IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received August 17, 2021, accepted October 21, 2021, date of publication October 26, 2021, date of current version October 29, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3122977

Automatic Methods and Neural Networks in Arabic Texts Diacritization: A Comprehensive Survey

MANAR M. ALMANEA
College of Languages and Translation, Al Imam Mohammad Ibn Saud Islamic University, Riyadh 11652, Saudi Arabia
e-mail: malmanea@imamu.edu.sa

ABSTRACT Arabic diacritics are signs used in Arabic orthography to represent essential morphophonological and syntactic information. It is a common practice to leave out those diacritics in written Arabic. Most Arabic electronic texts lack such diacritics. The processing of those texts for various purposes of Natural Language Processing is a complicated task. Diacritized words are necessary for applications such as machine translation, sentiment analysis, and speech synthesis. To address this problem, several studies proposed automatic systems to restore diacritics in Arabic texts. The present paper presents an in-depth survey of 56 most recent Arabic diacritization studies. Based on the diacritization approach, the studies are grouped into four sections in terms of method; rule-based, simple statistical, hybrid, and Neural Networks. While rule-based methods such as morphological analyzers and lexicon retrievals were the earliest approaches, results indicated that they are still valuable tools that can aid in the process of diacritization. Effective statistical methods that produced diacritics with acceptable accuracy include Hidden Markov Model, n-grams, and Support Vector Machines. They are often accompanied by either rule-based or neural networks in hybrid systems. Neural networks, specifically Bidirectional Long Short Term Memory, reached very high diacritization accuracy levels. Studies employing neural networks focused on evaluating and comparing the efficacy of several types of neural networks or a hybrid of them, testing alternatives of input units or suggested schemes for partial daicritization. The study synthesizes the results of the studies, identifies research gaps, and offers recommendations for future research.

INDEX TERMS Arabic text diacritization, neural networks, Arabic corpora, deep learning, Arabic natural language processing.

I. INTRODUCTION

There is a rapidly growing body of electronic texts in the Arabic language, making Arabic the 4th most used language on the internet [1]. Arabic is the first language of nearly hundreds of millions in the Arab world and the official language of over a dozen countries [2]. While the number of different Arabic corpora continues to grow, there is an increasing need for robust tools to process that data for other essential purposes [3].

However, the unique linguistic properties of the Arabic language play a significant role in complicating the development of Arabic Natural Language Processing (NLP). Arabic is special in many of its features, such as the right-to-left orientation in writing, the unique cursive, joint script, as well as the large number of important, non-alphabetic diacritic symbols. Such characteristics pose difficulties for computerized models which seek to work with Arabic texts [3].

It is estimated that up to 95% of Arabic digital texts do not include diacritics (e.g., short vowels and consonant doubling symbols) even though they are vital for understanding [2]. It is difficult for nonnative users of the language, young learners, and computerized programs to decide on the meaning of Arabic words without diacritics. Accordingly, many automatic methods to restore diacritics on Arabic electronic texts have been proposed with varying accuracy rates. Some of these methods followed traditional rule-based approaches, while others have applied statistical methods. Nonetheless, the most recent and promising systems, nevertheless, are those employing Artificial Intelligence (AI) applications, including Deep Learning (DL) and Neural Networks (NN) [4]. It has been reported the best-performing AI

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

systems depended on DL, such as speech recognition devices in smartphones or Google's automatic translator [5]. This study presents an overview of the current methods proposed to automatically restore diacritics in digital Arabic texts with a specific focus on DL. It compares their success rates in order to guide future research into the most successful methods of the automatic Arabic diacritization process.

II. SIGNIFICANCE OF THE STUDY

Concomitant with the growing body of Arabic digital texts, there is a parallel growing need arises; namely to automatically add the required diacritics to those texts. Diacritics are essential for computers to understand Arabic digital texts specifically in applications such as text-to-speech synthesis, machine learning (ML), machine translation, part-of-speech tagging, and sentiment analysis to mention but a few. Generally, restoring Arabic diacritics is expected to enhance semantic and syntactic related applications [2], [4]. Restoring diacritics is also a critical pre-step for text annotation practices such as part-of-speech tagging and tokenization [6].

In addition to AI-related fields, automatically restoring Arabic diacritics is essential in some linguistic areas, such as corpus linguistics, which depends on analyzing a massive body of electronic language texts. Adding diacritics to the texts can increase the accuracy of an Arabic concordancer, which are the search engines in language corpora [7]. Diacritics are also essential for producing reliable frequency counts [8].

To the researcher's knowledge, no earlier study has attempted to provide a survey of the NNs approaches to restore diacritics in Arabic texts. The available earlier surveys of diacritization approaches in general, namely, [9], [10], were limited to non-neural methods of diacritization, which were either rule-based or simple statistical methods. The present study aims to fill this research gap by providing an up-to-date survey of the different, recent, neural and non-neural approaches to restoring diacritics in Arabic texts.

III. STATEMENT OF THE PROBLEM

Like other languages, including diacritics besides alphabets, i.e., Vietnamese and Yoruba, Arabic electronic texts are primarily written without diacritics even though these are vital for understanding [4]. Leaving out diacritics in writing is a common practice in most Semitic languages [11]. Some researchers even consider decoding diacritics in Arabic script to be an only optional feature [12]. The linguistic explanation for that practice lies in the linguistic principle of economy, which involves mechanisms of simplification and conveying more information with less effort [13]. Skilled native speakers of the language can quickly and efficiently decipher the meaning without diacritics through cognitive top-down language processing in the brain, which depends on the mental lexicon and conceptual knowledge [14]. Nevertheless, if there is no access to that lexicon, as in the case of computers, young readers, and nonnative speakers of Arabic, diacritics are of paramount importance.

Orthographic systems of languages range from 'shallow orthography' to different degrees of 'deep orthography' [15]. In shallow orthography, there is a direct, systematic correspondence between the written symbol and the word's pronunciation. All information required for the correct pronunciation is available in the written sign. These systems are perfect for computer application and Machine Learning (ML) because it lowers the ambiguity level to a significant extent. On the other hand, in deep orthographic systems, the relationship between the symbol and the pronunciation of the word, and its meaning, is obscure. There is only a partial relationship between the symbol and the pronunciation of the word. Arabic texts which lack diacritics have a very deep orthographic system. The process of diacritization is a process of changing the orthography into a more shallow and systemic one.

In the Arabic orthographic system, all short vowels and gemination (consonant doubling) are indicated through diacritics rather than alphabets. Alphabets show consonants and long vowels only. Yet, short vowels and germination in Arabic are phonemic, i.e., they can change the word into another. If they are left out in writing, the resulting text contains a vast number of *homographs*. They are different words with different pronunciations but with the exact spelling. For instance, the word (علم) without diacritics can mean "flag," "knowledge," "taught," "knew," and "is known." The word (رجل) without diacritics can mean "foot" or "man." This problem of Arabic electronic texts is added to the usual linguistic problem of *hyponyms,* which are different words with different meanings but with the same pronunciation and diacritics. Table one represents an example of the Arabic word (سمر) and the various words it can mean if it is devoid of diacritics.

There are many similar examples in Arabic texts of symbols standing for a wide range of words due to the root -and-pattern methodology of Arabic. This creates extensive ambiguity cases for NLP applications such as machine translation and speech synthesis. Quite a large number of those ambiguity cases can be resolved if the diacritics differentiating various words are automatically and accurately produced. In addition, more accurate frequency counts in Arabic and type/token corpora analyses can be achieved if electronic Arabic texts are diacritized [7], [8].

IV. OBJECTIVE OF THE STUDY

The present study's objective is to conduct an in-depth survey of 56 recent research studies that provided Arabic automatic diacritization solutions. The study will trace the path of progress in automated Arabic diacritization systems. It will analyze, classify, and compare the results of the most recent studies in the field, specifically those published during the last decade and until the present time (mid-2021). Specifically, the study intends to clarify the different methodologies employed in Arabic diacritization systems and synthesize their results to provide a reference basis and guide for future research.

V. BACKGROUND

A. ARABIC DIACRITICS

Diacritics in Arabic are special marks or symbols that are added either above or below the letters in written orthography. They perform either phonological or syntactic functions [16]. They can be divided into two main categories:

1. The first category is termed lexical or morphological diacritics. They represent short vowels, consonant gemination, prolong vowels, and the glottal stop within the words' stems. They are essential for limiting the number of homographs in texts. Some researchers label this type of diacritics 'lexical diacritics' because they help differentiate one lexeme (word) from others [12]. They are also called morphological diacritics in the literature due to their origin in the morpheme pattern in Arabic.

TABLE 1. Different words represented by the same symbol (سمر) if written without diacritics.

Written word without diacritics	meaning	Written word with diacritics
سمر	a female name	سَمَر
سمر	a type of wood	سَمْر
سمر	night gathering and talk	سَمَر
سمر	darkened	سَمُر
سمر	nailed	سَمَّر

2. Syntactic diacritics, on the other hand, are linguistically designated as final letter 'case- marking'. They are short vowels and nunnation (a final-n) assigned to the last letter of the word to indicate its syntactic function in the sentence. Nunnation is an indefiniteness marker. This class is sometimes called inflectional diacritics [12]. Syntactic diacritics, while still an essential part of Arabic orthography, are allophonic rather than phonemic. Unlike phonological diacritics, which can alter the word's meaning, syntactic diacritics do not change the meaning. Instead, they change the role of the word in the sentence as a subject, object, etc. They are needed for the correct reading of Modern Standard Arabic (MSA) in official and educational settings.

Table two represents Arabic alphabets followed by Arabic diacritic symbols. Arabic diacritics include short vowel signs [a,u,i] and the nunnation and gemination symbol. I considered the Hamza (glottal stop sign) (e.g., سأل/سال) and Madd (prolonged vowel) (e.g., مآل/مال) as diacritic symbols because they are phonemic in Arabic and are often left out especially in informal writing.

It is generally not feasible to manually diacritize Arabic digital texts because it will be time-consuming and costly, especially in a massive text corpus [2]. In addition to that, accuracy is not guaranteed in manually diacritized texts, particularly in syntactic diacritics, i.e., case-endings. Even native speakers are not always accurate in choosing the correct diacritic to mark the end of Arabic words within sentences. Proper case-ending diacritic restoration requires linguistic experts. As an early step to dealing with Arabic electronic texts, researchers tried to make Arabic texts ready for computer use by transliterating Arabic words into Roman alphabets. As [3] mentioned, the Buckwalter system converts Arabic to an equivalent based on the Roman alphabet. Still, this approach was not successful because many aspects of Arabic, such as text orientation were altered. Due to such limitations, automated methods of diacritization of Arabic texts were necessary. One of the most promising methods of automatic systems is the use of NN and DL to perform the required diacritization process. DL is the basis of the best-performing systems in almost every AI research area [5]. The researcher in [4] argues that the use of DL and NN is superior to other publicly available competitors. This performance places it as the benchmark system which future works in the field needs to equal or surpass.

TABLE 2. Arabic alphabets and diacritic symbols.

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

Diacritic Name	Shape on letter	IPA representation
Fathah	ـَ	/a/
Dammah	ـُ	/u/
Kasrah	ـِ	/i/
Fathatan (nunnation)	ـً	/an/
Dammatan (nunnation)	ـٌ	/un/
Kastratan (nunnation)	ـٍ	/in/
Sukun	ـْ	None
Shaddah (gemination)	ـّ	consonant doubling
Hamza	ء	Glottal stop /ʔ/
Madd (prolonged vowel)	آ	A prolonged vowel

B. OVERVIEW OF NEURAL NETWORKS

NNs are mathematical sets of algorithms aiming to mimic the human brain functions, specifically its ability to trace relationships. When the human brain is provided with an input that includes a relationship, thinking moves from one neuron to another in a network of neurons to uncover details of that relationship. A computerized NN is modeled on the structure of the brain cells' networks (Fig. 1). It consists of a neural net that contains a vast number of nodes in different layers that are densely interconnected. The computer learns to perform some tasks through these networks by exposure to a set of trained examples. The examples are frequently hand-labeled in advance. A Deep Neural Network (DNN) includes an input layer, an out-put layer, and several layers in between. When a data item is received at the input layer, it is transferred through nodes that assign it specific weights and pass it to other layers until it reaches the output layer with

a particular value of the calculated weights [5], [18]. NNs do not require explicit rules to generate the required output. They only need exposure to sufficient training data and computing power. Their results are highly accurate.

There are various types of neural networks with different architecture, each with its features which suit specific tasks. The field of NLP mainly depends on either the feedforward DNN or the Recurrent Neural Network RNN. Feedforward DNN is designed as a one-way process with the inputs fed into the network through the first layer; then, the output is provided as input for the following layer. At the same time, the whole process is governed by supervised ML, and the final result could be a classification or regression [17]. Fig. 2 is a diagram of FeedForward DNN.

RNN has a unique feature: the feedback or return loop of the output to be input again. Fig. 3 below is a diagram of RNN:

This type of network has internal memory, and that is why it can be divided into two types: Long Short Term Memory (LSTM) and Bidirectional-LSTM (BiLSTM). The LSTM network comprises four components: the input gate, output gate, forget gate, and a cell. The cell contains the memory that characterizes the network and stores the values for a specific time. The other components (gates) control the flow of data to and from the cell. The main advantage of this type is the ability to analyze data composed of long sequences such as speech or text analysis [19], [20].

BiLSTM is similar to the LSTM because both are RNN with memory. It is bidirectional because the analysis of the sequence goes in both directions, either forward or backward. At the same time, both directions are connected to the same output layer. When a section or point analysis in a given sequence is completed, the surrounding points are also conducted before or after [19], [20].

