

Enhancing Arabic Diacritization Using BERT with BiLSTM and BiGRU Heads

Ahmed Abdelhamed Elgayar¹, Abdul-Hadi Nabi Ahmed¹, Basma E. El-Demerdash¹, Ahmed E. Soliman¹

¹ Faculty of Graduate Studies for Statistical Research (FGSSR), Cairo University, Giza, Egypt

Article Info

Article history:

Received Jul 28, 2025

Revised Apr 22, 2026

Accepted Jun 15, 2026

Keywords:

Arabic Diacritization

Transformer

Long Short-Term Memory

Gated Recurrent Unit

Natural Language Processing

ABSTRACT

Arabic diacritization is the task of adding diacritical marks to Arabic text to aid correct articulation, resolve lexical ambiguity, and support downstream natural language processing tasks. This study proposes a hybrid deep learning architecture that augments a pre-trained transformer encoder (AraBERTv02) with a sequential prediction head built on either a Bidirectional Long Short-Term Memory (BiLSTM) or a Bidirectional Gated Recurrent Unit (BiGRU) network, replacing the conventional linear classification layer. This design introduces an explicit sequential inductive bias at the output level, enabling the model to capture local label dependencies that are not modeled by a standard linear head.

The models were trained and evaluated on the Tashkeela Processed (TP) benchmark corpus of fully diacritized Modern Standard Arabic. The BiLSTM based model achieves a Diacritic Error Rate (DER) of 1.82% and a Word Error Rate (WER) of 2.58% while the BiGRU variant has a DER of 1.84% and a WER of 2.62% both outperforming the prior state-of-the-art on the same benchmark (WER: 3.34%). Error analysis reveals a systematic confusion of short vowels, especially in verb-initial contexts where fatha and damma alternate to mark active and passive voice, illustrating the inherent morphological complexity of Arabic diacritization. The proposed architecture is also shown to be suitable for real-world deployment with an inference time of 9.22 ms and 8.74 ms per sentence for BiLSTM and BiGRU models respectively. Thus, both variants are compatible with near real-time applications such as Arabic text-to-speech systems and educational tools.

Copyright © 2026 Institute of Advanced Engineering and Science.
All rights reserved.

Corresponding Author:

Ahmed Abdelhamed Elgayar,

Department of Data Science,

Faculty of Graduate Studies for Statistical Research, Cairo University,

Al Qusais, Al Qusais 1, Dubai, United Arab Emirates.

Email: a.elgayar17@gmail.com

1. INTRODUCTION

Arabic is one of the most widely spoken languages in the world, used by over 467 million speakers across 26 countries. It is identified by its script, which is written from right to left, comprises 28 letters, and features contextual letter shaping. Each regional dialect, from the Mashreq to the Maghreb, adds richness and depth to the Arabic linguistic landscape. However, in formal writing, education, and media, Arabic speakers rely on Modern Standard Arabic (MSA).

In Arabic, diacritical marks (tashkeel) provide essential information related to short vowels, pronunciation, and syntactic structures are required orthographic notations. Diacritical marks are crucial for adding grammatical meaning to the language and differentiate homographs. Regardless, diacritical marks are ignored in most of the present-day Arabic writing leaving ambiguity within writing that can effect both computational language and human understanding. Ignoring diacritical marks is a considerable issue for Natural Language Processing (NLP) applications because it lowers performance by applications such as information retrieval (IR), machine translation (MT), and text to speech synthesis. Table 1 illustrates common Arabic diacritics using the letter ب as an example.

Table 1. Arabic Diacritics Illustrated with the Letter ب

Arabic Name	English Name	Diacritized ب
فتحة	Fatha	بَ
ضمة	Damma	بُ
كسرة	Kasra	بِ
سكون	Sukun	بْ
تنوين فتح	Tanween Fath	بً
تنوين ضم	Tanween Damm	بٌ
تنوين كسر	Tanween Kasr	بٍ
شدة	Shadda	بّ

Automatic diacritical restoration or Arabic Text Diacritization (ATD) is among the greatest challenges of Arabic Natural Language Processing (NLP). Early research in this area relied on rule-based approaches that encoded both morphological and syntactic rules to infer the appropriate diacritical marks. These systems considered context-sensitive, implicit linguistic structures in a rigid manner; although they produced appealing results, they lacked flexibility. To overcome these limitations, more robust hybrid systems were adopted to provide robustness and to broaden generalization and efficiency by combining statistical methods with handcrafted linguistic rules. Next, based on recognized contextual dependencies in sequential data, machine learning and deep learning approaches particularly using bidirectional long-short-term memory (BiLSTM) network-based models have demonstrated significant improvements. Nonetheless, issues such as dialectal variation control, limited annotated resources, and distant contextual dependencies still present ongoing challenges.

Transformer-based models, which use self-attention mechanisms for modeling relationships among all tokens in a sequence simultaneously, have played a central role in natural language processing advances. In various tasks and languages, these models provide state-of-the-art performance. With our specific focus being on AraBERTv2, a pre-trained BERT specifically tailored for Arabic, we examine the efficiency of transformer models on the task of Arabic diacritization. In this work, we fine-tune AraBERTv2 to the problem of Arabic diacritization as a token classification task. Each character or token is marked with or without a diacritical mark based on the context in which it appears, leveraging the model's ability to generate rich contextual representations.

Two architectural extensions are added to the base AraBERTv2 model to better capture morphological subtlety and sequential interdependence in Arabic script. The additions aim to leverage both the contextual representation capabilities of transformers and the sequential modelling capabilities of recurrent neural networks.

- First Variant: In this configuration a bidirectional gated recurrent unit (BiGRU) layer and then a fully connected classification layer is added to the output of AraBERTv2 transformer encoder. The sequential dependencies are captured by the BiGRU to improve the model's understanding of the character-level patterns.
- Second Variant: Same architecture as above except replacing BiGRU layer with a Bidirectional Long Short-Term Memory (BiLSTM) layer. BiLSTM can effectively capture long range dependencies in sequential data.
- Objective: The architectural variants aim to enhance the model ability to detect and predict diacritical marks by merging the contextualised embeddings of the transformer with recurrent layers modelling sequential structure.

These hybrid architectures were chosen for their demonstrated capability to model both local and global relationships, which are required for modeling the intricate morphological patterns in Arabic as well as for distinguishing between homographic structures. We demonstrate the benefit that comes from the adoption of a mixture of transformer-based and recurrent elements for Arabic diacritization by empirically evaluating the performance of these variants against AraBERTv2 architecture and traditional BiLSTM-based architecture.

The main contributions of this work are twofold. First, we replace the conventional linear token-level classification head of a pre-trained transformer encoder (AraBERTv02) with bidirectional recurrent layers (BiLSTM and BiGRU), thereby introducing an explicit sequential inductive bias at the output level. While AraBERT and recurrent architectures have each been studied independently, their systematic integration as alternative prediction heads for Arabic diacritization has not been thoroughly investigated; existing approaches typically rely on a linear classifier that does not model label dependencies. Second, we provide a controlled empirical evaluation demonstrating that this architectural modification consistently improves diacritization performance, achieving a measurable reduction in both Diacritic Error Rate (DER) and Word Error Rate (WER) under identical benchmark conditions.

