA}(/r)a_{MDA\to MKS} \times b_{MKS}(/s)a_{MKS\to MFA} \times b_{MFA}(null) \quad (22)$$

Alternatively,

$$P(/d/r/s\ null|1,3,7,2\ ,\lambda) = b_1(/d)a_{1\to 3} \times b_3(/r)a_{3\to 7} \times b_7(/s)a_{7\to 2} \times b_2(null) \quad (23)$$

C. RETRIEVE THE DIACRITICAL SEQUENCE

In this paper, we aim to build an HMM-based engine that can retrieve a set of diacritics that best matches a given undiacritized Arabic text. However, as the letter sequence expands, the system could have unlimited state sequence possibilities. Therefore, given the observation sequence O, the objective is to attempt to uncover the hidden part of the model, that is, determine the optimal state sequence that best explains this observation sequence. The optimality criterion is selected to maximize $P(O|\lambda)$ over a complete state sequence. The Viterbi algorithm is a formal technique to determine this optimum path based on dynamic programming methods [27]. For natural language applications, such as Arabic text diacritization, the number of observations is very large; moreover, the state transition and observation symbol probabilities are very small. This may cause the machine to run out of precision and therefore generate unreliable computations. To overcome this problem and expand the precision, we take the logarithms of the model parameters at the beginning:

$$\begin{aligned} \hat{\pi}_i &= log(\pi_i)\ 1 \leqslant i \leqslant N \\ \hat{a}_{ij} &= log(a_{ij})\ 1 \leqslant i,j \leqslant N \\ \hat{b}_i(o_t) &= log(b_i(o_t))\ 1 \leqslant i \leqslant N, 1 \leqslant t \leqslant T \end{aligned} \quad (24)$$

We assign the value ϵ, where $\epsilon \approx 0$, rather than zero to π, a, and b to avoid an undefined number or negative infinity.

TABLE 4. Samples from the three datasets.

Datasets	Samples
DS_1	استوعبت الحضارة الإسلامية العلوم والمعارف وولفت منها توليفة غير مسبوقة. اِسْتَوْعَبَتْ اَلْحَضَارَةُ اَلإِسْلامِيَّةُ اَلْعُلُوْمَ وَاَلْمَعَارِفَ وَوَلَّفَتْ مِنْهَا تَوْلِيْفَةً غَيْرَ مَسْبُوْقَةٍ.
DS_2	ولله ملك السماوات والأرض يغفر لمن يشاء ويعذب من يشاء وكان الله غفورا رحيما وَلِلَّهِ مُلْكُ السَّمَاوَاتِ وَالأَرْضِ يَغْفِرُ لِمَن يَشَاءُ وَيُعَذِّبُ مَن يَشَاءُ وَكَانَ اللَّهُ غَفُورًا رَّحِيمًا
DS_3	أحمد عبدالرحمن العدناني أَحْمَدُ عَبْدُالرَّحْمَن اَلْعَدْنَانِيُّ

The algorithm proceeds in four steps without any multiplications:

- Initialization

$$\delta_1(i) = \hat{\pi}_i + \hat{b}_i(o_1)$$
$$\varphi_1(i) = 01 \leqslant i \leqslant N \tag{25}$$

Only $\hat{\pi}_1 = 0$ and the remaining initial state probabilities equal $log\ (\epsilon)$. In this study, we assign $\epsilon = 10^{-100} \rightarrow log(\epsilon) = -100$. This favors only paths that start from *STR*:

$$\delta_1(i) = \begin{cases} \hat{b}_1(o_1) & i = 1 \\ \hat{b}_i(o_1) - 100 & i \neq 1 \end{cases} \tag{26}$$

- Recursion: $1 \leqslant i \leqslant N, 2 \leqslant t \leqslant T$

$$\delta_t(i) = \max_{1 \leq j \leq N} [\delta_{t-1}(j) + \hat{a}_{ji}] + \hat{b}_i(o_t)$$
$$\varphi_t(i) = arg \max_{1 \leq j \leq N} [\delta_{t-1}(j) + \hat{a}_{ji}] \tag{27}$$