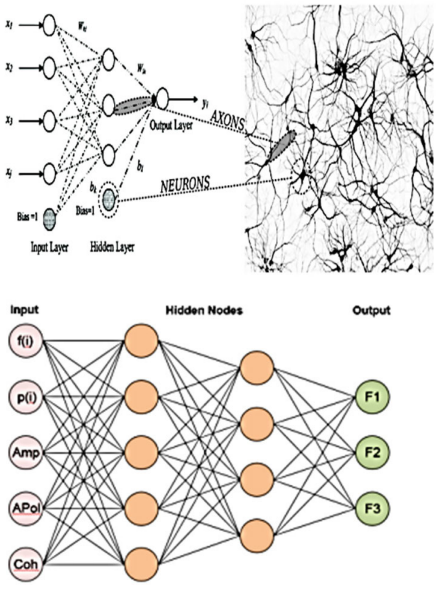

FIGURE 1. The similarities in structures of brain neural cells and neural networks Adapted with permission from *GeoTerra Technologies*: https://www.geoterratech.com/interpretation-services/neural-networks-and-machine-learning/.

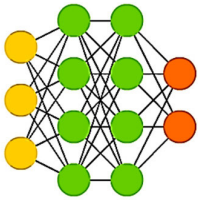

FIGURE 2. FeedForward DNN (adapted with permission from [18]).

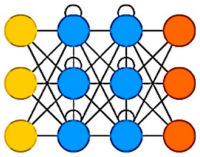

FIGURE 3. RNN (adapted with permission from [18]).

VI. METHODOLOGY

The present study reviews, classifies, and synthesizes the most recent studies in Arabic automatic diacritization. The selection criteria are based on the recency of publication and the linguistic variety of Arabic texts. This study reviews a pool of 55 studies published mainly during the last decade since 2010 and up to the year 2021. A few studies before 2010 were included in the review due to their valuable contribution to the field. The review is limited to studies investigating automatic diacritization in MSA and Classical Arabic (CA). Studies conducted on dialectal Arabic were excluded from this review because of the significant differences between dialectal Arabic and MSA or CA. To collect studies for this review, a search was done in the following databases: IEEE explorer, Clarivate Analytics, Google Scholar, and Science Direct. The resultant group of studies was categorized based on the diacritization method, detailed in the next section.

As the survey of studies will show, some of the proposed systems were evaluated empirically while others were presented and explained but not evaluated. The evaluation method varied from one study to another. Several studies provided the accuracy percentage. In contrast, most studies depended on the Word Error Rate (WER) calculation, which means the rate of words that have an error in diacritics and Diacritic Error Rate (DER), which refers to the rate of errors in all diacritics of letters within words. However, some studies include case-endings within this calculation, while other studies exclude case-endings. A further complication is the different corpora and databases used for training and evaluation purposes. They differ in type and size. All the above renders the process of comparison between the published results of systems challenging. Consequently, the results of each study are mentioned within the description of the research. Still,

a comparison between rates of accuracy has not been carried out so as to avoid false conclusions.

VII. ANALYSIS

After analyzing the diacritization studies included in this review, it was apparent that they can be classified into three categories as follows as illustrated in Fig. 4:

1) The first and earliest approach is the state-of-the-art, base-line methods which are rule-based. These studies developed programs fed with all the required knowledge and morphological/syntactic rules to produce Arabic diacritics adequately. The linguistic rules are formulated in most cases by human experts. They provided encouraging results with acceptable accuracy. At the same time, they introduced some errors and complications. The major drawback of rule-based models is the tedious, costly, and time-consuming task of formulating and maintaining rules that cover the rich linguistic aspects of Arabic. Additionally, developing these systems requires continuous overview and updates by linguistic experts [11]. Even though these traditional approaches were the earliest, this approach did not recede and fade away. It proved its efficacy in hybrid architectures with other ML systems.

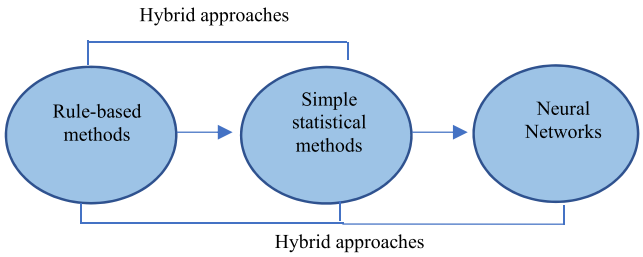

FIGURE 4. The development of diacritization systems in the literature.

While statistical methods automatically restored a considerable part of diacritics without the need for rules, as the review will show, their accuracy was not very high to make statistical methods sufficient to restore all diacritics. 2) The second category employed simple statistical methods to restore diacritics. These methods do not use explicit linguistic rules. Instead, statistical models learn diacritization from diacritized texts; by predicting the probability of distribution of a sequence of words or characters. Therefore, they are frequently aided with rule-based methods in different hybrid approaches. This approach performed the task and yielded very accurate results.

3) The third category falls under various types of DL. The latest trend in the field is using different kinds of NNs in DL approaches to perform the required diacritization process. It attracts the attention of many researchers lately. Fig. 5 shows the distribution of the studies reviewed in this article. The highest number of recent studies used NNs to restore diacritics in Arabic texts. This reflects the recent interest in this approach by researchers in the last few years. Indeed, the DL approach reached high accuracy rates and paved the way for more improvement in the future. It is argued that the use of DL and NN is superior to other publicly available competitors [4]. This fact places it as the benchmark system which future works in the field needs to improve upon and surpass. Within this group of studies, some researchers combined the merits of rule-based or statistical methods and DL. As discussed later, mixing both systems into a more extensive hybrid system with different subcomponents yielded significant findings.

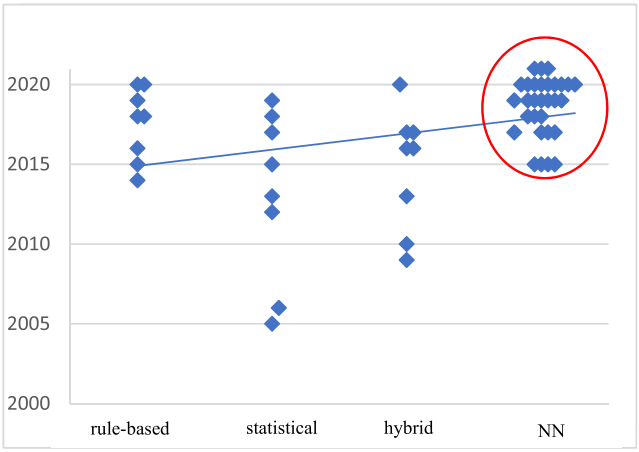

FIGURE 5. Distribution of reviewed studies across different diacritization approaches.

The following sections will review studies in two large areas, namely non-neural approaches and neural approaches. Within each section, studies are classified according to the methodology used, and the specific problem addressed as follows:

A. Non-neural approaches to Arabic diacritization:

1. Rule-based methods
2. Statistical methods
3. Hybrid approaches

B. Neural Networks approaches to Arabic diacritization:

1. Evaluating the efficacy of different NNs
2. Hybrid neural approaches
3. NN and diacritization for speech synthesis
4. Input unit
5. Partial diacritization

The review discusses different studies in each section. Summaries of the specific details of studies, including the date of publication, method of diacritization, corpora, evaluation measure, and results, are presented in Tables 3-6 in Appendix I.

A. NON-NEURAL APPROACHES

Non-neural approaches to restore diacritics consist of either rule-based methods or statistical methods. It is often hard to draw a clear boundary between these two groups because most studies depended on a hybrid of both approaches, i.e., linguistic rules in addition to statistical methods. Some rule-based systems were later developed in more recent versions with a simple statistical program to boost their performance. The opposite is true, as well. Many initially statistical

systems were aided by employment of some rule-based models to enhance their accuracy, as discussed below.

1) RULE-BASED METHODS

The earliest approach to automatically restore diacritics in Arabic texts was completely rule-based. It relied either on human input to set the linguistic rules required for diacritization or used digital dictionaries for that purpose. Most of the online published tools for Arabic diacritizition fall under this category. They usually employ morphological analyzers to restore diacritics. Other methods used in this type of studies are Weighted Finite-State Transducers, lexicon-retrieval and diacritic borrowings from other texts [2].

The majority of rule-based tools are not specifically designed to restore diacritics. Rather, they serve several Arabic language-related functions, including supplying diacritics. The study in [2] reviewed a list of the open-source, rule-based diacritization systems available online. One of them is the often-cited tool, namely, the *MADAMIRA* analyzer developed by Pasha *et al.* [26]. It provides diacritization, tokenization, part-of-speech tagging, and other Arabic language processing tools using morphological analysis. *MADAMIRA* is a short form of Morphological Analysis Disambiguation for Arabic (*MADA*) [25]–[27], while the rest of the name refers to some additional features inspired by the *AMIRA* toolkit [26]. It is often used in diacritic studies as a benchmark to compare its accuracy with other proposed systems. *MADAMIRA* has some issues regarding the accuracy of diacritization. Research showed that *MADAMIRA* system (which also used a statistical system, namely, Support Vector Machine (SVM)) was significantly lower in terms of accuracy than systems that used NNs (specifically BiLSTM) [28], [29]. It also suffers from a low processing speed [29].

Ali-Soft is an online diacritization tool that provides simple diacritization and text-to-speech synthesis. It has additional features to style the text while diacritizing it. However, the system does not diacritize each letter in each word. For the most part, it does not diacritize the last letters. It has many diacritic errors as well. It duplicates some letters and some words. Similarly, *Farasa* [30] provides some Arabic NLP functions, one of which is partial diacritization. It used linear kernels based upon SVM and was trained using the Penn Arabic Treebank. Yet, like Ali-Soft, the diacritization system of *Farasa* has many issues. It suffers from low accuracy [31]. Indeed, *Farasa's* diacritic restoration performance was compared to some examples of DNN performance [17]. The results show that the DNN system outperforms *Farsa* with an accuracy rate of 4.2% higher than its accuracy rate.

Harakat is another open-source system that provides diacritization services. It permits restoring diacritics in short texts, which are limited to only about 650 symbols. Like *Ali-Soft* and *Farasa*, the Harakat system has various issues, such as supplying the wrong diacritics and erroneously modifying input words. Finally, *Mishkal* is an open-source rule-based Arabic text diacritization system with issues regarding its accuracy and changing input words (dropping letters from words). Some studies found that it scores lower than its competitors in terms of accuracy [9], [12], [33].

To compare the performance of the tools mentioned above, researchers in [1] conducted a study that compared DER and WER in some of these systems. The analysis discussed whether to take the last letter diacritic into account or not because, in the researchers' view, diacritizing the last letter is a more complex problem compared to diacritizing other letters. The dataset for comparison was taken from the *Tashkeelah* corpus. The study compared them to a neural-based diacritization system *Shakkala* [2], [34]–[36], an open-source system developed by [38]. The system is a character-level deep learning system comprised of three bidirectional Long Short Term Memory networks and dense layers. It was trained on the *Tashkeela* [39] corpus. The results show that the neural network *Shakkala* significantly outperforms traditional rule-based approaches with DER and WER of 2.88% and 6.37%, respectively. The researchers compared these results with the lowest DER and WER values for the non-neural system (*Mishkal*); 13.78% and 21.92%, respectively. The results revealed that *Mishkal* and *Harakat* are the best systems among the non-neural ones, but their performance is not close to *Shakkala*. An advantage of *Shakkala* is its remarkable ability to handle the case of diacritizing the last letter of each word (case-marking). However, *Shakkala* has a limit for the number of symbols (490 symbols). This makes it of little benefit when considering longer texts.

The study in [37] developed a system for Arabic diacritization, namely *Alseraj*, which is entirely rule-based. The system depends on three modules, which are morphological, syntactic, and morph-phonological. Error rates in diacritics (8.68%) and words (18.63%) were significantly below three benchmarking statistical systems.

Following another approach, the study in [40] proposed a system that can accelerate the process of creating additional diacritized datasets in a highly cited Arabic text, namely, *Riyadh As-Salheen*. The system worked by copying or matching full and *n*-gram units of the target sentence with some manually diacritized datasets from different sources. An *n*-gram unit is a sequence of n adjacent units (e.g., letters or words). After borrowing diacritization, the percentage of diacriticized words in the text jumped from 48.66% to 76.41%. The system produced very low DER (0.004), compared to errors produced by MADAMIRA and *Farasa*. The system has reduced word ambiguity from 4.83 diacritized form/word to 1.91. However, the system reuses manual diacritization from other works. This resource is not large enough to restore diacritization of large corpora of texts.

A more recent study [41] is similarly rule-based. Against the current interest in NNs methods for diacritization, the researchers argued that NNs are not as efficient as rule-based systems since they lack comprehensive linguistic knowledge and they generate invalid words phonologically and typologically. The study presented a system named *Arabic-Unitex*, an Arabic Language Resource (a lexicon), emphasizing vowel representation and diacritization. The use

of a lexicon is among the methods utelized to restore diacritics in Arabic texts automatically. A lexicon serves as a resource to replace an undiacritized word with the diacritized version available in the lexicon. The lexicon proposed in the study contains a total of 24 rules which provided a detailed description of vowel omission in Arabic texts. The rules are integrated into large-coverage resources for morphological annotation. For restoring vowels, *Arabic-Unitex* can identify words in which the diacritics are not shown and words in which the diacritics are partially or fully included. By taking into account these rules, the system computes and restores for each word the diacritized version from a list of compatible fully diacritized candidates through omission-tolerant dictionary lookup. The advantage of this approach lies in the reversal of the traditional root-and pattern Semitic methodology into pattern-and-root, giving precedence to patterns then roots. The lexicon was built and updated manually. It contains 76,000 fully vowelized lemmas. Then, it was inflected using finite-state transducers (FSTs), which generate 6 million forms. The coverage of the inflected forms is extended by formalized grammars, which accurately detail inflections around verbs, nouns, adjectives, or prepositions. The lexicon works at a remarkable speed. It is reported that a computer requires only one minute to generate the whole 6 million inflected forms in a 340-MB flat file. This is compressed further in 2 min, into 11 MB for fast retrieval. In other words, the system analyzes 5000 words/second for running text (20 pages/second). Based on the rich linguistic resources, the lexicon also includes a spell checker. It detects any invalid/misplaced vowel in a fully or partially vowelized form. It is stated that the resources provide a lexical coverage of more than 99 % of the words used in newspapers and restore diacritics in words (out of context) simply and efficiently.