2. RELATED WORK

Throughout the last 20 years, Arabic diacritization technology has progressed from heuristic rule-based systems towards cutting edge machine and deep learning models. There was a history beforehand with rule-based studies from [1–3] which relied exclusively on hand-coded linguistic rules, and then long machine learning and neural models which began to be considered, previously developed through other Arabic NLP tasks using text [4–7] and speech [8–10].

Al Sallab et al. [11] were the first to systematically challenge the application of deep learning to automatic diacritization. They proposed a system that used a global Deep Neural Network (DNN) merged with a Confused Subset Resolution (CSR) approach and part-of-speech cues to disambiguate syntactic diacritics in an Arabic sentence and word. After training the DNN, they reviewed its confusion matrix, identified the most confusing subsets, and then trained sub-classifiers on those subsets. The ultimate prediction of diacritics would involve considering the outputs of the DNN and sub-classifier outputs simultaneously. When their approach was evaluated on the Arabic Treebank [12] and RDI corpora [13–15], they obtained a 22% reduction in classification error rate, with syntactic and morpheme error rates of 9.9% and 3%, respectively.

Abandah et al. [16] experimented with several FNN and RNN architectures on the Arabic Treebank (ATB) dataset. The authors reported their highest scoring system was a 3-layer bi-LSTM which achieved a Diacritic Error Rate (DER) of 8.14%. Alnefaie and Azmi [17] disclosed another principle called minimum diacritization, which them to restore only those diacritics that are necessary for disambiguation. Their method combined morphological rule-based analysis and statistical methods to automatically establish which characters in a sentence require diacritics. Their experiments revealed that ambiguity was removed in 56 of the 80 paragraphs of the ATB dataset.

Darwish et al. [18] introduced an automatic diacritization system called Farasa, which combines rule-based diacritization methods with a support vector machine (SVM) learning algorithm for MSA. Since the ATB dataset size is small, the authors trained on a larger closed-source RDI dataset containing 10 million tokens. The authors developed a new dataset with 70 Wikinews articles for testing their experimental system, which achieved a DER of 3.54% and WER of 12.76%.

Mubarak et al. [19] developed a sequence-to-sequence diacritization model evaluated using the evaluation corpus presented previously by [18]. Their diacritization system performed significantly better than the previous systems, with a WER of 4.49%. Fadel et al. [20] proposed the open-source pre-processed dataset called the AD dataset, which was based on the Tashkeela corpus on Kaggle corpus [21]. The AD dataset was composed of 2.5 million words divided into training, validation, and test datasets. They benchmarked existing diacritization systems on the AD dataset and the data indicated that the systems based on deep learning (e.g., Shakkala [22]) outperformed the rule-based systems with a DER of 2.8%.

In a different experiment, [23] evaluated 3 different feed-forward and recurrent neural models, each of the models were trained using their initial training set plus another 22.5 million tokens. The best-performing model, a block-normalized gradient RNN model, reported a Diacritic Error Rate (DER) of 1.69%, and a Word Error Rate (WER) of 5.09%. By enabling the exchange of results with prior work on identical data, they presented a method to diacritize Arabic text which simultaneously increased the validity of Arabic–English machine translation.

Darwish et al. [24] presented a feature-rich LSTM model for Modern Standard Arabic (MSA) and Classical Arabic (CA). Using the RDI corpus for MSA and a brand new curated 3 million token CA corpus, their best results recorded word-error-rates (WERs) of 6% for MSA and 4.3% for CA. AlKhamissi et al. [25] trained a deep neural sequence-labeling model and applied it to the Arabic dialect corpus developed by [20], achieving a document error rates (DER) of 1.94% and a WER of 5.34%.

Abbad and Xiong [26] created a hybrid system that consists of deep learning (RNN - LSTM) and rule-based, statistical context modules while they also released the TP benchmark, which was a pre-cleaned part of Tashkeela corpus [21] as well. Their results on TP showed a DER of 2.61% and WER of 5.83%, leaving out some non-core characters of Arabic. In a subsequent study, [27] applied a BiLSTM deep learning system on subset of TP train data that they published in their previous work [26] Their experimental results showed DER of 2.45% and WER of 5.42% on Arabic core context, and when non-core characters were included, the DER increased to 3% and WER 8.99%.

Hifny [28] suggested a hybrid BiLSTM taggers ensemble connected through a maximum entropy model which allows to maintain and enhance performance while reducing both time-complexity and memory-complexity. Their approach responded to a high rate of out of vocab (OOV) since Arabic is highly OOV, leveraging embeddings on both character and word levels in their method. The approach was tested using the ATB corpus and yielded a WER of 4.3% syntactically which proved reliable. Thompson and Alshehri [29] constructed a multitask learning model whereby they treated the English translation task as an auxiliary task to assist with Arabic diacritization. They tested their model using the ATB dataset and yielded a WER of 4.97%.

Masmoudi et al. [30] focused on diacritizing Tunisian Dialect (TD), and introduced a new corpus called TDTACHKIL and introduced an SMT-based model. In their experiment, they used the undiacritized text as the source and the diacritized text as the target. Their experiments yielded reasonably strong results with a DER of 8.89% and a WER of 16.7%. Masmoudi and Maalej suggested using a rule-based system to augment their system and were looking to further expand TDTACHKIL also.

Abandah et al. [31] took a transfer learning approach to improve the diacritization of Arabic poetry, which is commonly regarded as having higher error rates than standard Arabic text. The authors proposed a two-staged model with two classifiers, one for detecting poetry meter and the other for diacritizing the verse. Three transfer learning architectures were also explored by the authors featuring layers based on BiLSTM. The experiments showed that the transfer learning model with two stacked BiLSTM components performed best, with a DER of 3.54%. However, regarding all proposed systems, they were dependent on an existing poetry meter classifier, which is an important limitation and complexity to the construction of the systems.

Kharsa et al. [32] proposed Arabic diacritization system using BERT and provided favorable results on the Arabic Diacritization (AD) dataset, also known as the TB database. The results were so impressive that it shown the strength of transformer-based models for doing these types of tasks, achieving a syntactic Derivative Error Rate (DER) of 1.14% and a Word Error Rate (WER) of 3.34%, using the contextual possibilities of BERT.

Recent years have seen the rise of large-scale generative Arabic language models, such as Jais [33], ALLaM [34], and AceGPT [35]. These models perform well across many Arabic NLP tasks, including open-ended generation, question answering, and summarization. The shift towards these decoder-only architectures highlights the fast progress in Arabic language modeling. However, Arabic diacritization is quite different from the tasks these models are mainly built for. Diacritization is a structured token-level sequence labeling problem, where each character must get an accurate diacritic label under strict positional rules. Decoder-only generative LLMs focus on open-ended text generation, making them unsuitable for this type of structured prediction.

Note that the Jais model removes diacritics in its preprocessing pipeline, indicating that it does not model or restore diacritic information. Encoder-based architectures such as AraBERT are thus a more appropriate and coherent choice for diacritization since they are designed for token-level classification tasks and have shown competitive or better performance for structured sequence labelling objectives.

In general, despite many advances in Arabic diacritization using rule-based, statistical and deep learning techniques in the past, there remain many unanswered questions. Few efforts have exploited the potential of already trained transformer models specifically, via computationally light transfer learning pipelines, without additional modules.