- Termination:

$$P^* = \max_{1 \leq j \leq N} [\delta_T(j)]$$
$$q_T^* = arg \max_{1 \leq j \leq N} [\delta_T(j)] \tag{28}$$

- Path backtracking: $t = T-1, T-2, \cdots, 3, 2, 1$

$$q_t^* = \varphi_{t+1}(q_{t+1}^*) \tag{29}$$

III. IMPLEMENTATION AND EVALUATION RESULTS

In this section, we present the implementation of the proposed system to process Arabic corpora to diacritize raw text that is injected as a sequence of observations. The proposed system consists of a 15-state ergodic HMM. We developed a complete software program based on the HMM engine. Using this program, the user may either type an Arabic phrase that the system then diacritizes, or simply import an Arabic text file into the graphical user interface (GUI) window. This program passes through three modules. The first module reads the text line and transfers it into a sequence of predefined observation symbols. The second module generates the output sequence: the diacritics. The final module arranges the output sequence with the raw text by imposing the diacritics above and below the equivalent characters. Fig. 3 shows two screenshots that illustrate the input raw text and the final fully diacritized text.

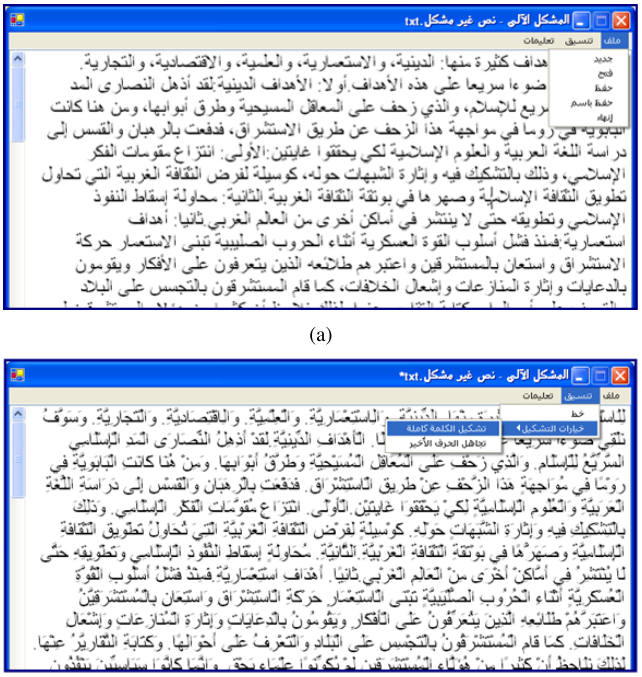


FIGURE 3. Screenshots from the program developed as a GUI. (a) Input raw text. (b) Fully diacritized text.

A. THREE DATASETS

To assess the performance of the proposed system, we applied the HMM-based engine to an Arabic text corpus. The corpus consisted of three datasets. Table 4 shows one text sample from each dataset. Table 5 illustrates various statistics

TABLE 5. Data analytics of the three datasets.

Features	Datasets		
	DS_1	DS_2	DS_3
Number of sentences without error	24 274	5 308	54 463
Number of sentences with error	289	1 040	0
Number of words	252 541	61 511	141 558
Average number of words/sentence	10,40	11,59	2,60
Number of characters	2 171 862	475 926	752 304
Number of Arabic letters	1 095 496	238 946	399 809
Number of Tashkeels (Diacritics)	848 099	180 777	265 400

of those sets. The system did not accept numbers, symbols, or special characters, only the Arabic alphabet.