CAMeL tool is a recent, open-source, rule-based and neural tool presented in [42]. It is a Python multitask toolkit that supports Arabic and Arabic dialect pre-processing, morphological modeling, dialect identification, named entity recognition and sentiment analysis. It provides command-line interfaces and application programming interfaces covering these utilities. CAMel was developed to address problems in previous rule-based systems, such as the fragmented tasks in different systems and the lack of flexibility. It is implemented in Python 3. The CAMeL tool provides services for text preprocessing, morphological analysis, and disambiguation, among others. Preprocessing includes transliteration services in addition to orthographic normalization and de-diacritization (removal of unnecessary diacritics, which are considered noise). Tasks such as diacritization, POS tagging, tokenization, and lemmatization are examples of morphological disambiguation. It is worth mentioning that CAMel is not totally rule-based. It includes two built-in disambiguators: a simple disambiguator based on a maximum likelihood estimation model and a neural network disambiguator that provides a highly improved disambiguation accuracy using multitask learning. The accuracy of the system is tested and compared to some state-of-the-art models. The results are promising. More CAMel tasks are still under development, such as a spell checker, dependency parser, and transliterator to and from Arabic script and Arabizi (spoken Arabic written with Roman letters).

The researcher in [43] presented a rule-based model focusing on restoring case-ending diacritics in Arabic texts. Diacritization was achieved through morphological analysis, associating the word and its POS, and syntactic analysis, which considers the position of the word in the sentence. The used corpus comprises three datasets from the International Corpus of Arabic, which contains 500,000 words and is annotated morphologically. The corpus helped, based on its training set, to extract the required linguistic rules. The two other datasets were used for validation and testing. The internal (morphological) diacritics were are firstly restored. To deduce case-ending diacritics, syntactic analysis was used. The extracted rules were: stem pattern rules, transitivity rules, and definiteness rules. After that, rules for extracting case-endings of imperfect verbs, noun phrases, nouns, proper nouns, adjectives, and adverbs were applied. The researchers used the LDC's Arabic Treebank to test the system, and the WER in case-endings was about 9.97%.

Table three in the appendix provides a summary of studies reviewed in this section.

2) SIMPLE STATISTICAL METHODS

Simple statistical methods to diacritize Arabic texts provided efficient results while requiring fewer efforts. In the main, they include traditional machine-learning (ML) techniques. Some studies used Memory-based learning, Hidden Markov Models, statistical language modeling (n-gram tagger), Maximum Entropy Models, SVM, Dynamic Programming, and Conditional Random Fields. Many of these techniques are supported with rule-based methods to increase the accuracy of diacritization.

Researchers in [44] followed a statistical approach using the Hidden Markov model (HMM) to find the best diacritization format for words. A Markov chain is functional when there is a need to compute a probability for a sequence of observable events. The hidden states are the diacritized word, and the observations are the undiacritized words. The transition and emission probabilities are estimated from an extensive training corpus. The most likely diacritized word sequence is selected using the Viterbi algorithm based on words' n-grams. The most common decoding algorithm for HMMs is the Viterbi algorithm. It is one type of dynamic programming algorithm that makes use of a dynamic programming trellis. Researchers used the Holy Quran as a dataset. The error rate was nearly 4.1% which was further improved to 2.5% after applying a word-preprocessing stage and applying trigram for some selected words. The researchers suggest eliminating the rest of the errors by using a syntactic analyzer and other knowledge-based, natural language processing tools.

Another diacritic restoration system using the Viterbi decoder along with SVM was used in [45]. The Viterbi decoder is implemented for word-level diacritization with backoffs to stems and morphological patterns. For case-endings, it uses SVM coupled with filtering heuristics. The diacritizer's accuracy has achieved a WER of 12.76% and DER of 3.54% on an entirely new testing dataset. They are higher than state-of-the-art accuracy rates such as MADAMIRA, [46], [47].

The study in [48] proposed a statistical method for providing Arabic diacritics with the aid of some rule-based tools. It depends on the maximum entropy framework, with several sources of linguistic information such as lexical, segment-based, and part-of-speech tag features. The model implicitly learns the correlation between these sources of information and provides diacritics as output. Using the Penn LDC's Arabic Treebank corpus, the model achieved a DER of 5.1% and a word error rate of 17.3% if case-ending diacritics are considered. If case-ending is not considered, the DER becomes as low as 2.2%, and WER becomes 7.2%.

The researcher in [50] followed a statistical approach to restore Arabic diacritics. The study depended only on an Arabic corpus annotated with diacritics (*Tashkeela*). The researcher presented an algorithm for Arabic diacritization using the dynamic programming approach. The possible word sequences with diacritics are assigned scores using the statistical n-gram language modeling approach. If case-ending is ignored, the preliminary results show that the algorithm can lead to good results. The researcher suggested that results could be improved if complex text normalization is used for the public domain and if large-scale data processing is involved.

Furthermore, in [49], the researchers proposed a way to solve sparsity in Arabic. Since Arabic is a rich language morphologically, the number of out-of-vocabulary (OOV) words are many. To reduce their number, the researcher suggested a smoothing methodology by taking some probability mass from the observed n-gram and distributing it to the unseen n-grams. The methodology dramatically improved the accuracy of the model.

Similarly, the study [31] proposed a novel approach to support open vocabulary diacritics restoration based on the Byte Pair Encoding (BPE) method. The BPE method segments the words into variable-length sub-word units and allows open vocabulary from a fixed sub-word units dictionary. The proposed system is evaluated using the *Tashkeela* diacritization task. The study's results show that the system outperforms the character-based methods commonly used to support open vocabulary diacritics restoration within the language model scoring framework. However, syntactic diacritics accuracy remains low in the proposed systems. The researcher suggests that future work may combine DL methods with the language model scoring framework to overcome this issue.

An earlier study [11] attempted to combine statistical methods with simple rule-based models to restore diacritics in Arabic. Specifically, the study used a cascade of probabilistic finite-state transducters trained on the Arabic treebank, integrating a word-based language model, a letter-based language model, and a straightforward morphological model. A trigram language model was used for words in the study. The model was built upon using compact transducers to reach higher levels of accuracy. Except for sporadic errors or some undiacritized letters, the model achieved a word accuracy of 7.33% without including case-ending and 23.61% when the case-ending is included.

The study in [51] proposed a system that can restore diacritics by injecting the undiacritized text into an engine based on a single HMM. The Arabic script is injected into the diacritization system as a sequence of ASCII codes of the text characters. This sequence represented the observation set for the diacritization system that aimed to restore the full diacritics of the given script. The engine generates an optimal path through its states. It consisted of a 15-state ergodic HMM. At the final stage, the system matches the state sequence with its equivalent diacritics and sets them within the text. Using the software program based on the HMM engine, the user may either type an Arabic phrase for the system to diacritize or import an Arabic text file into the graphical user interface window. The program is composed of three modules. The first module reads the text and transfers it into a sequence of predefined observation symbols. The second module provides the output sequence, namely, the diacritics. The final module arranges the output sequence with the raw text by imposing the diacritics above and below the equivalent characters. Several experiments were conducted on three corpora datasets. Experiential results on different data sets demonstrated the robustness of the proposed system, even with samples that are new to the system. The system depends on the letter rather than the word as the basic unit of analysis. According to the researcher, this feature provided more flexibility for the system to learn and expand the vocabulary.

In [52], researchers presented an automatic diacritization system of Arabic using HMMs with Viterbi's algorithm. The corpus, *Tashkeela*, comprised mainly was religious texts. The results were considered satisfactory, achieving a precision of up to 80% at the word level and 90% at the character level. When the results were compared to *Mishkal* system, they show the superiority of the proposed system.

Table four in the Appendix provides a summary of studies reviewed in this section.

3) THE HYBRID STATISTICAL AND RULE-BASED METHOD

A considerable number of studies follows this approach. That is because the diacritization problem in Arabic can be divided into several sub-problems, each dealt with using a specific method. Hybrid methods are a combination of rule-based methods and statistical methods in the same system [53]. They include hybridizing linguistic rules and dictionary retrievals with morphological analysis, N-grams, HMM, Dynamic Programming, and ML methods. Some DL models were also improved by rules or statistical methods as well.

The study in [54] used an Arabic lexicon and a large corpus of fully diacritized texts for training purposes. Case-ending is achieved as a separate post-processing task implementing syntactic information. The hybrid approach relies on lexicon retrieval, bigram, SVM, and prioritized statistical techniques. If the word has more than one version, different methodologies were suggested to choose the correct version. Information sources about part of speech, word segmentation, and a bi-gram dictionary were consulted to resolve that ambiguity. The researchers concluded that statistical methods show great promise in resolving the ambiguity problem in Arabic diacritization. However, the method of combining linguistic rules and lexicons with simple statistical-based approaches has its limitations. It is hard to apply it to new domains because of the definite possibility of generating incorrect diacritics. In addition, languages develop over time, and new vocabulary items are continually added to the language, which requires a continual update of the system.

Researchers in [55] built a hybrid Arabic diacritization system that factorizes Arabic input text into all the possible morphemes and case-ending diacritics (morphological and syntactic analysis), then statistically disambiguates the most likely sequence of these entities through an **A**$*$ deep lattice search. The system proposed is dual-mode. The first mode determines the most likely diacritics by choosing the sequence of full-form Arabic word diacritization with maximum marginal probability through lattice search and long-horizon n-grams probability estimation. The second mode factorizes each Arabic word into all its potential morphological parts. Then, it also uses the same techniques used by the first mode to obtain the most likely sequence of morphemes. This hybrid system benefits from the advantages of both modes and with a performance that is superior to the best performing reported systems at that time, namely Habash and Rambow, and of Zitouni *et al.* WER without case-ending was as low as 3.1%, while it is 12.5% if case marking is included.

In [56], a hybrid diacritization system is introduced, combining both rule-based and data-driven techniques. The system resembles a pipeline of components. It relies on automatic correction, morphological analysis, POS tagging, and OOV diacritization components. Diacritization is assigned through three main phases: preprocessing; second, the generation of valid morphological analyses for each word, including analyses of both statistical and rule-based morphological analyzers to build a lattice of analyses; and third, disambiguation to generate the diacritized text including case -ending. The statistical model relies on HMM and Viterbi algorithms. Results show that errors in words and diacritics are significantly lower when all the layers are working together than if each layer is considered alone. In addition, the hybrid system was more accurate than the best-reported systems in terms of full diacritization and had comparable accuracy on the level of morphological diacritization. The researchers expect further work on POS tagging and disambiguation techniques such as word sense disambiguation could improve the system.

The study in [57] proposed a multi-lexical levels statistical and rule-based approach to estimate missing diacritics in a given undiacritized Arabic text. The system is based on a hybrid technique that combines statistical n-grams and a morphological analyzer to achieve higher accuracy of diacritization than the state-of-the-art systems. The proposed approach operates on three different contextual lexical levels. The linguistic levels are word-level, morphemes-level, and letter-level. Several configurations were also tested. The best one was the diacritization through the word-level, morpheme-level, and letter-level, with consideration of sub-models for each one. The best reported results were a WER of 7.1% and a DER of 3.9%. If case-ending is ignored, the system resulted in a WER and DER of 5.1% and 2.7%, respectively.

In [58], researchers followed a hybrid approach to restore diacritics. Their proposed system diacritizes input of an Arabic sequence of words both morphologically and syntactically. It is a hybrid of the machine translation approach, morphological analysis, and sequence labeling approaches. The system is divided into three layers: the first layer uses HMM for the morphological diacritization of previously seen words, the second layer uses an external morphological analyzer for the morphological diacritization of OOV words, and the third layer uses CRF for the syntactic diacritization of all words. To evaluate the proposed system, the researchers used the benchmark LDC Arabic Treebank Part 3 datasets utilized by the state-of-the-art systems. The proposed system achieved a morphological WER of 4.3% and a syntactic WER of 9.4%. These results are, in general, comparable to other systems. One of the advantages of the proposed approach is that its operation depends mainly on ML techniques, which makes it applicable to any language that uses diacritics with almost no required changes. Another advantage is the modularity that is achieved through layering. The diacritization problem is divided into three sub-problems, each of which is handled in a separate layer. Accordingly, the approach of handling each sub-problem can be modified or even changed easily without affecting other techniques to control the other sub-problems. To develop the accuracy of the system, the researchers suggest handling OOV differently. For the syntactic diacritics, it is recommended to use knowledge-based approaches which contain syntactic rules to reduce errors in case-endings.

The researchers in [59] presented a hybrid system for automatic diacritization of Arabic sentences combining linguistic rules and statistical treatments. The approach is divided into four stages. The first phase consisted of a morphological analysis using the morphological analyzer *Alkhalil Morpho Sys2*. It is an open-source morphological analyzer that provides various linguistic details for individual words such as POS tag, vowelization, and the corresponding patterns of a stem with their vowelization. Outputs are then used in the second phase to eliminate invalid word transitions according to the syntactic rules. The model used in the third stage is a discrete HMM and Viterbi algorithm to determine the most probable diacritized sentence. The transitions in the

training corpus are processed using smoothing techniques. The last step treats the words not analyzed by *Alkhalil analyzer2*. Statistical treatments were used based on the characters. Thirty-six syntactic rules and three diacritics rules were implemented. The integration of syntactic rules has contributed to improving the error rate WER1, and they notably allowed for correcting some mistakes in the last character. It was evident that the integration of diacritic rules has significantly improved the accuracy of the system. WER1 decreased from 8.29% for the system that does not incorporate the diacritic rules, to 6.50% for those that incorporate these rules. Similarly, WER2 decreased from 4.10% for the first system to only 2.58% for the second. The word error rate of the presented system is around 2.58% if case-ending diacritics are ignored, and around 6.28% when these diacritics are taken into account. The researchers expect that the enrichment of the training and testing corpora and the incorporation of more syntactic rules may improve the accuracy even more.