Token level modelling, however, has not been studied much, though it has the potential for contextual disambiguation. From the observations of the previous writers, most of the proposed systems have been limited in terms of scalability, efficiency and generalisability due to their dependence on RNN based models and manually designed features. The observations emphasise the need for a lightweight transformer-based model that directly processes raw text and is built on token-level classification to enhance performance and applicability in modern Arabic diacritization.

To better capture sequential dependencies and improve diacritic prediction accuracy, we replace the standard linear classification head of AraBERT with more expressive bidirectional recurrent layers (BiLSTM and BiGRU). This modification introduces an explicit sequential inductive bias at the output level, which enables the model to better capture local label dependencies, that are not explicitly captured by a linear classifier.

Another approach would be to include a Conditional Random Field (CRF) layer to model global sequence consistency. However, in our formulation, the label space consists of 6,016 composite diacritic classes and hence would require a transition matrix of size $6,016 \times 6,016$. This makes the CRF approach computationally infeasible in terms of memory and training complexity.

Instead, we adopt label smoothing ($\epsilon = 0.1$) as a lightweight regularization technique that mitigates overconfidence and improves generalization across morphologically similar diacritic classes. While label smoothing does not explicitly model label transitions as CRFs do, it provides a practical and efficient alternative under large label spaces.

3. BACKGROUND

3.1. Encoder-Decoder Transformer

The recurrent neural networks (RNNs) and their variations like Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRUs) have been widely used for natural language processing (NLP) tasks. In these models, the inputs and outputs depend on the previous elements in the sequence. They process input

sequences sequentially. However, RNNs are challenging to capture long-term dependencies efficiently, since they work sequentially.

To tackle this issue, Vaswani et al. [36] suggested the Transformer architecture. Transformers leverage self-attention mechanisms to assign different weights to various parts of the input sequence. Unlike RNNs which rely on recent input and previous hidden states at each time step. So that the model can better learn the contextual relationship over the whole sequence.

Transformers do not have any notion of the order of the input sequences. Therefore, positional encodings are added to the input to provide information about the absolute or relative positions of tokens in the input.

In the encoder-decoder Transformer model the encoder receives the input sequence and generates a set of context-specific vector representations. The decoder takes these representations and produces the target output sequence one step at a time, attending to the most relevant parts of the encoded input.

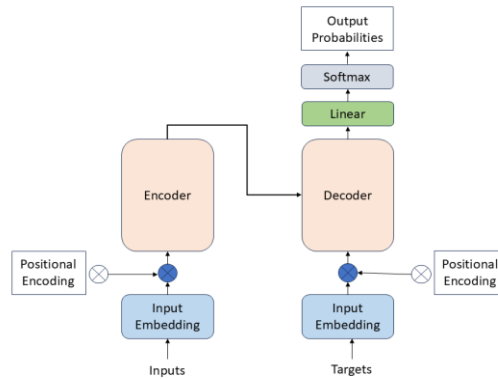


Figure 1. Architecture of the encoder–decoder Transformer model.

Both encoder and decoder components of the Transformer model are composed of multiple identical layers (typically six), each of which is comprised of two primary sub-layers: a multi-head self-attention mechanism and a position-wise feed-forward neural network. To enable learning and gradient propagation, residual connections [37] are put around each sub-layer, then followed by layer normalization. Each layer takes on a constant output dimensionality, which is denoted as d_m .

The multi-head self-attention mechanism enables the model to attend to all positions in the input sequence simultaneously, regardless of their distance. It is achieved by computing attention scores for word pairs through query (Q), key (K), and value (V) matrices. The attention is computed through the scaled dot-product attention equation:

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad (1)$$

Where:

- $Q \in \mathbb{R}^{n \times d_k}$: query matrix,
- $K \in \mathbb{R}^{n \times d_k}$: key matrix,
- $V \in \mathbb{R}^{n \times d_v}$: value matrix,
- d_k : dimensionality of key vectors,
- n : sequence length.

To allow the model to process jointly information from different representation subspaces, the Transformer employs multi-head attention, where h parallel attention heads are computed by using distinct learned linear projections:

$$head_i = Attention(QW_i^Q, KW_i^K, VW_i^V) \quad (2)$$

Where:

- $W_i^Q \in \mathbb{R}^{d_{m \times d_k}}$: projection matrix for the i -th query,
- $W_i^K \in \mathbb{R}^{d_{m \times d_k}}$: projection matrix for the i -th key,
- $W_i^V \in \mathbb{R}^{d_{m \times d_v}}$: projection matrix for the i -th value.

The output of each head is then concatenated and projected with a learnable matrix $W^O \in \mathbb{R}^{h d_v \times d_m}$, yielding the final output of the multi-head attention layer:

$$MultiHead(Q, K, V) = Concat(head_1, \dots, head_h)W^O \quad (3)$$

Where:

- $W^O \in \mathbb{R}^{h \times d_m}$: output projection matrix,
- h : number of attention heads.

To save computation, it is common to have $d_k = d_v = \frac{d_m}{h}$.

Following the attention mechanism, each layer has a position-wise feed-forward network (FFN), which acts independently on each position. It consists of two linear layers with a ReLU activation in between:

$$FFN(x) = (0, xW_1 + b_1) W_2 + b_2 \quad (4)$$

Where:

- $x \in \mathbb{R}^{1 \times d_m}$: input vector,
- $W_1 \in \mathbb{R}^{d_m \times d_{\{ff\}}}$, $W_2 \in \mathbb{R}^{d_{\{ff\}} \times d_m}$: weight matrices,
- $b_1 \in \mathbb{R}^{1 \times d_{\{ff\}}}$, $b_2 \in \mathbb{R}^{1 \times d_m}$: bias vectors,
- $d_{\{ff\}}$: hidden layer dimension of the feed-forward network.

3.2. Encoder-Decoder Transformer

BERT, proposed by [38], is the state-of-the-art language representation model that surpasses earlier unidirectional models based on processing all context of a text sequence. It employs a stack of identical Transformer encoder layers which process the input bidirectionally, both from left to right and right to left in parallel. BERT implements transfer learning, wherein it's pre-trained on huge amounts of unlabeled data and then fine-tuned on modest task-specific labeled sets. In pre-training, the model learns general-purpose parameters, whereas in fine-tuning, it begins with them and varies predominantly the output layer weights via supervised learning across the specific task.

Antoun et al. [39] pre-trained and created AraBERT, an Arabic variant of BERT, using extensive Arabic corpora. AraBERT was evaluated against multilingual BERT and other state-of-the-art models available today on several Arabic NLP tasks, including Sentiment Analysis (SA), Named Entity Recognition (NER), and Question Answering (QA), and outperformed most of the alternative systems. In another similar study, [40] explored how performance and prediction quality are affected by model size and architecture. Based on their observations, the "AraBERTv0.2-base" variant was selected in this research because it performed better on various benchmarks.

3.3 Recurrent Neural Network

Recurrent Neural Networks (RNNs) have been widely applied to sequence modeling in natural language processing (NLP) applications. Vanilla RNNs suffer from vanishing and exploding gradient problems and thus cannot learn long-term dependencies. To mitigate this limitation, Long Short-Term Memory (LSTM) networks were introduced by [41]. LSTMs possess memory cells and gating mechanisms forget, input, and output gates which regulate the flow of information to allow the model to recall important information over lengthy sequences.