The first dataset, DS_1, was built at King Abdulaziz City for Science and Technology [28]. It included 231 files of proofread Arabic text. Each of these files contained 100 or more diacritized sentences, where each sentence had an average of 10 words, with a total number of more than 255,455 words. The dataset consisted of more than 2 million characters of Arabic letters, spaces, and diacritics. During the course of building this dataset, we focused on the diversity of the subjects covered to avoid the diacritization process being biased toward a particular subject.

The second dataset, DS_2, included the fully diacritized text from the Holy Quran, where each verse was considered as a separate sentence. In this dataset, the average number of words per sentence was higher than that in DS_1, which affected the diacritization process, as shown below. DS_2 included approximately 476,000 characters, more than 50% of which were Arabic letters, 38% were diacritics, and the remainder were spaces.

The third dataset, DS_3, consisted of 54,463 Arabic names, each of which was considered as a separate sentence that could include up to four parts: surname, father's name, grandfather's name, and family name. The total number of words in DS_3 was around 141,000, with an average of 2.6 words per sentence. This low figure, compared with the previous datasets, positively influenced the diacritization process, as will be illustrated later. DS_3 included 752,304 characters of which 400,000 were Arabic letters, more than 265,000 were diacritics, and the remainder were spaces.

B. EXPERIMENTS

We conducted several experiments with different configurations. In the following, we present the performance and characteristics of the most notable experiments. These experiments reported two values: word error rate (*WER*) and diacritization error rate (*DER*). *WER* is the percentage of words that have at least one faulty diacritic. *DER* is the percentage of diacritics that were incorrectly restored.

We performed the first set of experiments using 10,000 sentences from DS_1 as training samples. The testing samples were 30%, 60%, and 90% of the training samples. In all these experiments, the *DER* varied between 27% and 28%, as shown in Table 6. The variation among the three values is not significant. We also observed similar results when we

TABLE 6. *DER* (%) for DS_1 using different training and testing samples.

Testing sample size (%)	No. of training sentences	
	10 000	20 000
30	27,21	27,10
60	27,18	27,22
90	27,68	27,33

expanded the training samples to 20,000 sentences from DS_1. The two groups of experiments resulted in a very poor *WER* that reached 78%. This indicates that the diversity among DS_1 samples may have resulted in favoring a set of diacritics that did not correctly match the context of the sentence. For example, a set of characters may frequently occur in Arabic text with a certain diacritic sequence. If the context dictates that this character set must have a different diacritic sequence to serve the meaning, then the likelihood probability of the correct diacritic sequence is less than that of its equivalent most-frequently wrong diacritic sequence. In this case, both *DER* and *WER* increase.

TABLE 7. Comparing *DER* (%) among the three datasets.

Testing sample size (%)	Datasets		
	DS_1	DS_2	DS_3
33	26,87	29,09	19,43
67	26,96	29,07	19,93
100	27,06	29,08	19,95

We conducted a second set of experiments on the three datasets in which we used the entire dataset as training samples: 24,274 samples from DS_1, 5,308 samples from DS_2, and 54,463 samples from DS_3. For each dataset, we performed three experiments: using one third of the dataset as a testing set, using two thirds of the dataset, and finally, using the full dataset. Table 7 presents the results of these experiments. It shows consistent behavior across the three datasets using various set sizes for testing. It also illustrates a noticeable increase in *DER* for DS_3 compared with the other two datasets. This is mainly because of the dataset size available for training, which enabled the system to capture the dynamicity of the dataset. By contrast, DS_2 had the lowest *DER* because of the limited dataset size. Another reason for this difference relates to the nature of the content of each dataset. Whereas DS_1 and DS_2 included a diversity of topics, DS_3 consisted of Arabic names, which are more limited in range. DS_2 also demonstrated the lowest performance among the three datasets in terms of *WER*: more than 80%. DS_1 improved *WER* by 8% and DS_3 improved *WER* by 33%.