The system proposed in [60], which was rule-based, has been developed in a later study [61] to restore all diacritics rather than only case-endings. It is called SHAKKIL [61]. This is a two-layer system. The first layer is morphological (which detects internal diacritics), and the second layer provides case- ending diacritics. The first layer contains multiple sub-layers. The first sub-layer is for the uni-morphological form: it matches words with one diacritized form or POS-tag with their analysis. The second sub-layer is a rule-based morphological disambiguation layer. It excludes the wrong solutions to facilitate selecting the best POS or morphological analysis of non-disambiguated words. The third sub-layer is a statistical-based morphological disambiguation layer. The fourth sub-layer deals with the out of vocabulary words. The syntactic layer is based on rules related to POS, definiteness, stem patterns, and transitivity. The study's results revealed that WER is 4.81% and DER is 1.93% when case-ending is ignored, and WER 14.78% and DER 4.11% when case-ending is considered.

Table five summarizes the studies mentioned in this section.

B. DEEP LEARNING AND NEURAL NETWORKS APPROACHES

DL is a rich family of methods, which includes Convolution Neural Network (CNN), Recurrent Neural Network (RNN), Auto-Encoders (AE), Long Short-Term Memory (LSTM), and Generative Adversarial Nets (GAN). It is the latest trend in diacritization studies as it is extensively used in other modern NLP studies [61]. DL approaches are remarkable because it is sufficient for them to be exposed to diacritized training data before it generates diacritics for undiacritized texts. They do not require linguistic resources.

Studies in this section are categorized depending on the specific research problem they address. The first and largest group investigated the efficacy of different NNs in restoring diacritics in Arabic. While some studies focused on testing the use of one type of NNs, other studies (second group of studies) proposed hybrid systems combining different NN architectures and statistical or rule-based methods. A group of studies compared the effect of varying input units from words to characters or sub-words on the accuracy of their systems (third group). The purpose of the diacritization of Arabic texts also varied. There are systems developed for speech synthesis (fourth group of studies) and others for homograph disambiguation. Homograph disambiguation entails the introduction of partial discretization only of ambiguous words in Arabic texts. These are covered in the last group of the reviewed studies.

1) EVALUATING THE EFFICACY OF DIFFERENT TYPES OF NEURAL NETWORKS

Among the earliest studies evaluating the efficacy of DL in diacritization is the language-agnostic study presented in [46]. It tested the possibility of using NN for diacritization without the use of any rule-based resources. The authors used the Arabic Treebank dataset with no use of any other supporting tools or rule-based methods. They followed a language-independent approach in which the system is trained solely from diacritized texts without relying on external tools. The system depended on a RNN, specifically a LSTM network. Interestingly, the performance of the model effectively matched state-of-the-art solutions available at that time. The study found that LSTM models perform much better than simple feed-forward networks. The bidirectional LSTM works better than a unidirectional one. Deeper models achieve the best results. The system also offers the ability for further development to be used in speech recognition. According to [45], it performed somewhat lower than available tools, but without using any linguistic methods and with complete reliance on RNN.

The BiLSTM network is used in the system proposed by the researcher in [34], where a RNN is utilized for automatic diacritization of Arabic text. Researchers intended to develop a fast and accurate model that uses RNNs to diacritize raw texts in MSA and Classical Arabic. To train and test their model, they used two datasets, namely from LCD ATB3 and *Tashkeela*. Throughout their experiments, they analyzed the effect of changing several network parameters, such as the number of network layers and dropout, on the accuracy and execution time of the tested models. They tested three network architectures: unidirectional long short-term memory (LSTM), BiLSTM, and encoder/decoder LSTM networks. The researchers recommend wrapping very long sequences to segments not longer than 400 letters based on the results. The results also showed the superiority of the bidirectional LSTM network over the encoder/decoder network and the unidirectional LSTM. The best accuracy results were reported for a bidirectional RNN LSTM model with four layers that uses dropout. The best result of DER was 2.46% and 1.97% for ATB3 and *Tashkeela*, respectively. This DER for *Tashkeela* provides an improvement of 47% over the best published results. The researchers suggested that accuracy could be improved if a larger dataset is used. There is also a need to

search for ways to solve the missing diacritics in some parts of the output.

Another model depending on stacked B-LSTM [33] adopted texts from the Arabic Treebank similar to [46]. Additionally, it also used the Holy Qur'an, and the results were able to match those of the available state-of-the-art solutions also using preprocessing and error correction methods. No lexical, morphological, or syntactical analysis was performed on the data before being processed by the net. However, when the network is post-processed with the error correction techniques, it reached state-of-the-art performance, with an average of DER and WER of 2.09 and 5.82 %, respectively, based on samples from 11 books. Regarding the LDC ATB3 benchmark, this model reduced the DER by 25%, the WER by 20 %, and the last letter diacritization error rate by 33% over the best-published results. Researchers argue that this system outperforms the best hybrid approaches. They think that implementing morphological analysis as a preprocessing stage could provide higher accuracy.

The researcher in [12] proposed a hybrid system for Arabic diacritization that depends on a deep LSTM and a morpho-syntactic analyzer, namely, *Alkhalil Morpho Sys2* [61]. As the researchers put it, the model processes the undiacritized text separately in both systems. *Alkhalil Morpho Sys 2* outputs the diacritized forms and morpho-syntactic features. The BiLSTM network provides the diacritized text. The undiacritized words from the latter's output are selected with their contexts. The corresponding diacritized forms generated by *Alkhalil Morpho Sys2* is supplied. Accordingly, the whole text is diacritized. However, this proposed system was not tested experimentally.

Another proposed DL system is by Rashwan *et al.* [47] taking advantage of Deep Beliefs Nets (DBN). The focus of this system is final word diacritics (case-marking). According to the researchers, the process of diacritization could be viewed as a classification problem. The task is to classify the input based on a specifically designed feature vector and to restore the original diacritics of the raw text. The original text is presented to the classifier, and a group of subnets works to extract the desired features, like POS tags. Each sub-net is trained to extract certain kinds of features. Among the study's contributions is the introduction of Confused Sub-set Resolution (CSR) to raise the degree of accuracy of the diacritics. The performance of the system is benchmarked against three best performing systems, and it outperformed them.

Mubarak *et al.* in their study [62] presented a diacritization model which considers the process of diacritization as machine translation (MT). The model depends on the mechanism of sequence to sequence DL. Using the analogy of translation that uses a sequential encoder and a sequential decoder, the undiacritized input text will be encoded and then decoded into the diacritized output. The researchers presented this model as a solution for specifically restoring case-ending diacritics, which had presented a problem for earlier models and usually requires extensive rule-based programs. Diacritization of words is carried out using a voting technique to select the most common form is employed if it is present in different contexts. However, there was the issue of OOV output. The DER of 1.21% and WER of 4.49% were reached; this indicates that the WER is 63.3% lower than the best similar published results.

The study in [28] utilizes NNs to restore diacritics in three languages, namely, Arabic, Yoruba, and Vietnamese. The task is viewed as a sequence labeling problem. Recent research confirmed Temporal Convolutional Neural Networks' (TCN) superiority over RNNs for sequence modeling performance and computational resources. Accordingly, the study in [42] evaluated a variant of TCN, specifically Acausal TCN (A-TCN), which incorporates context from both directions (previous and later) rather than strictly incorporating the previous context as in the case of TCN. Across all three languages, TCN outperforms LSTM in unidirectional architectures. A-TCN yields comparable results to BiLSTM except in Arabic, where the WER drops by ∼2%. Regarding Arabic, researchers further tested the impact of both BiLSTM and A-TCN on syntactic diacritics. BiLSTM yields a better performance (5.1% WER) compared to A-TCN (5.9% WER). Generally, A-TCN and BiLSTM have a comparable performance, making A-TCN an efficient alternative over BiLSTM since convolutions can be trained in parallel. A-TCN is significantly faster than BiLSTM at inference time (270%∼334% improvement in text diacritized per minute).

The study in [63] presented a novel approach for automatic Arabic text diacritization using deep encode-decode RNN followed by several text correction techniques to improve the overall system output accuracy. Experimental results of the proposed system on a *WikiNews* test set show its superior performance, which is competitive with state-of-the-art diacritization methods. Specifically, the system achieved morphological diacritization Word Error Rate (WER) 3.85% and DER 1.12%. These results are higher than MADAMIRA and Belinkov and Glass results and close to [95], [27].

The study in [64] presented several DL models for the automatic diacritization of Arabic text. The models were built using two main approaches: Feed-Forward Neural Network (FFNN) and RNN, with a number of enhancements such as 100-hot encoding, embeddings, and Conditional Random Field (CRF), and Block-Normalized Gradient. The models were tested on a freely available benchmark dataset, and the results show that the models are either better or comparable with other models, that require language-dependent postprocessing. The outcomes of the RNN models on the test set are much better than the FFNN models by about 67%. In addition, the study showed that diacritics in Arabic could be used to develop the systems of NLP tasks, for instance, Machine Translation (MT), by proposing the 'Translation over Diacritization' approach.

The study in [65] focused on restoring the syntactic diacritics in Arabic. Previous studies showed that restoring syntactic diacritics using Long Short-Term Memory (LSTM) networks leads to state-of-the-art performance. Usually, these LSTM networks include Maximum Entropy (MaxEnt) sparse direct

connections between the input and the output layers of the tagger. The presented research suggests improving such tagger performance by using an ensemble of taggers. However, an ensemble of taggers will require enormous computational and memory resources. Accordingly, the proposed study implemented a knowledge distillation technique in which an ensemble of teachers/taggers is used to train a single student tagger. Another issue regarding Arabic is that its a morphologically rich language, and therefore it has a high OOV rate. To solve this problem, the researcher utilized character embeddings encoded using LSTMs. The researcher states that implementing rule-based correction of errors can develop the system. Therefore, three postprocessing correction rules were implemented. On the ATB task, the hybrid LSTM/MaxEnt tagger achieves 1.0% absolute WER improvement over a strong baseline using the two techniques discussed above. The proposed system leads to a state-of-the-art performance as compared to multiple automatic diacritization systems.

The study in [66] is the first to use Gated RNN to restore diacritics in Arabic. A Gated Recurrent Unit (GRU) is similar to an LSTM, but requires less computation and fewer parameters. According to the researchers, RNNs are suitable for capturing dependencies among sequential data types. Their problem is that they remain weak on long-term dependencies. Indeed, the superior performance of RNN comes at the expense of costly model training, requiring days or weeks in some experimental settings and a significant computation capability. Gated recurrent neural networks (GNN) have been proposed to resolve this problem. They are mainly used in situations where fast training is needed with limited computation capability. The analysis showed that GRU gives comparable results to LSTM in diacritization accuracy and improves the training process. GRU outperforms LSTM in training. The training time is reduced by 18.82%. Accordingly, it is assumed that GRU gives satisfactory results in Arabic diacritization.

Researchers in [67] argued that previous sequence-to-sequence based approaches showed significant results in NLP. They used a RNN involving LSTM models. Interestingly, these are language-independent and do not require any morphological and syntactic analysis or external resources. Accordingly, the model becomes agnostic to new languages. In their study, they considered the diacritization problem as a classification problem. They extended the previous work with an alternate RNN architecture that involves GRU as in [66]. The basic RNN model is augmented with attention-based mechanisms. Compared to the state-of-the-art results, the GRU attentive model improved the DER by 0.19% and WER by 2.32%. The findings indicated that Arabic diacritic patterns do not require as many parameters as LSTMs, but rather fewer parameters as GRU. It is worth noting that the approach followed in the study is character-based. It is expected that incorporating POS tagging in the system will provide a better pattern caption.

The study presented in [68] discussed BiLSTM neural networks with conditional random fields (CRF) for Arabic diacritization. The system required no morphological analyzers, dictionary, or feature engineering. Instead, it used a sequence-to-sequence schema. The input was presented as a sequence of characters that constitute the sentence, while the output consisted of the corresponding diacritic(s) for each character in the sentence. The model is composed of six layers: an input layer, a character embedding layer, a BiLSTM layer, a time-distributed layer, a CRF layer, and an output layer.

The performance of the proposed system was tested using four datasets with varying sizes and genres. The datasets are the King Abdulaziz City for Science and Technology text-to-speech (KACST TTS) dataset, the Quran, Sahih Al-Bukhary, and the Penn Arabic Treebank (ATB). The trained models achieved a DER of 3.41%, 1.34%, 1.57%, and 2.13%, and WER of 14.46%, 4.92%, 5.65%, and 8.43% on the KACST TTS, Quran, Sahih Al-Bukhary, and ATB datasets, respectively. Comparison of the proposed system with earlier studies and existing systems revealed that its results are comparable to or higher than the results of the state-of-the-art methods. The researchers argued that this increased accuracy is because the system is based on character units rather than word units. The prediction capabilities resulting from combining DL networks with CRF were very accurate. However, as with other DL approaches, the proposed system required larger memory, computational resources, and time than other ML approaches.

In a recent study [69], the Deep Belief Network (DBN) is used as a diacritizer for Arabic texts. DBN is one type of DL algorithm that has recently proved to be very effective for various ML problems. A DBN is a probabilistic generative model which is trained using a greedy layer-wise method. In this algorithm, the DBN learns one layer at a time in an unsupervised way and then undergoes fine-tuning through supervised learning with backpropagation. DBNs were introduced as a solution for the dilemmas encountered when using traditional deep neural networks, such as slow-paced learning, getting stuck in local minima, and demanding a lot of training data. The study provides the first attempt to implement and examine the performance of the DBN to automatically diacritize Arabic. The system does not employ any prior morphological or contextual analysis, and it does not require post-processing of data. The DBN was trained to classify each input letter with the corresponding diacritized version. Evaluation of the system was done using two benchmark datasets, the LDC ATB3 and *Tashkeela*. The best settings achieved a DER and WER of 2.21% and 6.73%, respectively, on the ATB3 with an improvement of 26% over the best reported results. On the *Tashkeela* benchmark, the system reaches a high accuracy with a DER of 1.79% and 14% improvement. Such promising results are expected to be even higher if a hybrid approach is followed in future studies combining DBN with LSTM.