Later on, [42] suggested Gated Recurrent Units (GRUs) as a simpler alternative to LSTMs. GRUs merge the forget and input gates into a single update gate and the cell state and hidden state, thereby reducing the number of parameters with comparable performance. Both GRU and LSTM models have been at the core of most NLP tasks, including machine translation, speech recognition, and Arabic diacritization.

4. RESEARCH METHOD

4.1 Dataset

The Tashkeela corpus is an open-source dataset, which is widely used in Classical Arabic (CA) studies [21]. It comprises about 75 million tokens and is mainly based on Islamic literature. However, the corpus has some limitations, most of the texts do not have diacritics, and it does not have a standard preprocessing pipeline or a correct division into training, validation and test sets. These limitations make it unsuitable as a reference point in comparative studies.

To solve these problems, Abbad and Xiong [26] proposed the TP dataset by randomly sampling a subset of the fully diacritized files of the Tashkeela corpus. They performed extensive preprocessing steps such as removal of English characters and markup tags, and line segmentation. The developed dataset was divided into training, validation and testing subsets to provide a consistent evaluation and a fair comparison of Arabic diacritization models. In this work, training and evaluation are performed on the TP dataset.

4.1.1 Pre-processing the Training Data

The raw Tashkeela-derived files were normalized and segmented prior to token-label generation, which included the following steps:

- Lines longer than 1,000 characters were segmented at punctuation marks such as . : () { } [].
- If none of these were present, lines were divided into fixed chunks of 500 characters.
- This maintained sequence lengths manageable while having sufficient context information for effective modelling.
- Elongation mark (tatweel "-") was removed to remove redundant graphemes.
- No punctuation was used. While punctuation could theoretically help the inflectional analysis, the empirical results showed that removing punctuation increased the overall diacritization accuracy.
- Superscript indicators like the dagger alif were removed as they are not very common in classical Arabic texts and can add noise.
- Non-Arabic characters, including Latin letters and digits, were entirely removed.
- Each Arabic character may carry zero, one, or two diacritics.
- To obtain a one-to-one mapping between characters and labels, the diacritics attached to a character were collapsed into a single code, as defined in Table 2.
- This process yields an unambiguous label set of 15 symbols:
 - 8 single-diacritic values
 - 7 shadda combinations
 - Plus C for “no diacritic”

Table 2. Diacritic Code Mapping

Diacritic(s)	Code
None	C
Shadda	S
Sukūn	o
Kasra	k
Damma	d
Fathā	f
Tanwīn-Kasr	R
Tanwīn-Damm	M
Tanwīn-Fath	H
Shadda + Kasra	K
Shadda + Damma	D
Shadda + Fathā	F
Shadda + Tanwīn-Kasr	R
Shadda + Tanwīn-Damm	M
Shadda + Tanwīn-Fath	H

4.2 Tokenization and Label Alignment to Tokens

To pre-process the data for training, each diacritized Arabic sentence is normalized and tokenized with a pretrained BERT tokenizer (aubmindlab/bert-base-arabertv02). The tokenizer splits the sentence into subword units, and this allows the model to handle rare or complex words effectively.

Each token is then converted into a sequence of diacritic labels. Since no Arabic character can contain more than two diacritics, they are first consolidated into one code by a pre-specified mapping to simplify labeling. A label of the form position:code is attached to each character of a token, where position is the position of the character in the token and code is the corresponding diacritic code. If a token has no diacritics, it is marked as {SAME} to reflect no change in diacritic.

Additional special tokens, such as [CLS] and [SEP], are prefixed and suffixed to the beginning and end of the token sequence to match the formatting of BERT's input, and padding tokens, [PAD], are suffixed to ensure an appropriate fixed length for the sequence.

Suppose one takes the diacritized word “كُتِبَ” with its undiacritized version “كتب”. The BERT tokenizer puts forth the token:

كُتِبَ

For each token character, the diacritic tag is the same as the fatha diacritic (denoted as f), which produces the label sequence:

0:f;1:f;2:f

Here, each digit corresponds to the character index of the token and the letter denotes the reduced diacritic code. This mapping enables the model to predict diacritic patterns at the token level, taking advantage of the tokenizer's subword segmentation to preserve morphological structure.

Importantly, Shadda (consonant doubling) and its combinations with vowel marks are handled explicitly within this labeling scheme. Each Shadda–vowel combination (e.g., Shadda+Fathā, Shadda+Damma, Shadda+Kasra, and their tanwīn variants) is represented as a single unified label, as shown in Table 2. This design ensures that such combinations are predicted as atomic units, preventing partial prediction errors where either the Shadda or the accompanying vowel mark is predicted in isolation.

4.3 Evaluation Metrics

We assess diacritization performance using two standard evaluation metrics: Diacritization Error Rate (DER) and Word Error Rate (WER).

DER measures the proportion of individual characters that have been incorrectly diacritized and is calculated as follows:

$$DER = \frac{\text{Number of incorrectly diacritized characters}}{\text{Total number of characters}} \quad (5)$$

WER, on the other hand, computes the ratio of words that contain at least one diacritization error:

$$WER = \frac{\text{Number of incorrectly diacritized words}}{\text{Total number of words}} \quad (6)$$

4.4 BiLSTM and BiGRU as Transformer Heads

Even though the Transformer encoder gives powerful contextual representations using self-attention, it may be useful to augment these representations with some sequential modeling. Therefore, we try to add bidirectional recurrent neural networks, specifically BiLSTM and BiGRU layers, to the output of Transformer encoders. To further improve training stability and reduce overconfidence in class predictions across the large label space (6,016 classes), label smoothing with $\varepsilon = 0.1$ is applied during training. This serves as a lightweight regularization mechanism in place of a Conditional Random Field (CRF) layer, which would be computationally prohibitive at this label-space scale.

This design choice is motivated from the complementary advantages of Transformers and recurrent architectures. Transformer models perform well at capturing long-range dependencies and global context, while recurrent layers such as BiLSTM and BiGRU are good at local pattern representation; this is important for character level relations in Arabic and detailed morphological attributes.

In our architecture, either a BiLSTM or a BiGRU layer receives as input token embeddings produced by the pre-trained Transformer encoder (i.e., BERT). These layers act as heads for specific tasks that are designed to map contextual embeddings to sequences of diacritic labels. We evaluate both BiLSTM and BiGRU to identify which one better adapts the Transformer's representation for the diacritization task.

Let the Transformer encoder takes as input a sequence of tokens:

$$X = [x_1, x_2, \dots, x_T] \quad (7)$$

and produce contextual embeddings:

$$H_{BR} = BERT(X) = [h_1, h_2, \dots, h_T], \quad h_t \in R^d \quad (8)$$

Where:

- x_t : input token at time step t
- T : sequence length
- h_t : contextual embedding of token x_t
- d : embedding dimension

4.4.1 BiLSTM Head

The BiLSTM processes the BERT embeddings as follows:

$$\vec{h}_t^{LSTM} = LSTM_{fwd}(h_t) \quad (9)$$

$$\overleftarrow{h}_t^{LSTM} = LSTM_{bwd}(h_t) \quad (10)$$

$$\tilde{h}_t = [\vec{h}_t^{LSTM}; \overleftarrow{h}_t^{LSTM}] \quad (11)$$

$$y_t = Softmax(W\tilde{h}_t + b) \quad (12)$$

Where:

- $\vec{h}_t^{LSTM}, \overleftarrow{h}_t^{LSTM}$: forward and backward LSTM outputs
- $\tilde{h}_t$: concatenated BiLSTM representation at time step t
- W, b : weight matrix and bias vector for the classification layer
- y_t : predicted probability distribution over diacritic labels

The forward and backward LSTM outputs are concatenated into $\tilde{h}_t$, which is then passed through a linear layer with weights W and bias b . The softmax function outputs the predicted diacritic distribution y_t .