The third set of experiments measured system performance using testing samples that differed from the training samples. Because of the limitation in dataset sizes, we could only conduct this test on DS_1. We performed two experiments:

- In the first experiment, we trained the engine using 10,000 sentences. We then used 4,000, 8,000, and 120,000 different sentences as a testing set. *DER* was 27.07%, 27.22%, and 27.41%, respectively.

TABLE 8. Performance summary for the four systems based on HMMs.

Systems	Corpus	No. of diacritics	No. of words in the training set	No. of words in the testing set	Systems performance (%)	
					Unigram	Bigram
Gal [17]	DS_2 (Transliterated)	3	90000	9000	26	14
Elharby et al. [18]	DS_2	11	1366	1366	7.5	4.8
Elshafei et al. [20]	DS_2	8	78000	995	NA	28
Elshafei et al. [21]	DS_1 (Only 100 files)	8	29500	NA	NA	5.5

- In the second experiment, we trained the engine using 20,000 sentences. We then used 4,000 different sentences as a testing set. *DER* was 26.91%.

We observed a slight difference between these results and those listed in Table 6. Recall that the testing samples used to deduce *DER* were different from the training samples, whereas in Table 6, the testing samples used to deduce *DER* were part of the training samples. Therefore, the randomly selected new samples from DS_1 resulted in the same accuracy compared with testing the engine with part of the training samples.

C. COMPARISON WITH OTHER HMM-BASED APPROACHES

In the literature review section, we surveyed a number of diacritic restoration systems. These systems vary between morphological, syntactic, and semantic rules, neural networks, dynamic programming and HMMs. In this section, we compare the proposed system with four systems that are all based on HMMs [17], [18], [20], [21]. These systems have almost the same approach, in which they consider undiacritized Arabic text as an observation sequence for HMMs, and the hidden states are the fully diacritized word expressions. Each system extrudes the optimal state sequence that represents the fully diacritized Arabic text. During the testing phase, these systems search through all the words that have a similar non-voweled structure and select the diacritized word that has the highest likelihood.

This approach can generate diacritic marks only for those non-voweled words that already exist in the model. This approach has two drawbacks: it is lexicon-based and therefore limited from a practical point of view. Second, each non-voweled word has a number of distinct models that are equivalent to all possible diacritic combinations. The two drawbacks lead to the problem of whether, when adding a new word or new combination of diacritics the model has to undergo a complete training cycle to accommodate the modification. Moreover, sentences that contain unlisted words are excluded from the testing results. This highly contributed to the implausibly low *WER* shown in Table 8. Though Elshafei *et al*. [20] highlighted a *DER* of 5.43%, the aforementioned four systems do not calculate *DER* using the aforementioned definition because the basic unit of their models is the word rather than the letter.

Another factor that affects overall performance is the number of diacritics the system can recognize. Whereas Gal [17] used only three basic diacritics, Elshafei *et al*. [20], [21] extended the diacritic set into eight, and El-Harby *et al*. [18] used 11 diacritical marks to vowelize the Arabic text. Extending the diacritic set expands the practicality of the system however; it complicates the implementation of HMMs.

The size of the testing set reflects the actuality of system performance. Training the model with tens of thousands of diacritized words and testing it with a few hundreds of undiacritized words cannot demonstrate the robustness of the system [18], [20].

IV. CONCLUSION

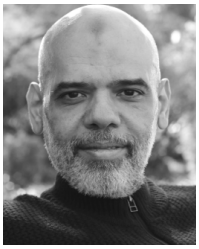
In this paper, we proposed a system to diacritize Arabic text. The system was based on HMMs. The raw text was considered as a sequence of observations, where each observation represented the ASCII code of the equivalent character of the alphabet. The observation sequence was injected into the HMM engine, which transitioned its states to deduce the optimal path across the model. The optimal output state sequence represented the set of diacritics that accompanied the Arabic raw text. The performance of the proposed system was assessed using an Arabic corpus that included three distinct datasets.