2) HYBRID NEURAL APPROACHES

The study in [29] combines both rule-based and neural methods to restore diacritics in Arabic texts. It presented a model

for Arabic morphological diacritization using RNN. LSTM cells were trained in several configurations and embedding levels to model the various morphological features of Arabic words. The experiments show that these models outperform state-of-the-art systems without explicit use of feature engineering. Nevertheless, adding learning features from a morphological analyzer to model the space of possible analyses provided a significant improvement in the system, which is not achieved even through character-level embeddings. The results show substantial gains in accuracy with the use of several evaluation metrics. The proposed model resulted in a 4.4% absolute increase over the state-of-the-art in full morphological analysis accuracy (30.6% relative error reduction) and 10.6% (31.5% relative error reduction) for OOV. The researchers suggest exploring other DL architectures for Arabic diacritization, such as sequence to sequence models. They also suggest investigating the role of syntax features in DL models for case-marking diacritics.

Researchers in [70] investigated the efficiency of a hybrid approach for Arabic diacritization based on a RNN in addition to a rule-based method. The network is BiLSM which exploits long contexts in both directions to reach highly accurate results. The MADAMIRA morphological and syntactical analyzer was used to assist the RNN. High confidence diacritics and the word segmentation output of the MADAMIRA analyzer is fed to the RNN, which produces the fully diacritized output. Using the LDC ATB3 benchmark, the suggested hybrid approach performed better than the statistical approach. It achieved a DER and WER of 2.39 and 8.40%, respectively. This means 34% and 26% improvements, respectively, over the best previous hybrid results. Parallel software and hardware were used to build the RNN, while the CURRENT library was used to run the RNN on a GPU with 16 streaming multiprocessors. Compared to earlier RNN-based systems, the proposed system is 326 times faster to train and took an average of 0.003 seconds to diacritize a word. This speed makes training on vast data sets feasible to build larger and more accurate deep neural networks. Researchers expect better diacritization accuracies by training the RNN on larger data sets in the range of tens of million words.

The study in [71] also followed a LSTM and a maximum entropy (MaxEnt) network hybrid system to restore syntactic diacritics in Arabic. The study views assigning syntactic diacritics as a tagging problem. The LSTM networks were augmented with sparse direct connections between the input and output layers of the tagger (which are the MaxEnt connections). On the Arabic Treebank task, this hybrid LSTM/MaxEnt approach achieved results competitive with to the state-of-the-art systems. Furthermore, the word level tagger is significantly faster than traditional character-based RNN methods. The researcher suggests the use of an ensemble of classifiers to improve the syntactic diacritization accuracy. In addition, POS embeddings are expected to enhance the modeling process.

Researchers in [72] used a feature-rich RNN model that depended on various linguistic and surface-level features to restore both core-word diacritics and case-endings. The study employed a character-level RNN model informed using word morphological information and a word unigram language model to recover CW diacritics. Data for training was the one that was used in training *Farasa*. For testing, the researchers used the freely available *WikiNews* test set. The proposed model was higher than the best state-of-the-art system by 6% for MSA. Also, a new feature-rich RNN-based CE recovery model was introduced; it achieved an error rate that is 60% lower than the current state-of-the-art for MSA. The model surpasses all previous state-of-the-art systems with a CW error rate (CWER) of 2.9% and a CE error rate (CEER) of 3.7% for Modern Standard Arabic (MSA) and CWER of 2.2%, and CEER of 2.5% for Classical Arabic (CA). When combining diacritized word cores with case-endings, the WERs are 6.0% and 4.3% for MSA and CA, respectively. This highlights the effectiveness of feature engineering for such deep neural models. Reliable NLP tools can raise the accuracy of the diacritizing systems. It is expected that augmenting the model with POS tagging information and a bigram language model can increase its accuracy. As for case-endings, the researchers intend to examine the effectiveness of the proposed features for Arabic parsing.

In [53], the researchers followed a hybrid approach that combines DL with rule-based systems to reach diacritization optimal accuracy. The system includes three components in a sort of pipeline, namely, a DL model, which is a multi-layer RNN with LSTM and Dense layers, a character-level rule-based corrector to prevent a group of errors, and a word-level statistical corrector which uses the context and the distance information to fix some diacritization issues. This approach is unique in a way that combines methods of different diacritization systems and adds distance-based corrections. The system was trained and evaluated on a large part of the *Tashkeela* corpus. It outperformed all the tested systems by achieving a DER of 3.39% and a WER of 9.94% when considering all diacritics, and a DER of 2.61%, and a WER of 5.83% when neglecting the diacritization of the last letter of every word. The system performs well even on documents that contain unseen words or non-Arabic words and symbols

3) NEURAL NETWORKS AND DIACRITIZATION FOR SPEECH SYNTHESIS

While the study in [17] falls under the category of Arabic text to speech synthesis field, the realization of diacritics in non-diacritized texts is an essential step for that task. The study tackles restoring gemination at one step then restoring other diacritics as a second step. Different types of DNN were tested for that purpose: feedforward, recurrent (both LSTM and BiLSTM), and hybrid DNN models. The results prove the particular efficiency of RNN system in the different phases of the study (e.g. namely gemination and diacritization). Specifically, the BiLSTM models, used either on its own or

in a hybrid architecture, yielded the highest accuracy rate of more than 99% for gemination sign prediction and 85% for diacritization, with balanced values of precision, recall, and F1 score.

A system within the same field is presented in [22]. The researchers argued that Arabic text-to-speech synthesis from non-diacritized text is still a large challenge because the missing short vowels and consonant doubling have a significant effect on the accurate pronunciation of Arabic. The study adopted a grapheme-to-phoneme conversion system for Arabic based on deep neural networks (DNN). The first step in the system starts with predicting the diacritic and gemination signs achieved via the DNN. The second step is the grapheme-to-phoneme conversion of the diacritized text, which was performed using the Buckwalter code. Because germination is a specific sign, it is treated separately in a whole module. This module was processed before the diacritization module in the text analysis phase. For each type of the DNN, namely, feed-forward DNN, LSTM, and BiLSTM, several models were implemented with different parameters, i.e., number of hidden layers, number of nodes in each hidden layer, activation functions, optimizer type, etc. Hybrid architectures merging feed-forward layers (dense layers) and recurrent ones (LSTM/BiLSTM) were also implemented. The final selection of the DNN models was made upon the accuracy rate given on the validation set. The study confirmed that RNN, i.e., LSTM and BiLSTM, are more fitted to natural language modeling than the pure feedforward network, i.e., All-Dense architecture. Compared to state-of-the-art approaches, the proposed model gives a higher accuracy rate either for all diacritics or for each class, and high precision, recall, and F1 score for each class of diacritic signs.

In [23], an experimental study focused on automatic geminating of Arabic consonants using deep neural networks. Gemination prediction was achieved as a part of an automatic diacritization module for DNN-based Arabic text-to-speech synthesis. Different DNN models were examined using feed-forward and recurrent architectures. Several models were tested using varying parameters for each kind of DNN network, i.e., feed-forward DNN, LSTM (long-short term memory), and BLSTM (Bi-direction LSTM). Also, hybrid architectures merging feed-forward layers (dense layers) and recurrent ones (LSTM/BiLSTM) were implemented. The results showed the ability of specifically RNN to detect the consonants which have to be geminated in a non-diacritized Arabic text, with very high accuracy. The study confirmed that recurrent networks are more suitable for natural language modeling. This supports earlier research which revealed that LSTM and BLSTM-based networks had shown more ability to model the recurrent nature of speech and language.

The study presented in [23] attempted to resolve problems faced in earlier studies in text-to-speech synthesis models in Arabic. The researchers found that their earlier systems synthesized speech with many pronunciation errors. The basic source of these errors is the missing diacritics in Arabic texts. They proposes three charter-level, DL models to restore Arabic text diacritics. The three models were trained using the three corpora extracted from the *Tashkeela* corpus. The first model is a baseline model, which consists of an embedding layer and three BiLSM layers. The second model uses an encoder-decoder architecture, similar to the researchers' text-to-speech synthesis model with the required modifications. The last model focuses on the encoder part of the text-to-speech model, which achieved state-of-the-art performances in both WER and DER measures. It also trained much faster than other models. In addition, the results showed that there are significant differences in diacritics of Modern Standard Arabic (MSA) and Classical Arabic (CA). These differences require a large amount of data from both varieties. Researchers urge more work in future to collect more diacritized MSA text and build sufficient diacritized corpora, which will benefit future studies in the field. Furthermore, it is argued that there are a number of ways to improve the systems, e.g., trying different hyper-parameters, changing the encoder-decoder model's attention mechanisms, and trying other architecture such as the transformer language model.

4) INPUT UNIT

The study in [74] focused on the best candidate units of analysis in diacritization systems. Word level analysis captures semantic and syntactic relationships in sentences. However, most word-level models suffer from the problem of sparsity due to insufficient training examples. Developing large training datasets that include all possible diacritic variants is not feasible because this task requires extensive time and effort. Furthermore, word-level models face computational complexity in training due to the extensive input and output vocabulary size.

On the other hand, diacritic restoration at the character level encodes local contextual information, minimizing the sparsity issue and improving model generalization. Nevertheless, character-level diacritic restoration systems lose a degree of semantic and syntactic knowledge, increasing the possibility of the composition of invalid words. The study investigated employing sub-words as input units and their diacritic patterns as output instead of a word or character-based models to solve this problem. The experiments show that characters provide the optimal level of information for sequence-based diacritic restoration models among different languages. The diacritic restoration model was improved by maximizing the output diacritic sequence using a Conditional Random Field (CRF). Adding the CRF layer in the BiLSTM architecture improved the performance on observed and unobserved words substantially for Arabic.

The study in [35] proposed a novel architecture for labeling character sequences that achieves state-of-the-art results on the *Tashkeela* benchmark. The system's core is a hierarchical, two-level recurrence hierarchy that separately operates on the word and character levels. This feature enabled faster training and inference between the two levels than earlier traditional models. A cross-level attention module connects the two and opens the door for network interpretability.

Two modules were proposed: the Two-Level Diacritizer (D2) and the Two-Level Diactritizer with Decoder (D3). D3 extends D2 by allowing partially diacritized text as input which is a much-needed feature for neural diacritizers. The results showed that D2 outperforms fully character-based models in both task and runtime performance measures. The task module is a softmax classifier. It enumerates valid combinations of diacritics. This system can be extended with a recurrent decoder that optionally accepts priors from partially diacritized text, which improves results. The best model achieves a WER of 5.34%, outperforming the previous state-of-the-art with a 30.56% relative error reduction.

The researchers in [75] proposed a study using NN to restore case-ending diacritics, which is a significant challenge to automatic Arabic diacritization. They offered a word-level bidirectional LSTM model for the prediction of diacritics on word endings. The model was evaluated using the filtered Arabic dataset [6], constructed from the freely available *Tashkeela* corpus. This approach's prediction accuracy is compared with that of character-level systems that indistinctly model word endings and aim to evaluate its effectiveness assessed solely on the word ending diacritics. Although the model yielded relatively poor performance, it captured salient features of word-ending diacritics that are rarely captured by the character-level baseline. Analysis showed that the order of word-ending diacritics is not well defined in the system, and there is a greater level of diversity in what is allowable. Arabic is a Verb-Subject-Object (VSO) order language. Yet, Object-Subject-Verb (OSV) and Object-Verb-Subject (OVS) orders are allowed in Arabic, and they occur with notable frequency in ancient Arabic texts. Therefore, due to this variation, the model could not sufficiently learn word-ending diacritics based on the word's position in the sentence. Furthermore, compared to the model in [2], which utilized stacked LSTMs and stacked RELU layers that worked sufficiently well for their use case, the model created in this study used one LSTM and one RELU layer. This model is notably weaker than the baseline character-level counterpart. More complex stacked LSTM models are expected to yield better results. Researchers concluded that overall, prediction at the word level is less effective than character-level.

The study in [4] offered a solution to combine the benefits of both character-level and word-level diacritic restoration approaches. Most state-of-the-art diacritic restoration models are built on the character level. This approach helps in generalizing the model to unseen data but at the same time loses valuable information at the word level. Thus, to compensate for this loss, the researchers investigated the use of multi-task learning to optimize diacritization with related NLP problems, which are word segmentation, part-of-speech tagging, and syntactic diacritization. For all architectures, the main component is BiLSTM. Results showed that when morpheme boundaries and diacritics are learned jointly, the WER performance is slightly reduced on all and OOV words. This reduction is mainly attributed to lexical diacritics. In addition, enforcing inflectional diacritics through an additional focused layer within the diacritic restoration model improves syntactic WER compared to the baselines. The proposed joint models significantly outperform the baseline systems. They are comparable to the more complex, state-of-the-art models, relying on morphological analyzers and a lot more data. In general, there were statistically significant improvements across all evaluation metrics. This result clarifies the importance of considering additional linguistic information at morphological and sentence levels. Furthermore, including semantic information through pre-trained word embeddings within the diacritic restoration model developed the diacritic restoration performance. However, the OOV performance is still an issue in need of a solution for diacritic restoration systems.