4.4.2 BiGRU Head

Similarly, the BiGRU processes the BERT embeddings as:

$$\vec{h}_t^{GRU} = GRU_{fwd}(h_t) \quad (13)$$

$$h_t^{GRU} = GRU_{bwd}(h_t) \quad (14)$$

$$\tilde{h}_t = [\vec{h}_t^{GRU}; \overleftarrow{h}_t^{GRU}] \quad (15)$$

$$y_t = Softmax(W\tilde{h}_t + b) \quad (16)$$

Where:

- $\vec{h}_t^{GRU}, \overleftarrow{h}_t^{GRU}$: forward and backward GRU outputs
- $\tilde{h}_t$: concatenated BiGRU representation at time step t
- y_t : predicted diacritic distribution

The GRU works similarly to the LSTM, but with fewer gates. Forward and backward hidden states are concatenated and passed through a linear + softmax layer to generate label probabilities.

5. RESULTS AND DISCUSSION

5.1 Overall Performance Comparison

We evaluate our diacritization models using two standard metrics: Diacritic Error Rate (DER) and Word Error Rate (WER). Table 3 shows the performance comparison between our proposed models and previous work. To ensure methodological soundness, comparisons are restricted to models evaluated on the same Tashkeela Processed (TP) benchmark with identical data splits. Direct comparison across systems using different preprocessing pipelines or data partitions may yield misleading conclusions and is therefore excluded.

Table 3 Comparison of DER and WER with Previous Works

Model	DER (%)	WER (%)
Abbad et al. [27]	2.45	5.42
Kharsa et al. [32]	1.14	3.34
Our Model (Transformer + BiGRU)	1.84	2.62
Our Model (Transformer + BiLSTM)	1.82	2.58

Both our Transformer + BiGRU and our Transformer + BiLSTM models outperform the Abbad et al. [27] system with improved DER and WER metrics. This is proof of the robustness of combining context-rich embeddings from a pre-trained Transformer with fine-tuning with bidirectional recurrence.

Compared to Kharsa et al.'s system [32], both our BiLSTM and BiGRU models surpass their WER of 3.34%, achieving WERs of 2.58% and 2.62%, respectively. Furthermore, our BiLSTM model achieves a DER of 1.82%, which is competitive with their reported DER of 1.14% given the architectural simplicity of the proposed approach. These results demonstrate that introducing sequential modeling on top of transformer representations yields a clear and measurable improvement at the word level.

Importantly, this comparison constitutes a controlled ablation over the prediction head: both systems share the same AraBERT backbone and are evaluated on the same benchmark, with the only architectural difference being the replacement of the linear classification head with a bidirectional recurrent layer. The consistent gains across both BiLSTM and BiGRU variants confirm that sequential modeling at the output level is both sufficient and effective for this task, without the added complexity of autoregressive decoding.

Among our models, the Transformer + BiLSTM model achieves the best overall performance with a DER of 1.82% and a WER of 2.58%, outperforming the BiGRU variant which records a DER of 1.84% and a WER of 2.62%. This is consistent with the notion that BiLSTM cells are better suited to encoding Arabic character sequences in diacritization due to their capacity to learn deep morpho-phonetic structures of the language.

5.2 Inference Time Analysis

To add a practical touch to the accuracy metrics, we performed a dedicated inference time analysis on the full test set on GPU hardware. Table 5 shows the average inference time per sentence for the two proposed models. The BiGRU model has about 8.74 ms per sentence while the BiLSTM model has about 9.22 ms per sentence. Both values are well below the commonly accepted 300 ms latency threshold for interactive voice AI applications [43], suggesting that the proposed architecture is suitable for near real-time deployment scenarios such as Arabic text-to-speech (TTS) and educational systems. Therefore, these results indicate that diacritization with the proposed approach is a rather small part of the whole processing pipeline in these applications.

Table 5. Average Inference Time per Sentence (GPU)

Model	Avg. Inference Time (ms/sentence)
Transformer + BiLSTM	9.22
Transformer + BiGRU	8.74

6. ERROR ANALYSIS

6.1 Wrong Diacritic Predictions

The most common error in both systems is of confusing one diacritic with another. Figure 2 shows the frequency of the five most common substitution errors.

The most common substitution error is Damma predicted as Fatha (BiLSTM : 624, BiGRU : 610) followed by Fatha predicted as Damma (BiLSTM : 516, BiGRU : 540) in both models. Kasra↔Fatha confusions are also prominent (Kasra→Fatha: BiLSTM: 430, BiGRU: 442; Fatha→Kasra: BiLSTM: 338, BiGRU: 343), with Kasra→Damma being the least frequent among the top five errors (BiLSTM: 206, BiGRU: 223). In general, BiLSTM makes fewer substitution errors in most categories, indicating a slightly better ability to discriminate short vowels in context.

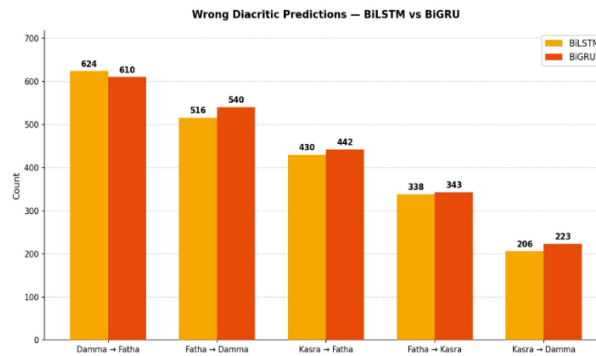


Figure 2. Counts of common wrong diacritic predictions

In general, the models are confusing *fatha* with both *damma* and *kasra*, such that these short vowels are difficult to distinguish under real conditions. This is likely due to the similarity in appearance and sound of the marks, and also their tendency to appear close to each other in words and in writing. The slightly higher number of mistakes in BiGRU suggests that this architecture has it more difficult to distinguish these vowels from each other compared to BiLSTM.

6.2 Missing Diacritic Predictions

Another significant class of errors relates to the omission of diacritic marks where they are expected. Figure 3 shows the number of diacritics missed by each model.

The Most Missed Category is No Diacritic (BiLSTM: 23, BiGRU: 24) and then Kasra (BiLSTM: 23, BiGRU: 19). BiLSTM has much less Fatha omissions (11) than BiGRU (16), which shows BiLSTM preserves fatha predictions better. The number of Damma omissions is the same in both models (3 each), whereas Sukoon is the least missed, with BiLSTM missing it only once, compared to three times for BiGRU.