The proposed system is novel in the sense that the basic unit is the letter not the word, as in systems in the literature. This provided more flexibility for our system to learn and expand the vocabulary. We achieved a higher *DER* and *WER* than other systems that inexcusably excluded unknown words when calculating *DER* and *WER*.

REFERENCES

[1] A. M. Azmi and R. S. Almajed, "A survey of automatic Arabic diacritization techniques," *Natural Lang. Eng.*, vol. 21, no. 3, pp. 477–495, 2013.
[2] S. Ananthakrishnan, S. Narayanan, and S. Bangalore, "Automatic diacritization of arabic transcripts for automatic speech recognition," in *Proc. 4th Int. Conf. Natural Lang. Process.*, Kanpur, India, 2005, pp. 1–8.
[3] M. Elshafie, H. Almuhtaseb, and M. Alghamdi, "Techniques for high quality Arabic speech synthesis," *Inf. Sci.*, vol. 140, pp. 255–267, Feb. 2002.
[4] I. Rebai and Y. BenAyed, "Text-to-speech synthesis system with Arabic diacritic recognition system," *Comput. Speech Lang.*, vol. 34, no. 1, pp. 43–60, 2015.
[5] Y. A. Alotaibi, A. H. Meftah, and S.-A. Selouani, "Diacritization, automatic segmentation and labeling for Levantine Arabic speech," in *Proc. IEEE Digit. Signal Process. Signal Process. Educ. Meeting (DSP/SPE)*, Napa, CA, USA, Aug. 2013, pp. 7–11.
[6] A. Zouaghi, L. Merhbene, and M. Zrigui, "Combination of information retrieval methods with LESK algorithm for Arabic word sense disambiguation," *Artif. Intell. Rev.*, vol. 38, no. 4, pp. 257–269, 2011.
[7] A. Farghaly and K. Shaalan, "Arabic natural language processing: Challenges and solutions," *ACM Trans. Asian Lang. Inf. Process.*, vol. 8, no. 4, pp. 1–22, 2009.