5) PARTIAL DIACRITIZATION

To solve problems, such as sparsity encountered in previous research, some researchers argued against restoring all the diacritics in Arabic texts. The study in [9] divided the diacritization schemes into four types. Each type is appropriate for some specific research objectives. First, full diacritization is the process of adding all the diacritics to the text. Second, half diacritization involves supplying only morphological diacritics while leaving out case-endings. Third, partial diacritization refers to selectively restoring a group of diacritics required for specific purposes, such as homograph disambiguation. This results in a less crowded output text. Finally, minimal diacritization is the process of employing the smallest number of diacritics required for a specific target. Studies in this section attempted to try to limit the number of the needed diacritics in Arabic texts.

Sparsity is a major drawback of most DL approaches to diacritic restoration [76]. To propose a solution for that problem, some studies suggested the idea of partial or minimal diacritization, such as [76]. Rather than restoring all diacritics in the written text, the solution is to add diacritics that are sufficient for resolve lexical ambiguity and leave out other diacritics. The study in [76] focused on supplying diacritics to a subset of words, namely homographs. To balance sparsity and lexical disambiguation, the study proposed approaches for automatically marking a subset of words for diacritic restoration, which leads to selective homograph disambiguation. Two methods for creating ambiguity dictionaries from undiacritized text were proposed. They are using a morphological analyzer and unsupervised sense induction. Clustering and translation-based methods were used to create ambiguity dictionaries from diacritized texts. The various ambiguous word selection strategies were evaluated extrinsically on several downstream applications, such as neural machine translation, part-of-speech tagging, and semantic textual similarity. The study revealed promising results, where the devised strategies on selective diacritization led to a more balanced and consistent performance in downstream applications. The findings demonstrated that partial diacritization achieves a balance between homograph disambiguation and sparsity effects.

Similarly, the study in [27] introduced a partial (i.e., minimal) diacritization scheme for Arabic. According to them, some of the diacritics restored in previous studies are unnecessary and are considered noise in data. They believe that it is crucial to decide which diacritics are needed. In their view, this is a more challenging task than restoring all diacritics. Their model is based on two resources, namely, the output of a morphological analyzer (MADAMIRA) in addition to WordNet [32]. The goal of the morphological analyzers is to generate all word candidates for the diacritics, and the model eliminates word ambiguity through a statistical approach and context similarities. They implemented their system using Java. Out of 80 paragraphs, the system resolved 57 cases. The researchers did not evaluate their methods empirically to demonstrate their effectiveness for NLP applications.

The study in [77] was the first to use NN in classifying and diacritizing Arabic poetry. They stated the problem of the study, emphasizing that reading a poem eloquently requires knowing the poem's meter and obtaining a diacritized version of its verses. However, most of the digital Arabic poems are not diacritized. The presented study proposed solutions to classify input Arabic text into the 16 poetry meters and prose. The adopted approach is ML, using a large dataset of 1657 k verses of poems and prose to develop neural networks that can classify and diacritize Arabic poetry. The dataset is a refined version of the Arabic Poetry Comprehensive Dataset, which is extended to include prose samples. Deep RNN with bidirectional long short-term memory cells was used. The network has an embedding layer at the input, four hidden bidirectional LSTM layers, and a softmax output layer. The model classifies the input with an average accuracy of 97.27%, which is significantly high. It also detected the whole 17 classes of texts. Solutions were proposed to achieve an accuracy that approaches 100% when multiple verses of the same poem are available. This could be done by predicting the class from the aggregate probabilities of the numerous verses. Concerning diacritization, the researchers investigated adding the verse missing diacritics using another RNN. They concluded that diacritizing poetry is more complex than diacritizing prose because of the poet's unique selection of words and the relaxation of some diacritization and grammatical rules.

VIII. RESULTS AND DISCUSSION

The previous overview of 56 recent studies targeting automatic diacritization in Arabic revealed many significant findings. The synthesis of the results of the mentioned studies can be briefly summarized in the following general points:

1. While rule-based methods were the earliest methods in retrieving diacritics in Arabic texts, they are still valuable for capturing certain aspects of the diacritic system which could not be achieved via other methods. Accordingly, interest in this approach did not completely decline. The implementation of those morphological and syntactic rules raised the accuracy of many statistical and DL methods. However, formulating those rules is laborious and requires linguistic experts, however. In addition, living languages are in a constant state of change. New words are added to languages, and some grammatical rules are changed or relaxed. This necessitates the continuous revision and update of those rules by language specialists.

2. Hybrid approaches to restore diacritics seem to reach the highest accuracy rates among other approaches. That is because each system can suit specific diacritic issues, and a hybrid system can benefit from different advantages of its sub-systems. Indeed, the diacritization process can be divided into several tasks of various natures. Each task could be achieved with the use of a specific model. There is a wide range of possibilities in this research venue. Future studies are recommended to continue building effective hybrid systems for Arabic diacritization. It was evidenced that restoring gemination as an independent step reached almost a perfection level of accuracy. Therefore, it is clear that accurate diacritization cannot be achieved through a ''one size fits all'' approach.

3. Statistical methods such as HMM, SVM, and n-grams were efficient for restoration of a good portion of Arabic diacritics with minimum efforts. Nevertheless, they cannot reach optimal accuracy without the aid of other rule-based or DL methods.

4. DL methods using NNs achieved very accurate results. Collectively, BiLSTM RNN was the most suitable NN for Arabic diacritization purposes. BiLSTM networks are successful because they exploit long stretches of context in both directions of input. Hence, they have more extensive linguistic resources than other systems. Some other types of NNs were effective in Arabic diacritization, such as DBN, TCN, and GRN. While the DL approach effectively supplies diacritics in Arabic texts, it suffers from low speed and huge computational and memory resources. In addition, they also require a large corpus for training purposes. Future studies can address those issues to develop accurate, speedy, and economical DL systems for Arabic diacritization. One of the critical elements of training a DL model for Arabic diacritization is the quality and sufficiency of training data. Most of the time and effort put into this stage critically affects the following steps of learning. Feeding a DL system clean and normalized data allows for proper learning.

5. Among the issues that have not been resolved yet are the sparsity and OOV, a complex which stems from the rich morphological nature of Arabic. There is still a need for more studies that address this problem and suggest practical solutions.

6. The evaluation measures in different studies need to consider and report the processing speed of the system. In addition, it is recommended that the cost of developing the diacritization system, the limited number of symbols, and the amount of effort implemented in the construction of the system, are mentioned in the study.

7. There is consensus among studies that a large proportion of the errors generated by different systems are due to case-ending diacritics. Linguistically, morphological and syntactic

TABLE 3. Summary of the reviewed studies which depended on rule-based methods of diacritization.

Researchers	Date	Tool name	Method	Corpus	Evaluation measure	Results	
Fadel et al.	2019	*Farasa*	Basically Rule-based , some with SVM	A dataset from Tashkeela Corpus	DER/WER (with and without case-ending) including (no diacritic and excluding no diacritic)	DER 21.42%	WER 58.88%
		Harakat				18.37%	41.83%
		MADAMIRA				34.38%	76.58%
		Mishkal				16.09%	39.87%
		Tashkeela-Model				49.96%	96.80%
		Shakkala	Neural Network			3.73%	11.19%
Alansary	2018	*Alseraj*	Linguistic rules and a dictionary	International Corpus of Arabic	DER/WER	8.68%	18.63%
Alosaimy & Atwell	2018	Not mentioned	Diacritic borrowing	Riyadh As-Salheen	Percentage of diacritized words/DER	76.41%.	DER 0.04%
Neme & Paumier	2020	*Arabic-Unitex*	Dictionary lookup	A composed language resource	Not Evaluated	Not Evaluated	
Obeid et al.	2020	*CAMeL*	Python 3 toolkit Rule-based program, a statistical model, and a neural network (unigram language model)	Dev. Set of Penn Arabic Treebank (1, 2, 3)	Percentage of accuracy	90.9%	
Fashwan et al.	2016	Not mentioned	Morphological and syntactic analysis	International Corpus of Arabic/ LDC's Arabic Treebank	WER (case-ending only)	9.97%	

TABLE 4. Summary of the reviewed studies which depended on statistical methods to restore diacritics.

Researchers	Date	Method	Corpus	Evaluation measure	Results	
Elshafei et al.	2006	Hidden Markov model/ Viterbi decoder	The Holy Quran	WER	2.5%	
Darwish et al.	2017	Viterbi decoder/SVM	A diacritized corpus containing 9.7 million tokens for training / Wiki news for testing	WER/DER	12.76%	3.54%
Zitouni et al.	2006	maximum entropy framework	Penn LDC's Arabic Treebank	WER/DER	17.3% with case-ending 7.2% without case-ending	5.1with case-ending 2.2 without case-ending
Hifny Hifny	2012 2013	dynamic programming approach/ n-gram language models	*Tashkeela*	WER/WER (no case-ending)	12.5%	4.4%
Hifny	2019	Byte Pair Encoding	*Tashkeela*	WER with and without case-ending	method outperforms character-based methods commonly used	Syntactic diacritics remained low
Nelken & Shieber	2005	A cascade of weighted finite-state transducers & simple rule-based methods	Arabic Treebank	WER/WER (no case-ending)	23.61%	7.33%
Khorsheed	2018	Hidden Markov model	DS1: built by King Abdulaziz City for Science and Technology DS2: Quran Ds3: a corpus of Arabic names	DER in three DSs	DS1: 27.06% DS2: 29.08% DS3: 19.95%	
Hadjir et al.	2019	Hidden Markov model/ Viterbi decoder	*Tashkeela*	Accuracy percentage	Word level 80%	Character level 90%

TABLE 5. Summary of the reviewed studies which depended on hybrid methods to restore diacritics.

Researchers	Date	Method	Corpus	Evaluation measure	Results	
Shaalan et al.	2009	lexicon retrieval, bigram, SVM & statistical prioritized techniques	LDC's Arabic Treebank	WER/DER	11.79%	3.24%
Rashwan et al.	2010	A* deep lattice search, long-horizon n-grams probability estimation, & factorized morphological analysis and POS tagging	1. a 750K corpus compiled for the study's purposes 2. TRN_DBII 3. TST_DB	WER with and without case-ending	12.5%	3.1%
Said et al.	2013	automatic correction, morphological analysis, POS tagging, out-of-vocabulary diacritization components, & HMM and Viterbi algorithms	ATB test And a personally compiled corpus	WER with and without case-ending	11.4%	4.4%
Zayyan et. al.	2016	n-grams and a morphological analyzer	Nemlar written corpus Le Monde Diplomatic corpus	WER/DER	7.1%/ 3.9%. If case-ending is considered	5.1% and 2.7% If case-ending is ignored
Metwally et al.	2016	machine translation approach, morphological analysis approach, & the sequence labeling approach. HMM, a morphological analyzer and CRF	LDC Arabic Treebank 3	Morphological WER and syntactic WER	4.3%	9.4%
Chennoufi & Mazroui	2017	morphological analyzer Alkhalil Morpho Sys2, hidden Markov model and Viterbi algorithm,	Tashkeela, Nemler corpus,a part of RDI corpus	WER with and without case-ending	6.28%	2.58%
Fashwan et al.	2017	rule-based morphological disambiguation, statistical-based morphological disambiguation & smoothing techniques	LDC's Arabic Treebank	WER/DER with and without case-ending	14.87% 4.81%	4.11% 1.93%

diacritics perform two distinct linguistic rules. While morphological diacritics are phonemic, which means that they are part of the phonological representation of the lexical item, syntactic diacritics are allophonic and are assigned depending on the word position in the sentence. While both are important, morphological diacritics are more crucial for NLP applications. The difference between these types of diacritics is reflected in the different computational requirements

TABLE 6. Summary of the reviewed studies using NN.