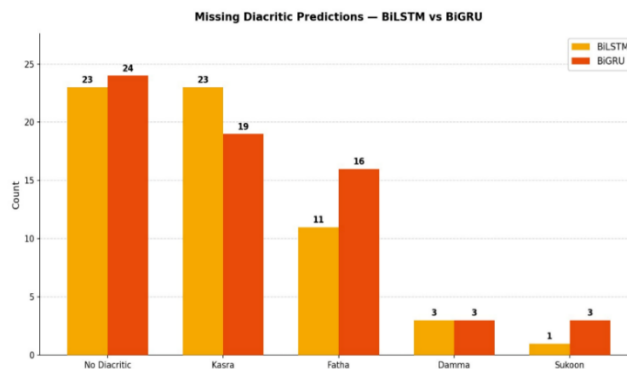


Figure 3. Counts of missing diacritic predictions

The main omission categories are No Diacritic and Kasra, and the Fatha omissions are relatively lower, especially in BiLSTM. Both systems also sometimes produce cases that are tagged as having no diacritic at all. In Arabic, where small marks can alter pronunciation or meaning, these omissions can mislead readers and make texts much harder to read.

6.3 Shadda-Specific Error Analysis

We conducted a dedicated error analysis on the model outputs to study how the Shadda-vowel combinations are handled. The five most common prediction errors for Shadda for both BiLSTM and BiGRU are shown in Figure 4. The most dominant error pattern is the confusion of Shadda+Fatha with plain Fatha (BiLSTM: 70, BiGRU: 68), which suggests that the model sometimes predicts only the vowel component and drops the Shadda. The reverse confusion, predicted by Fatha as Shadda+Fatha, is also common (BiLSTM: 64, BiGRU: 56). Other errors are confusions between related Shadda-vowel pairs, such as Shadda+Fatha predicted as Shadda+Kasra (BiLSTM: 54, BiGRU: 57) and Shadda alone predicted as Shadda+Fatha (BiLSTM: 53, BiGRU: 48). These results show that the model is able to detect the presence of Shadda, but struggles to distinguish between fine-grained Shadda-vowel combinations, in line with the general ambiguity of short vowels discussed in Section 6.1.

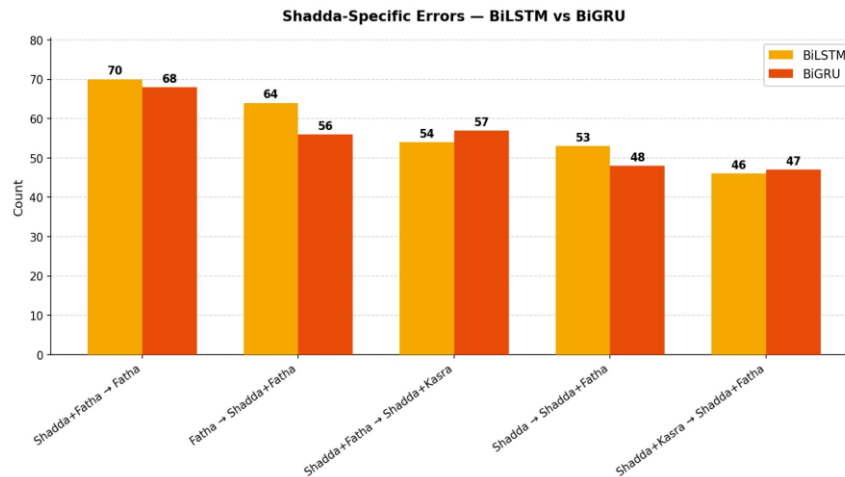


Figure 4. Counts of Shadda-specific diacritic prediction errors

6.4 Grammatical Structure Analysis

In addition to overall error tallies, a focused investigation of short-vowel confusions shows a systematic association between prediction errors and particular grammatical structures. The most frequent error pattern is the confusion between fatha and damma in the initial character position of verbs (pos=0), which directly corresponds to the morphological marking of voice in Arabic. In the active voice, the verb stem typically begins with fatha (e.g., يَمْنَعُ, "he prevents"), whereas the passive voice is marked by damma at the same position (e.g., يُمْنَعُ, "it is prevented"). Because this distinction is signaled solely by an internal vowel alternation, without any overt syntactic marker, it represents one of the most challenging disambiguation tasks for token-level classifiers.

Table 4 quantifies these voice-related errors across all three models. Passive-to-active misclassification accounts for 133, 135, and 127 errors in the BiLSTM, BiGRU, and Baseline models respectively (approximately 25% of all verb-related errors), while active-to-passive errors contribute 84, 86, and 89 instances respectively (approximately 16%). The consistency of these figures across architectures indicates a systematic linguistic challenge rather than random noise, all models struggle equally with voice disambiguation, suggesting the difficulty is rooted in Arabic morphological ambiguity rather than any specific architectural limitation.

Table 4. Voice-Related Diacritic Errors (fatha ↔ damma at verb-initial position)

Error Type	BiLSTM	BiGRU
Passive→Active (damma→fatha)	133	135
Active→Passive (fatha→damma)	84	86

The results are illustrated with concrete examples from the model outputs. The passive verb يُطْلَقُ ("is launched") is always confused with the active تَطْلُقُ ("launches"). The active يَمْنَعُ ("prevents") is often predicted as the passive يُمْنَعُ ("is prevented"). These errors are due to the inherent difficulty of the passive pattern of the yu'fal where voice is marked only through internal vowel alternations with no overt syntactic marker.

7. CONCLUSION

Our own analysis indicates that while both BiLSTM and BiGRU models are good at identifying common diacritic patterns, distinguishing similar short vowels such as fatha, damma, and kasra is a serious

issue. This ambiguity points towards the need for improved context representation or the enforcement of linguistic or phonetic constraints in learning.

The other very important area that needs model improvement is omission of diacritical markings, especially fatha. Avoiding such errors will become significant for downstream practical applications in real life such as text-to-speech, education software, and Arabic-focused NLP models since omission of diacritics may cause ambiguity in meaning as well as pronunciation.

In general, performance on Arabic diacritization is significantly enhanced by inserting a BiLSTM or BiGRU layer as a sequential head following the Transformer encoder. Both architectures perform well, with the BiLSTM version consistently returning lower Diacritic Error Rates (DER) and Word Error Rates (WER). These findings are meant to further confirm that local sequential modeling based on global contextual representation is very effective for challenging morphological tasks like Arabic diacritization.

8. FUTURE WORK

The present experiments demonstrate the effectiveness of augmenting the transformer encoders with sequential modelling layers for Arabic diacritization, but many promising directions are left for future work. First, adding explicit linguistic and phonological data could further improve performance. Features like syllable structure, root-pattern morphology, and phonetic representations can potentially help the model better disambiguate confusion between similar short vowels.

Secondly, our error analysis shows that some diacritics, especially fatha, are more often omitted. This indicates the potential benefit of task-specific optimisation strategies, such as designing loss functions that heavily penalise omission errors, or introducing auxiliary objectives (e.g. predicting the presence of diacritics) to guide the learning process.

Third, extending the framework to a multi-task learning (MTL) setting is a promising direction. Joint modelling of diacritization and part-of-speech (POS) tagging could provide more fine-grained syntactic supervision, since diacritics in Arabic are strongly associated with grammatical structure and word role. Furthermore, testing the proposed approach on larger and more diverse diacritized corpora may improve generalisation and context understanding. Another natural extension is to explore transfer learning strategies, e.g., fine-tuning multilingual transformer models with task-specific prediction heads.