[8] M. Maamouri and A. Bies, "The Penn Arabic treebank," in *Proc. Arabic Comput. Linguistics*, Stanford, CA, USA, 2010, pp. 1–42.
[9] A. O. Bahanshal and H. S. Al-Khalifa, "A first approach to the evaluation of Arabic diacritization systems," in *Proc. 7th Int. Conf. Digit. Inf. Manage. (ICDIM)*, Macau, China, Aug. 2012, pp. 155–158.
[10] T. El-Sadany and M. Hashish, "Semi-automatic vowelization of Arabic verbs," in *Proc. 10th Nat. Comput. Conf.*, Jeddah, Saudi Arabia, 1988, pp. 725–732.
[11] Y. A. El-Imam, "Phonetization of Arabic: rules and algorithms," *Comput. Speech Lang.*, vol. 18, no. 4, pp. 339–373, 2003.
[12] M. Attia, "A large-scale computational processor of the Arabic morphology, and applications," M.S thesis, Faculty Eng., Cairo Univ., Giza, Egypt, 2000.
[13] M. Attia, "Handling Arabic morphological and syntactic ambiguity within the LFG framework with a view to machine translation," Ph.D. dissertation, School Lang., Linguistics Culture, Univ. Manchester, Manchester, U.K., 2008.
[14] M. Badrashiny, "Automatic diacritizer for Arabic texts," M.S. thesis, Faculty Eng., Cairo Univ., Giza, Egypt, 2009.
[15] G. Abandah, A. Graves, B. Al-Shagoor, A. Arabiyat, F. Jamour, and M. Al-Taee, "Automatic diacritization of Arabic text using recurrent neural networks," in *Proc. 13th Int. Conf. Document Anal. Recognit.*, Nancy, France, vol. 18, 2015, pp. 183–197.
[16] A. Ibraheem. *A Concept Paper on a New Language Model for Automatic Diacritic Restoration*. Accessed: Dec. 6, 2016. [Online]. Available: http://ifecisrg.org/sites/default/files/articles/diacritic%20graph%20arabic20.pdf
[17] Y. Gal, "An HMM approach to vowel restoration in Arabic and Hebrew," in *Proc. ACL Workshop Comput. Approaches Semitic Lang.*, 2002, pp. 27–33.
[18] A. A. El-Harby, M. A. El-Shehawey, and R. El-Barogy, "A statistical approach for Qur'an vowel restoration," *J. Artif. Intell. Mach. Learn.*, vol. 8, no. 3, pp. 9–16, 2008.
[19] M. S. Khorsheed, "A HMM-based system to diacritize Arabic text," *J. Softw. Eng. Appl.*, vol. 5, pp. 124–127, Dec. 2012.
[20] M. Elshafei, H. Al-Muhtaseb, and M. Alghamdi, "Statistical methods for automatic diacritization of Arabic text," in *Proc. 18th Nat. Comput. Conf.*, Riyadh, Saudi Arbia, 2006, pp. 301–306.
[21] M. Elshafei, H. Al-Muhtaseb, and M. Alghamdi, "Machine generation of Arabic diacritical marks," in *Proc. Int. Conf. Mach. Learn., Models, Technol. Appl.*, Las Vegas, NV, USA, 2006, pp. 128–133.
[22] R. Nelken and S. M. Shieber, "Arabic diacritization using weighted finite-state transducers," in *Proc. ACL Workshop Comput. Approaches Semitic Lang.*, 2005, pp. 79–86.
[23] Y. Hifny, "Higher order n-gram language models for Arabic diacritics restoration," in *Proc. 12th Conf. Lang. Eng. (ESOLEC)*, Cairo, Egypt, 2012, pp. 1–5.
[24] I. Zitouni, J. Sorensen, and R. Sarikaya, "Maximum entropy based restoration of Arabic diacritics," in *Proc. 21st Int. Conf. Comput. Linguistics 44th Annu. Meeting ACL*, Sydney, NSW, Australia, 2006, pp. 577–584.
[25] I. Zitouni and R. Sarikaya, "Arabic diacritic restoration approach based on maximum entropy models," *Comput. Speech Lang.*, vol. 23, no. 3, pp. 257–276, 2009.
[26] M. Rashwan, M. Al-Badrashiny, M. Attia, S. Abdou, and A. Rafea, "A stochastic Arabic diacritizer based on a hybrid of factorized and unfactorized textual features," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 19, no. 1, pp. 166–175, Jan. 2011.
[27] G. D. Forney, Jr., "The Viterbi algorithm," *Proc. IEEE*, vol. JPROC-61, no. 3, pp. 268–278, Mar. 1973.
[28] M. Alghamdi *et al.*, "Automatic Arabic text diacritizer," King Abdulaziz City Sci. Technol., Riyadh, Saudi Arabia, Tech. Rep. CI.25.02, 2006.



MOHAMMAD S. KHORSHEED received the B.S. and M.S. degrees in computer engineering from King Saud University, Riyadh, Saudi Arabia, in 1989 and 1994, respectively, and the Ph.D. degree from the Computer Laboratory, University of Cambridge, U.K., in 2000, with a focus on recognizing cursive text in historical Arabic manuscripts.

In addition to his research activities, he led the establishment of a number of national programs in Saudi Arabia in technology incubation, university–industry research collaboration, and intellectual property management. In his secondment to General Electric, he was appointed as the Innovation Officer in Saudi Arabia. He established the GE Innovation Center, Dhahran Techno Valley, which was the first center in the Middle East, North Africa, and Turkey regions.

He has over 35 patents issued from USA, European, Japanese, Chinese, and Saudi patent offices. His research activities span image and text processing, data mining, 2-D and 3-D barcode recognition, pattern recognition, Arabic document processing, and optical character recognition.