Researchers	Date	Method	Corpus	Evaluation measure	Results	
Belinkov & Glass	2015	FeedForward, RNN, LSTM, BiLSTM	ATB	DER on all diacritics and on case-ending	RNN is better than FeedForward, BiLSTM is better than LSTM	5.39% 8.74%
Ali et al.	2020	FeedForward, RNN, LSTM, BiLSTM, hybrid	RDI training data	Percentage of accuracy	The efficiency of RNN (specifically BiLSTM) over other types of NN	99% for gemination 85% for diacritiztion
Rabai & Ben Ayed	2015	A hybrid approach: FeedForward DNN, LSTM, BiLSTM	Not specified	DER	RNN (LSTM & BiLSTM) are more fitted to NLP than FeedForward	17.7%
Ali et al.	2019	FeedForward DNN, RNN, LSTM, BiLSTM (and a hybrid of them)	Source is not specified	Accuracy percentage	RNN is more suitable to NLP	
Abandah et al.	2015	Stacked BiLSTM	LDC ATB3 Quran	WER/DER Percentage of error reduction	Sample of 11 books DER 2.09% WER 5.82%	LDC ATB3 Reduced errors DER 25% WER 20% Last letter error 33%
Mijlad et al.	2019	Hybrid of deep LSTM, morpho-syntactic analyzer (*Alkhalil Morpho Sys2*)	Not available	Not evaluated		
Rashwan et al.	2015	Deep Beliefs net, confused sub set resolution	TRN_DB_I TRN_DB_II TST_DB ATB	Accuracy percentage	Syntactic accuracy 88.2%	Morphological accuracy 97
Abandah & Abdel-karim	2020	RNN, LSTM, BiLSTM, encoder decoder LSTM with features	LDC ATB3 *Tashkeela*	DER	The superiority of BiLSTM over the other types	DER ATB3 2046% DER *Tashkeela* 1.97%
Mubarak et al.	2019	Sequence to sequence DL (encoder-decoder, voting technique)	A modern diacritized corpus of 4.5 million tokens WikiNews	DER WER	1.21%	4.49% lowered by 63% when compared to best-published results
Alqahtani et al.	2019	Acausal TCN (A-TCN), LSTM, BiLSTM	ATB 1,2,3	WER	TCN outperforms LSTM, comparable results to BiLSTM	For syntactic diacritics BiLSTM WER 5.1% TCN 5.9%
Noaman et. al.	2018	deep encode-decode RNN, text correction techniques	WikiNews	WER/DER	3.85%	1.12%
Hifny	2021	Hybrid of LSTM, Maximum Entropy, Postprocessing error correction techniques	ATB	WER	1.0% absolute improvement over a strong baseline	
Alqahtani & Diab	2019	BILSTM & Conditional Random Field	ATB 123	WER/DER	characters provide the optimal level of information for sequence-based diacritic restoration models	7.6 % 2.7%
Zalmout & Habash	2017	Hybrid approach: LSTM & morphological analyzer	Penn ATB 1,2,3	Percentage of accuracy	4.4% absolute increase over state-of-art	10.6% Increase for OOV accuracy
Alqudah et al.	2017	Hybrid approach: LSTM & MADAMIRA morphological & syntactic tool	LDC ATB3	WER/DER	8.40%˙	2.39%
Hifny	2018	LSTM & MaxEnt	LDC ATB 2 &3	Syntactic WER	5.3%	
Madhfar &Qamar	2020	Hybrid: BiLSM layers, encoder-decoder architecture, the encoder part of the text-to-speech model	*Tashkeela*	WER/DER With and without case-ending (CA and MSA)	For MSA 23.55% 6.86% With case-ending	For MSA 17.11% 5.87% Without case-ending
Alkhathlan et al.	2020	Gated-Recurrent Units (GRU)	Not specified	WER/DER	2.32%	0.19%
Al-Thubaity et al.	2020	BiLSTM & conditional random fields (CRF)	King Abdulaziz City for Science and Technology text-to-speech (KACST TTS) dataset, the Holy Quran, Sahih Al-Bukhary, & the Penn Arabic Treebank (ATB)	WER/DER for each corpus	14.46%, 4.92%, 5.65%, 8.43%	3.41%, 1.34%, 1.57%, 2.13%
Darwish et al.	2021	RNN model, morphological information, & unigram language model	*Farasa* *WikiNews*	Core WER Case-ending error rate	MSA 2.9% 3.7%	WER combined: MSA: 6.0% CA: 4.3%
Fadel et al.	2019	Hybrid approach: Feed-Forward Neural Network (FFNN), Recurrent Neural Network (RNN), 100-hot encoding, embeddings, Conditional Random Field (CRF) and Block-Normalized Gradient (BNG).	An adaptation of the *Tashkeela* corpus	DER/WER	RNN models on the test set are much better than the FFNN models by about 67%.	DER/WER with case-ending 1.78% 5.38% DER/WER without case-ending 1.39% / 3.04%
Alqahtani et al.	2019	a morphological analyzer & unsupervised sense induction Clustering & translation-based methods	Gigaword 5th edition: Linguistic Data Consortium (LDC); Corpus of Contemporary Arabic; LDC (ATB).	Ambiguous words selection strategies were evaluated on downstream applications: neural machine translation, POS tagging, and semantic textual similarity.	partial diacritization achieves a balance between homograph disambiguation and sparsity effects	
Alnefaie & Azmi	2017	MADAMIRA, WordNet	ATB	Not evaluated empirically		
Alkhamissi et al.	2020	two-level recurrence hierarchy: Two-Level Diacritizer (D2) & Two-Level Diactritizer with Decoder (D3)	*Tashkeela*	WER/DER	5.34% 1.83%	D2 outperforms fully character-based models in both task and runtime performance measures
Abbad & Xiong	2020	Hybrid approach: a multi-layer RNN with LSTM and Dense layers, a character-level rule-based corrector, and a word-level statistical corrector	*Tashkeela*	WER/DER With and without case-ending	DER 3.39% & WER of 9.94% with case-ending	DER of 2.61% and WER of 5.83% without case-ending

TABLE 6. *(Continued.)* **Summary of the reviewed studies using NN.**

Moumen et al.	2018	Gated recurrent neural networks & LSTM	Not specified	error rate and runtime of both GRU and LSTM:	GRN DER 7.31% (all diacritics) 10.79% (last diacritics) Training time is reduced by 18.82% with the use of GRN	LSTM DER 7.15% (all diacritics) 10.50% (last diacritics)
Almanaseer et al.	2021	Deep Belief Network	LDC ATB3 and *Tashkeela*	DER /WER on ATB3 & *Tashkeela*	2.21% and 6.73%, on ATB3 with an improvement of 26%	DER of 1.79% and 14% improvement
Fahmy & Shuaibi	2020	one LSTM and RELU layer	A data set from *Tashkeela*	DER	81.63%	
Alqahtani et al.	2020	multi-task: word segmentation, POS tagging & syntactic diacritization: the main component is BiLSTM	ATB 1,2,3	DER/WER/LER	7.51% 2.54%	4.54%
Abandah et al.	2020	BiLSTM & A softmax output layer	A version of the Arabic Poetry Comprehensive Dataset extended to include prose samples	Accuracy percentage	97.27%	

of each type. That is to say, a system that successfully restores morphological diacritics might not be efficient in restoring syntactic diacritics. It is recommended that morphological diacritics and syntactic diacritics should be viewed two independent problems requiring different systems to handle them.

8. Research into the best models' input unit, namely, character, word, or sub-word, yielded inconclusive results. Generally, models which depended on characters as input seem to reach higher accuracy albeit with some complications. More research is recommended to develop systems that can benefit from the features of both characters and words as input units.

9. The purpose of restoring diacritics vary in different studies. Accordingly, the number of diacritics restored also varied. For instance, in speech synthesis, morphological diacritics are crucial. In homograph disambiguation studies, however, diacritics only of ambiguous words are required. In poetry, case-ending and morphological diacritics are highly important. More linguistic and computational studies are required to decide on the required sub-group of diacritics sufficient for different NLP purposes and how to retrieve this group in Arabic undiacritized Arabic texts.

10. There is a great need to create sufficient Arabic language resources. Future projects need to develop large diacritized Arabic corpora which can be used for training purposes. Most reviewed studies used either the *Tashkeela* corpus or ATB. There is a need for more and larger Arabic diacritized corpora. The use of those corpora is expected to raise the accuracy of diacritization systems. There is a growing need for lexicon retrieval approaches to develop efficient and comprehensive Arabic language diacritized lexicons and update them regularly.

11. It is recommended that prospective researchers to publish their systems and make them open-source. This helps develop and channeling researchers' efforts in one direction rather than leaving them scattered. Collaborative efforts in diacritization systems are expected to achieve outstanding results.

12. It is evident that there is a need to benefit from linguists' expertise in the Arabic language while designing systems and programs of Arabic diacritization. Linguistic knowledge can aid computational methods and suggest possible solutions for different complications encountered. Furthermore, systems have to consider the glottal stop *Hamza* and prolonged vowel *madd* as vital diacritic signs that need to be restored in Arabic texts.

13. A remarkable progress in research about diacritic restoration in Arabic is evident, particularly during recent years. While many studies reached a high level of accuracy, there is still some room for improvement in future studies. No system has yet reached the level of perfection in performance and accuracy with reasonable processing time and computational resources. More studies are recommended to develop diacritization systems that can reach those research targets.

IX. CONCLUSION

This paper presented a survey of 56 research studies in automatic diacritization of Arabic texts, mainly during the last decade. The review classified studies and summarized their collective results. It confirmed that providing diacritics for Arabic texts is not easy since it entails several complications. None of the reviewed studies reached total accuracy, but a number of results in the studies reviewed came closer to this goal.

The recent keen interest in ML and DL is reflected in the growing number of diacritization studies that employed those architectures in systems developed to restore diacritics in Arabic electronic texts. Specifically, the use of BiLSTM networks revealed highly accurate results. However, earlier rule-based methods, including morphological analyzers with extensive coverage, syntactic approaches, and lexicons, also proved to aid statistical methods in ML and DL. The superiority of the Microsoft Arabic spell checker, which is based on lexical resources, over the Google Docs spell checker, which depends on ML, provides further evidence for this conclusion [41]. For Arabic NLP and diacritization systems, it is recommended to take the best from all fields of NLP and linguistics such as morpho-syntactic rules, lexicography, ML, and DL technologies, by developing hybrid approaches that combine advantages from different techniques.

It is recommended for future research projects to develop representative diacritized corpora and lexicons for Arabic with different genres. Even though they may require extensive

efforts, they are necessary tools that will serve many Arabic NLP efforts and enhance the accuracy of developed systems. Future research can address problems not entirely resolved by available diacritization systems such as OOV, partial diacritization schemes, and input units. Based on this survey, it is recommended that Arabic diacritization should be best approached by dividing the tasks into smaller subtasks, specifically regarding morphological and syntactic diacritization.

APPENDIX I

See Tables 3–6.

REFERENCES

[1] I. Guellil, H. Saâdane, F. Azouaou, B. Gueni, and D. Nouvel, "Arabic natural language processing: An overview," *J. King Saud Univ.-Comput. Inf. Sci.*, vol. 33, no. 5, pp. 497–507, 2021. [Online]. Available: https://doi.org/10.1016/j.jksuci.2019.02.006

[2] A. Fadel, I. Tuffaha, B. Al-Jawarneh, and M. Al-Ayyoub, "Arabic text diacritization using deep neural networks," in *Proc. 2nd Int. Conf. Comput. Appl. Inf. Secur. (ICCAIS)*, May 2019, pp. 1–7, doi: 10.1109/CAIS.2019.8769512.

[3] A. Roberts, L. Al-Sulaiti, and E. Atwell, "AConCorde: Towards an open-source, extendable concordancer for Arabic," *Corpora*, vol. 1, no. 1, pp. 39–60, May 2006, doi: 10.3366/cor.2006.1.1.39.

[4] S. Alqahtani, "Full and partial diacritic restoration: Development and impact on downstream applications," Ph.D. dissertation, George Washington Univ., Washington, DC, USA, 2020.

[5] L. Hardesty. *Explained: Neural Networks Ballyhooed Artificial-Intelligence Technique Known as, 'Deep Learning' Revives 70-Year-Old Idea.*' MIT News Office. Accessed: Apr. 4, 2017. [Online]. Available: https://news.mit.edu/2017/explained-neural-networks-deep-learning-0414

[6] N. Habash, A. Shahrour, and M. Al-Khalil, "Exploiting Arabic diacritization for high quality automatic annotation," in *Proc. 10th Int. Conf. Lang. Resour. Eval. (LREC)*, 2016. [Online]. Available: http://www.lrec-conf.org/proceedings/lrec2016/pdf/878_Paper.pdf

[7] M. Almanea, "Lexical associations of the word eden in the Qur'an: A corpus linguistics approach," *Open J. Mod. Linguistics*, vol. 11, no. 4, pp. 4298–4304, Aug. 2021, doi: 10.4236/ojml.2021.114045.

[8] O. Hamed and T. Zesch, "Exploring the effects of diacritization on Arabic frequency counts," in *Proc. 2nd Int. Conf. Natural Lang. Speech Process. (ICNLSP)*, Apr. 2018, pp. 1–6, doi: 10.1109/ICNLSP.2018.8374378.

[9] A. M. Azmi and R. S. Almajed, "A survey of automatic Arabic diacritization techniques," *Natural Lang. Eng.*, vol. 21, no. 3, pp. 477–495, May 2015, doi: 10.1017/S1351324913000284.

[10] F. O. Asahiah, O. À. Odéjobé, and E. R. Adagunodo, "A survey of approaches to diacritic restoration," *Natural Lang. Eng.*, vol. 1, no. 1, pp. 1–23, 1998, doi: 10.13140/RG.2.2.21379.55842.

[11] R. Nelken and S. M. Shieber, "Arabic diacritization using weighted finite-state transducers," in *Proc. ACL Workshop Comput. Approaches Semitic Lang. (Semitic)*, 2005, pp. 79–86. [Online]. Available: https://aclanthology.org/W05-07.pdf

[12] A. Mijlad and Y. E. Younoussi, "Arabic text diacritization: Overview and solution," in *Proc. 4th Int. Conf. Smart City Appl.*, Oct. 2019, pp. 1–7, doi: 10.1145/3368756.3369088.

[13] G. Zhou, "On the embodiment of economy principle in the English language," *English Lang. Literature Stud.*, vol. 2, no. 2, May 2012, doi: 10.5539/ells.v2n2p100.

[14] J. Field, *Psycholinguistics: Key Concepts*. London, U.K.: Routledge, 2004.

[15] P. Whitney, *The Psychology of Language*. Boston, MA, USA: Houghton Mifflin, 1998.

[16] N. Habash and O. Rambow, "Arabic diacritization through full morphological tagging," in *Proc. Human Lang. Technol.: Conf. North Amer. Chapter Assoc. Comput. Linguistics; Companion Volume, Short Papers XX (NAACL)*, 2007, pp. 53–56. [Online]. Available: https://aclanthology.org/N07-2.pdf

[17] I. Hadj Ali, Z. Mnasri, and Z. Lachiri, "DNN-based grapheme-to-phoneme conversion for Arabic text-to-speech synthesis," *Int. J. Speech Technol.*, vol. 23, no. 3, pp. 569–584, Sep. 2020, doi: 10.1007/s10772-020-09750-7.

[18] S. Leijin and F. Veen, "The neural network zoo," *Proceedings*, vol. 47, no. 1, p. 9, 2020, doi: 10.3390/proceedings47010009.

[19] J. Brownlee, "Long short-term memory networks with Python: Develop sequence prediction models with deep learning," in *Machine Learning Mastery*. 2017.

[20] M. S. H. Ameur, Y. Moulahoum, and A. Guessoum, "Restoration of Arabic diacritics using a multilevel statistical model," in *Proc. IFIP Int. Conf. Comput. Sci. Appl.* Springer, 2018, pp. 181–192, doi: 10.1007/978-3-319-19578-0_15.

[21] A. Graves and J. Schmidhuber, "Framewise phoneme classification with bidirectional LSTM and other neural network architectures," *Neural Netw.*, vol. 18, no. 5, pp. 602–610, 2005. [Online]. Available: https://doi.org/10.1016/j.neunet.2005.06.042

[22] 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, Nov. 2015, doi: 10.1016/j.csl.2015.04.002.