Finally, scaling the model to the zero-resource, low-resource and dialectal arabic setting is still an important challenge. The proposed architecture combining contextual encoding and sequential modelling offers a good foundation for domain adaptation and semi-supervised learning for cases with limited labelled data.

From the application perspective, deploying these models in real world systems, e.g., educational tools or text-to-speech pipelines, poses additional challenges, such as real time inference constraints, robustness to noisy input, and handling dialectal variation. These factors will need to be addressed for practical adoption.

REFERENCES

- [1] Y. A. El-Imam, "Phonetization of Arabic: rules and algorithms," *Comput. Speech Lang.*, vol. 18, no. 4, pp. 339–373, 2004. [https://doi.org/10.1016/S0885-2308\(03\)00035-4](https://doi.org/10.1016/S0885-2308(03)00035-4)
- [2] K. Shaalan and Khaled, "Rule-based Approach in Arabic Natural Language Processing," *International Journal on Information and Communication Technologies (IJICT)*, vol. 3, pp. 11–, Jun. 2010. https://doi.org/10.1142/9789813229396_0003
- [3] I. Zitouni and R. Sarikaya, "Arabic diacritic restoration approach based on maximum entropy models," *Computer Speech & Language*, vol. 23, no. 3, pp. 257–276, 2009. <https://doi.org/10.1016/j.csl.2008.06.001>
- [4] H. El Rifai, L. Al Qadi, and A. Elnagar, "Arabic text classification: the need for multi-labeling systems," *Neural Comput. Appl.*, vol. 34, no. 2, pp. 1135–1159, 2022. <https://doi.org/10.1007/s00521-021-06390-z>
- [5] A. Elnagar, S. M. Yagi, A. B. Nassif, I. Shahin, and S. A. Salloum, "Systematic literature review of dialectal Arabic: identification and detection," *IEEE Access*, vol. 9, pp. 31010–31042, 2021. <https://doi.org/10.1109/ACCESS.2021.3059504>
- [6] A. B. Nassif, A. M. Darya, and A. Elnagar, "Empirical evaluation of shallow and deep learning classifiers for Arabic sentiment analysis," *ACM Trans. Asian Low-Resour. Lang. Inf. Process.*, vol. 21, no. 1, pp. 1–25, Nov. 2021. <https://doi.org/10.48550/arXiv.2112.00534>
- [7] A. B. Nassif, A. Elnagar, O. Elgendy, and Y. Afadar, "Arabic fake news detection based on deep contextualized embedding models," *Neural Comput. Appl.*, vol. 34, no. 18, pp. 16019–16032, May 2022. <https://doi.org/10.48550/arXiv.2205.03114>
- [8] M. Lataifeh, A. Elnagar, I. Shahin, and A. B. Nassif, "Arabic audio clips: Identification and discrimination of authentic Cantillations from imitations," *Neurocomputing*, vol. 418, pp. 162–177, 2020. <https://doi.org/10.1016/j.neucom.2020.07.099>

- [9] A. B. Nassif, I. Shahin, A. Elnagar, D. Velayudhan, A. Alhudhaif, and K. Polat, "Emotional speaker identification using a novel capsule nets model," *Expert Syst. Appl.*, vol. 193, p. 116469, May 2022. <https://doi.org/10.48550/arXiv.2201.02994>
- [10] I. Shahin, A. B. Nassif, N. Nemmour, A. Elnagar, A. Alhudhaif, and K. Polat, "Novel hybrid DNN approaches for speaker verification in emotional and stressful talking environments," *Neural Computing and Applications*, vol. 33, no. 23, pp. 16033–16055, Jun. 2021. <https://doi.org/10.48550/arXiv.2112.13353>
- [11] A. Al Sallab, M. Rashwan, H. M. Raafat, and A. Rafea, "Automatic Arabic diacritics restoration based on deep nets," in *Proceedings of the EMNLP 2014 Workshop on Arabic Natural Language Processing (ANLP)*, Doha, Qatar, pp. 65–72, 2014. <https://doi.org/10.3115/v1/W14-3608>
- [12] M. Maamouri, A. Bies, T. Buckwalter, and W. Mekki, "The Penn Arabic Treebank: Building a Large-Scale Annotated Arabic Corpus," 2004. <https://api.semanticscholar.org/CorpusID:16205731>
- [13] M. Diab, N. Habash, O. Rambow, and R. Roth, "LDC Arabic Treebanks and associated corpora: Data divisions manual," *arXiv preprint arXiv:1309.5652*, 2013. <https://doi.org/10.48550/arXiv.1309.5652>
- [14] M. Rashwan, M. Elbadrashiny, M. Attia, and S. Abdou, "A hybrid system for automatic Arabic diacritization," 2009.
- [15] 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 Lang. Process.*, vol. 19, no. 1, pp. 166–175, Jan. 2011. <https://doi.org/10.1109/TASL.2010.2045240>
- [16] G. A. Abandah, A. Graves, B. Al-Shagoor, A. Arabiyat, F. Jamour, and M. Al-Tae, "Automatic diacritization of Arabic text using recurrent neural networks," *Int. J. Doc. Anal. Recognit.*, vol. 18, no. 2, pp. 183–197, Jun. 2015. <https://doi.org/10.1007/s10032-015-0242-2>
- [17] R. Alnefaie and A. M. Azmi, "Automatic minimal diacritization of Arabic texts," *Procedia Computer Science*, vol. 117, pp. 169–174, 2017. <https://doi.org/10.1016/j.procs.2017.10.106>
- [18] K. Darwish, H. Mubarak, and A. Abdelali, "Arabic Diacritization: Stats, Rules, and Hacks," in *Proceedings of the Third Arabic Natural Language Processing Workshop*, Valencia, Spain, Apr. 2017, pp. 9–17. <https://doi.org/10.18653/v1/W17-1302>
- [19] H. Mubarak, A. Abdelali, H. Sajjad, Y. Samih, and K. Darwish, "Highly Effective Arabic Diacritization using Sequence to Sequence Modeling," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, Minneapolis, Minnesota, pp. 2390–2395, 2019. <https://doi.org/10.18653/v1/N19-1248>
- [20] A. Fadel, I. Tuffaha, B. Al-Jawarneh, and M. Al-Ayyoub, "Arabic Text Diacritization Using Deep Neural Networks," *arXiv preprint arXiv:1905.01965*, 2019. <https://doi.org/10.48550/arXiv.1905.01965>
- [21] T. Zerrouki and A. Balla, "Tashkeela: Novel corpus of Arabic vocalized texts, data for auto-diacritization systems," *Data in Brief*, vol. 11, pp. 147–151, 2017. <https://doi.org/10.1016/j.dib.2017.01.011>
- [22] Z. Barqawi, "Shakkala, Arabic text vocalization," 2017. [Online]. Available: <https://github.com/Barqawiz/Shakkala>
- [23] A. Fadel, I. Tuffaha, B. Al-Jawarneh, and M. Al-Ayyoub, "Neural Arabic text diacritization: state of the art results and a novel approach for machine translation," in *Proc. 6th Workshop Asian Translation, Association for Computational Linguistics*, 2019, pp. 215–225. <https://doi.org/10.48550/arXiv.1911.03531>
- [24] K. Darwish, A. Abdelali, H. Mubarak, and M. Eldesouki, "Arabic diacritic recovery using a feature-rich biLSTM model," *arXiv preprint arXiv:2002.01207*, 2020. <https://doi.org/10.48550/arXiv.2002.01207>
- [25] B. AlKhamissi, M. N. ElNokrashy, and M. Gabr, "Deep Diacritization: Efficient Hierarchical Recurrence for Improved Arabic Diacritization," *arXiv preprint arXiv:2011.00538*, 2020. <https://doi.org/10.48550/arXiv.2011.00538>
- [26] H. Abbad and S. Xiong, "Multi-components System for Automatic Arabic Diacritization," in *Advances in Information Retrieval, Cham: Springer International Publishing*, pp. 341–355, 2020. https://doi.org/10.1007/978-3-030-45439-5_23
- [27] H. Abbad and S. Xiong, "Simple extensible deep learning model for automatic Arabic diacritization," *ACM Trans. Asian Low-Resour. Lang. Inf. Process.*, vol. 21, no. 2, pp. 1–16, Nov. 2021. <https://doi.org/10.1145/3480938>
- [28] Y. Hifny, "Recent Advances in Arabic Syntactic Diacritics Restoration," in *2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 7768–7772, 2021. <https://doi.org/10.1109/ICASSP39728.2021.9414500>
- [29] B. Thompson and A. Alshehri, "Improving Arabic Diacritization by Learning to Diacritize and Translate," *arXiv preprint arXiv:2109.14150*, 2021. <https://doi.org/10.48550/arXiv.2109.14150>
- [30] A. Masmoudi, C. Aloulou, A. G. S. Abdellahi, and L. H. Belguith, "Automatic diacritization of Tunisian dialect text using SMT model," *Int. J. Speech Technol.*, vol. 25, no. 1, pp. 89–104, 2022. <https://doi.org/10.1007/s10772-021-09864-6>