[23] I. H. Ali, Z. Mnaseri, and Z. Lachiri, "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, doi: 10.1109/SSD.2019.8893275.

[24] 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. [Online]. Available: https://doi.org/10.1145/1644879.1644881

[25] S. K. Ray and K. Shaalan, "A review and future perspectives of Arabic question answering systems," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 12, pp. 3169–3190, Dec. 2016, doi: 10.1109/TKDE.2016.2607201.

[26] A. Pasha, M. Al-Badrashiny, M. T. Diab, and A. El Kholy, "Madamira: A fast, comprehensive tool for morphological analysis and disambiguation of Arabic," in *Proc. LREC*, vol. 14, 2014, pp. 1094–1101.

[27] R. Alnefaie and A. M. Azmi, "Automatic minimal diacritization of Arabic texts," *Proc. Comput. Sci.*, vol. 117, pp. 169–174, Mar. 2017, doi: 10.1016/j.procs.2017.10.106.

[28] S. Alqahtani, A. Mishra, and M. Diab, "Efficient convolutional neural networks for diacritic restoration," 2019, *arXiv:1912.06900*. [Online]. Available: http://arxiv.org/abs/1912.06900

[29] 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, doi: 10.18653/v1/D17-1073.

[30] A. Abdelali, K. Darwish, N. Durrani, and H. Mubarak, "Farasa: A fast and furious segmenter for Arabic," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics: Demonstrations*, 2016, pp. 11–16, doi: 10.18653/v1/N16-3003.

[31] Y. Hifny, "Open vocabulary Arabic diacritics restoration," *IEEE Signal Process. Lett.*, vol. 26, no. 10, pp. 1421–1425, Oct. 2019, doi: 10.1109/lsp.2019.2933721.

[32] Y. Regragui, L. Abouenour, F. Krieche, K. Bouzoubaa, and P. Rosso, "Arabic wordNet: New content and new applications," *Proc. 8th Global WordNet Conf. (GWC)*, 2016, pp. 333–341, [Online]. Available: https://aclanthology.org/2016.gwc-1.47

[33] 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. (IJDAR)*, vol. 18, no. 2, pp. 183–197, Jun. 2015, doi: 10.1007/s10032-015-0242-2.

[34] G. Abandah and A. Abdel-Karim, "Accurate and fast recurrent neural network solution for the automatic diacritization of Arabic text," *Jordan. J. Comp. Inform. Technol*, vol. 6, pp. 103–121, 2020, doi: 10.5455/jjcit.71-1567402817.

[35] B. AlKhamissi, M. N. ElNokrashy, and M. Gabr, "Deep diacritization: Efficient hierarchical recurrence for improved Arabic diacritization," 2020, *arXiv:2011.00538*. [Online]. Available: http://arxiv.org/abs/2011.00538

[36] Z. Alyafeai and M. Al-Shaibani, "ARBML: Democritizing Arabic natural language processing tools," in *Proc. 2nd Workshop NLP Open Source Softw. (NLP-OSS)*, 2020, pp. 8–13.

[37] S. Alansary, "Alserag: An automatic diacritization system for Arabic," in *Intelligent Natural Language Processing: Trends and Applications*. 2018, pp. 523–543, doi: 10.1007/978-3-319-48308-5_18.

[38] T. Barqawi and T. Zerrouki. (2017). *Shakkala, Arabic Text Vocalization*. [Online]. Available: https://github.com/Barqawiz/Shakkala

[39] M. Anwar. (2018). *Tashkeela-Model*. [Online]. Available: https://github.com/Anwarvic/Tashkeela-Model

[40] A. Alosaimy and E. Atwell, "Diacritization of a highly cited text: A classical Arabic book as a case," in *Proc. IEEE 2nd Int. Workshop Arabic Derived Script Anal. Recognit. (ASAR)*, Mar. 2018, pp. 72–77.
[41] A. A. Neme and S. Paumier, "Restoring Arabic vowels through omission-tolerant dictionary lookup," *Lang. Resour. Eval.*, vol. 54, no. 2, pp. 487–551, Jun. 2020, doi: 10.1007/s10579-019-09464-6.
[42] O. Obeid, N. Zalmout, S. Khalifa, D. Taji, M. Oudah, B. Alhafni, G. Inoue, F. Eryani, A. Erdmann, and N. Habash, "CAMeL tools: An open source Python toolkit for Arabic natural language processing," in *Proc. 12th Lang. Resour. Eval. Conf.*, 2020, pp. 7022–7032. [Online]. Available: https://www.aclweb.org/anthology/2020.lrec-1.868v2.pdf
[43] A. Fashwan and S. Alansary, "A rule based method for adding case ending diacritics for modern standard Arabic texts," in *Proc. 16th Int. Conf. Lang. Eng.*, 2016, pp. 1–16.
[44] M. Elshafei, H. Al-Muhtaseb, and M. Alghamdi, "Statistical methods for automatic diacritization of Arabic text," in *Proc. Saudi 18th Nat. Comput. Conf.*, vol. 18, 2006, pp. 301–306.
[45] 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, doi: 10.18653/v1/W17-1302.
[46] Y. Belinkov and J. Glass, "Arabic diacritization with recurrent neural networks," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2015, pp. 2281–2285, doi: 10.18653/v1/D15-1274.
[47] M. A. A. Rashwan, A. A. A. 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, Language Process.*, vol. 23, no. 3, pp. 505–516, Mar. 2015, doi: 10.5555/2817174.2817183.
[48] 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)*, 2006, pp. 577–584, doi: 10.3115/1220175.1220248.
[49] Y. Hifny, "Smoothing techniques for Arabic diacritics restoration," in *Proc. 12th Conf. Lang. Eng. (ESOLEC)*, no. 1, 2012, pp. 6–12.
[50] Y. Hifny, "Restoration of Arabic diacritics using dynamic programming," in *Proc. 8th Int. Conf. Comput. Eng. Syst. (ICCES)*, Nov. 2013, pp. 3–8, doi: 10.1109/ICCES.2013.6707161.
[51] M. S. Khorsheed, "Diacritizing Arabic text using a single hidden Markov model," *IEEE Access*, vol. 6, pp. 36522–36529, 2018, doi: 10.1109/ACCESS.2018.2852619.
[52] I. Hadjir, M. Abbache, and F. Z. Belkredim, "An approach for Arabic diacritization," in *Natural Language Processing and Information Systems* (Lecture Notes in Computer Science), vol. 11608, E. Métais, F. Meziane, S. Vadera, V. Sugumaran, and M. Saraee, Eds. 2019, doi: 10.1007/978-3-030-23281-8_29.
[53] H. Abbad and S. Xiong, "Multi-components system for automatic Arabic diacritization," in *Advances in Information Retrieval*, vol. 12035. 2020, pp. 341–355, doi: 10.1007/978-3-030-45439-5_23.
[54] K. Shaalan, H. M. A. Bakr, and I. Ziedan, "A hybrid approach for building Arabic diacritizer," in *Proc. Workshop Comput. Approaches Semitic Lang. (EACL)*, 2009, pp. 27–35, doi: 10.5555/1621774.1621780.
[55] M. A. A. Rashwan, M. A. S. A. A. Al-Badrashiny, M. Attia, S. M. Abdou, and A. Rafea, "A stochastic Arabic diacritizer based on a hybrid of factorized and unfactorized textual features," *IEEE Trans. Audio, Speech, Language Process.*, vol. 19, no. 1, pp. 166–175, Jan. 2011, doi: 10.1109/TASL.2010.2045240.
[56] 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.* Springer, 2013, pp. 53–64.
[57] A. A. Zayyan, M. Elmahdy, H. B. Husni, and J. M. Al Ja'am, "Automatic diacritics restoration for modern standard Arabic text," in *Proc. IEEE Symp. Comput. Appl. Ind. Electron. (ISCAIE)*, May 2016, pp. 221–225, doi: 10.1109/ISCAIE.2016.7575067.
[58] A. S. Metwally, M. A. Rashwan, and A. F. Atiya, "A multi-layered approach for Arabic text diacritization," in *Proc. IEEE Int. Conf. Cloud Comput. Big Data Anal. (ICCCBDA)*, Jul. 2016, pp. 389–393, doi: 10.1109/ICCCBDA.2016.7529589.
[59] 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, doi: 10.1016/j.jksuci.2016.06.004.
[60] A. Fashwan and S. Alansary, "SHAKKIL: An automatic diacritization system for modern standard Arabic texts," in *Proc. 3rd Arabic Natural Lang. Process. Workshop*, 2017, doi: 10.18653/v1/W17-1311.
[61] M. Boudchiche, A. Mazroui, M. O. A. O. Bebah, A. Lakhouaja, and A. Boudlal, "AlKhalil morpho Sys 2: A robust Arabic morpho-syntactic analyzer," *J. King Saud Univ.-Comput. Inf. Sci.*, vol. 29, no. 2, pp. 141–146, Apr. 2017, doi: 10.1016/j.jksuci.2016.05.002.
[62] 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, 2019, pp. 2390–2395, doi: 10.18653/v1/N19-1248.
[63] H. Noman, S. Sahran, and M. Rashwan, "A hybrid approach for automatic morphological diacritization of Arabic text," *Hum.-Centric Comput. Inf. Sci.*, vol. 8, p. 12, Apr. 2018, doi: 10.1186/s13673-018-0133-x.
[64] 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," 2019, *arXiv:1911.03531*. [Online]. Available: http://arxiv.org/abs/1911.03531
[65] Y. Hifny, "Recent advances in Arabic syntactic diacritics restoration," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Jun. 2021, pp. 7768–7772, doi: 10.1109/ICASSP39728.2021.9414500.
[66] R. Moumen, R. Chiheb, R. Faizi, and A. El, "Evaluation of gated recurrent unit in Arabic diacritization," *Int. J. Adv. Comput. Sci. Appl.*, vol. 9, no. 11, pp. 360–364, 2018, doi: 10.14569/IJACSA.2018.091150.
[67] A. Alkhathlan, F. Kateb, and J. Kalita, "Attention-based sequence learning model for Arabic diacritic restoration," *Proc. 6th Conf. Data Sci. Mach. Learn. Appl. (CDMA)*, Mar. 2020, pp. 7–12, doi: 10.1109/CDMA47397.2020.00007.
[68] A. Al-Thubaity, A. Alkhalifa, A. Almuhareb, and W. Alsanie, "Arabic diacritization using bidirectional long short-term memory neural networks with conditional random fields," *IEEE Access*, vol. 8, pp. 154984–154996, 2020, doi: 10.1109/ACCESS.2020.3018885.
[69] W. Almanaseer, M. Alshraideh, and O. Alkadi, "A deep belief network classification approach for automatic diacritization of Arabic text," *Appl. Sci.*, vol. 11, no. 11, p. 5228, Jun. 2021, doi: 10.3390/app11115228.
[70] S. Alqudah, G. Abandah, and A. Arabiyat, "Investigating hybrid approaches for Arabic text diacritization with recurrent neural networks," in *Proc. IEEE Jordan Conf. Appl. Electr. Eng. Comput. Technol. (AEECT)*, Oct. 2017, pp. 1–6, doi: 10.1109/AEECT.2017.8257765.
[71] Y. Hifny, "Hybrid LSTM/MaxEnt networks for Arabic syntactic diacritics restoration," *IEEE Signal Process. Lett.*, vol. 25, no. 10, pp. 1515–1519, Oct. 2018, doi: 10.1109/LSP.2018.2865098.
[72] K. Darwish, A. Abdelali, H. Mubarak, and M. Eldesouki, "Arabic diacritic recovery using a feature-rich biLSTM model," *ACM Trans. Asian Low-Resource Lang. Inf. Process.*, vol. 20, no. 2, pp. 1–18, Apr. 2021, doi: 10.1145/3434235.
[73] M. A. H. Madhfar and A. M. Qamar, "Effective deep learning models for automatic diacritization of Arabic text," *IEEE Access*, vol. 9, pp. 273–288, 2021, doi: 10.1109/ACCESS.2020.3041676.
[74] S. Alqahtani and M. Diab, "Investigating input and output units in diacritic restoration," in *Proc. 18th IEEE Int. Conf. Mach. Learn. Appl. (ICMLA)*, Dec. 2019, pp. 811–817, doi: 10.1109/ICMLA.2019.00142.
[75] H. Fahmy and A. Shaib, "Enhancing Arabic diacritization with word-ending specific models," in *Proc. Stanford CS224N Natural Lang. Process. Deep Learn.*, 2020, pp. 1–8. [Online]. Available: https://web.stanford.edu/class/archive/cs/cs224n/cs224n.1204/reports/custom/report06.pdf
[76] S. Alqahtani, H. Aldarmaki, and M. Diab, "Homograph disambiguation through selective diacritic restoration," 2019, *arXiv:1912.04479*. [Online]. Available: http://arxiv.org/abs/1912.04479
[77] G. A. Abandah, M. Z. Khedher, M. R. Abdel-Majeed, H. M. Mansour, S. F. Hulliel, and L. M. Bisharat, "Classifying and diacritizing Arabic poems using deep recurrent neural networks," *J. King Saud Univ.-Comput. Inf. Sci.*, Dec. 2020, doi: 10.1016/j.jksuci.2020.12.002.

MANAR M. ALMANEA received the B.A. (Hons.) and M.A. degrees in applied linguistics from Princess Nourah Bint Abdulrahman University and the Ph.D. degree (Hons.) in applied linguistics from King Saud University. She is currently an Assistant Professor of applied linguistics with the College of Languages and Translation, Imam Mohmmad Ibn Saud Islamic University, Riyadh, Saudi Arabia. Her research interests include the areas of Arabic applied linguistic studies, Arabic corpus linguistics, and computer applications in linguistic research.