- [31] G. A. Abandah, A. E. Suyyagh, and M. R. Abdel-Majeed, "Transfer learning and multi-phase training for accurate diacritization of Arabic poetry," *J. King Saud Univ. - Comput. Inf. Sci.*, vol. 34, no. 6, pt. B, pp. 3744–3757, 2022. <https://doi.org/10.1016/j.jksuci.2022.04.005>
- [32] R. Kharsa, A. Elnagar, and S. Yagi, "BERT-based Arabic diacritization: A state-of-the-art approach for improving text accuracy and pronunciation," *Expert Syst. Appl.*, vol. 248, p. 123416, 2024. <https://doi.org/10.1016/j.eswa.2024.123416>
- [33] N. Sengupta et al., "Jais and Jais-chat: Arabic-centric foundation and instruction-tuned open generative large language models," arXiv preprint arXiv:2308.16149, 2023. <https://doi.org/10.48550/arXiv.2308.16149>
- [34] M. Bari et al., "ALLaM: Large Arabic language models," arXiv preprint arXiv:2407.15390, 2024. <https://doi.org/10.48550/arXiv.2407.15390>
- [35] D. Huang et al., "AceGPT, localizing large language models in Arabic," arXiv preprint arXiv:2309.12053, 2023. <https://doi.org/10.48550/arXiv.2309.12053>
- [36] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention Is All You Need," *arXiv preprint arXiv:1706.03762*, 2023. <https://doi.org/10.48550/arXiv.1706.03762>
- [37] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," *arXiv preprint arXiv:1512.03385*, 2015. <https://doi.org/10.48550/arXiv.1512.03385>
- [38] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2019. <https://doi.org/10.48550/arXiv.1810.04805>
- [39] W. Antoun, F. Baly, and H. Hajj, "AraBERT: Transformer-based model for Arabic language understanding," *arXiv preprint arXiv:2003.00104*, 2021. <https://doi.org/10.48550/arXiv.2003.00104>
- [40] G. Inoue, B. Alhafni, N. Baimukan, H. Bouamor, and N. Habash, "The interplay of variant, size, and task type in Arabic pre-trained language models," *arXiv preprint arXiv:2103.06678*, 2021. <https://doi.org/10.48550/arXiv.2103.06678>
- [41] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997. <https://doi.org/10.1162/neco.1997.9.8.1735>
- [42] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation," *arXiv preprint arXiv:1406.1078*, 2014. <https://doi.org/10.48550/arXiv.1406.1078>
- [43] AssemblyAI, "The 300ms rule: Why latency makes or breaks voice AI applications," 2025. [Online]. Available: <https://www.assemblyai.com/blog/low-latency-voice-ai>

BIOGRAPHY OF AUTHORS



Ahmed Abdelhamed Elgayar is a Senior BI & Data Analytics Engineer with over five years of professional experience in business intelligence, data analytics, and data visualization, along with a broader background in civil engineering and project controls. He holds a Master's degree in Data Science and a Postgraduate Diploma in Data Science with distinction from the Faculty of Graduate Studies for Statistical Research, Cairo University. His expertise includes Power BI, SQL, Python, ETL development, data warehousing, and the Microsoft BI stack (SSIS, SSAS, and SSRS). He has contributed to large-scale business intelligence projects across governmental and private sectors in the United Arab Emirates, Saudi Arabia, and Egypt. His research interests include Arabic natural language processing (NLP), deep learning, machine learning, and AI-based language processing systems. He can be contacted at : a.elgayar17@gmail.com.



Abdul-Hadi Nabi Ahmed holds a Ph.D. and an M.Sc. in Statistics (Quantitative Methods) from Florida State University, USA, and an M.Sc. in Mathematical Statistics from Cairo University, Egypt. He also earned diplomas in Applied Statistics and Operations Research, and a B.Sc. in Communication Engineering. He has taught undergraduate and graduate courses in statistics, operations research, mathematics, and economics at institutions including Florida State University, Cairo University, King Saud University, King Abdulaziz University, and the American University in Cairo. He has supervised over 45 graduate theses, including 10 Ph.D. dissertations. His research interests include statistical modeling, stochastic processes, and quantitative methods in economics and engineering. He can be contacted at: dr.hadi@cu.edu.eg.



Basma Ezzat El-Demerdash is an associate professor at the Department of Operations Research and Management, Faculty of Graduate Studies for Statistical Research (FGSSR), Cairo University. She graduated with a B.Sc. in Operations Research and Decision Support from the Faculty of Computers and Information, Cairo University in 2007. Her outstanding academic performance earned her a position as a teaching assistant in the department. She then went on to receive her M.Sc. degree from the same department in 2011. Her thesis focused on evaluating higher education systems in Egypt using data envelopment analysis. Her academic career began as a teaching assistant in 2007. Since then, she has accumulated 17 years of teaching experience at the Faculty of Computers and Artificial Intelligence, Cairo University. She can be contacted at: basma.ezzat@cu.edu.eg.



Ahmed E. Soliman is a Senior Team Lead Software Engineer and Data Scientist with extensive expertise in business process automation, data engineering, and deep learning. He holds a Master's degree in Data Science from Cairo University. His research focus lies at the intersection of Natural Language Processing (NLP) and Large Language Models (LLMs), with specific contributions in Few-Shot Learning and Data Augmentation techniques. Throughout his career, Ahmed has led and contributed to numerous high-impact projects across government and private sectors in Egypt, Saudi Arabia, and the UAE. His technical proficiency spans Python-based machine learning, Deep Learning, neural networks, and advanced data visualization. He can be contacted at: aelsaeid.soliman@gmail.com.