IAES International Journal of Artificial Intelligence (IJ-AI)
Vol. 14, No. 1, February 2025, pp. 702~711
ISSN: 2252-8938, DOI: 10.11591/ijai.v14.i1.pp702-711

Arabic text diacritization using transformers: a comparative study

Iman Zubeiri, Adnan Souri, Badr Eddine El Mohajir
New Technology Trends for Innovation Team, Faculty of Sciences, Abdelmalek Essaadi University, Tetouan, Morocco

Article Info

Article history:

Received Nov 28, 2023
Revised Aug 22, 2024
Accepted Aug 30, 2024

Keywords:

AraBERT
Arabic text diacritization
Bidirectional long short-term memory
Natural language processing
Transformers

ABSTRACT

The Arabic language presents challenges for natural language processing (NLP) tasks. One such challenge is diacritization, which involves adding diacritical marks to Arabic text to enhance readability and disambiguation. Diacritics play a crucial role in determining the correct pronunciation, meaning, and grammatical structure of words and sentences. However, Arabic texts are often written without diacritics, making NLP tasks more complex. This study investigates the efficacy of advanced machine learning models in automatic Arabic text diacritization, with a concentrated focus on the Arabic bidirectional encoder representations from transformers (AraBERT) and bidirectional long short-term memory (Bi-LSTM) models. AraBERT, a bidirectional encoder representation from transformers (BERT) derivative, leverages the transformer architecture to exploit contextual subtleties and discern linguistic patterns within a substantial corpus. Our comprehensive evaluation benchmarks the performance of these models, revealing that AraBERT significantly outperforms the Bi-LSTM with a diacritic error rate (DER) of only 0.81% and an accuracy rate of 98.15%, against the Bi-LSTM's DER of 1.02% and accuracy of 93.88%. The study also explores various optimization strategies to amplify model performance, setting a precedent for future research to enhance Arabic diacritization and contribute to the advancement of Arabic NLP.

Corresponding Author:

Zubeiri Iman
New Technology Trends for Innovation Team, Faculty of Sciences, Abdelmalek Essaadi University
Av. Khenifra, Tétouan 93000, Morocco
Email: imane.zubeiri@gmail.com

1. INTRODUCTION

Arabic is a widely spoken language, with over 467 million speakers. It ranks as the fourth most spoken language in the world and serves as the official language of 26 countries. Arabic is distinctive for its right-to-left script, which comprises 28 letters as shown in Table 1 that change form depending on their position within a word. Beyond the script, Arabic dialects, from the Maghreb's sunny expanses to the Mashreq's lively essence, offer a rich auditory palette. While modern standard Arabic (MSA) provides a scholarly and literary standard, the everyday dialects weave through conversations, carrying the heartbeats of cultural heritage and daily life's vibrancy.

Beneath the melody of Arabic words lies a hidden code: diacritics, or tashkeel [1]. These tiny symbols, guide the pronunciation and grammar, ensuring clarity and unlocking the true meaning of the word or the sentence. Without them, the language risks becoming ambiguous and not clear even for natives. In the realm of natural language processing (NLP), a subfield of artificial intelligence focused on the interaction between computers and human languages, the task of adding diacritical marks to text or diacritization is fundamental for processing the Arabic language. Diacritization not only aids in clarifying meanings to improve reading comprehension but also is pivotal for applications like text-to-speech synthesis (TTS), machine translation (MT), and information retrieval, enhancing their accuracy and effectiveness. Despite its significance, the absence of diacritics in most written texts presents challenges, complicating language processing tasks.

Table 1. Arabic letters and their transcription

Letter	Alif	Bā	Tā	Thā	Jīm	ḥā	Khā	Dāl
Arabic	ا	ب	ت	ث	ج	ح	خ	د
Transcription	ā	b	t	th	j	ḥ	kh	d
Letter	Dhāl	Rā	Zāy	sīn	shīn	ṣād	ḍād	ṭā
Arabic	ذ	ر	ز	س	ش	ص	ض	ط
Transcription	dh	r	z	s	sh	ṣ	ḍ	ṭ
Letter	ẓā	ʿayn	ghayn	fā	qāf	kāf	lām	mīm
Arabic	ظ	ع	غ	ف	ق	ك	ل	م
Transcription	ẓ	ʿ	gh	f	q	k	l	m
Letter	nūn	hā	wāw	yā	hamza	maksura	ta	
Arabic	ن	ه	و	ي	ء	ى	ة	
Transcription	N	h	w	y	ʿ	ā	t	

The exploration of Arabic text diacritization (ATD) has spanned various methodologies, evolving from rule-based to sophisticated deep-learning models. Rule-based methods [2], drawing on linguistic expertise, provide foundational accuracy improvements by embedding grammatical and syntactic knowledge into the diacritization process. Hybrid approaches [3] then build on this foundation, combining the precision of linguistic rules with the adaptive power of statistical and neural models, offering a balanced path towards enhanced diacritization accuracy. Statistical methods [4] contribute to this ecosystem by applying probabilistic algorithms to predict diacritics, often blending with other techniques for improved outcomes. At the cutting edge, deep learning models [5], especially bidirectional long short-term memory (Bi-LSTM) networks, push the boundaries of accuracy by processing vast contextual data, despite facing challenges related to computational resources and data requirements. Collectively, these diverse approaches represent the field's ongoing efforts to refine and advance the task of ATD.

In recent years, transformer models have emerged as a powerful alternative, revolutionizing NLP with their ability to capture long-range dependencies in text. This paper delves into the performance of transformer models, particularly bidirectional encoder representation from transformers (BERT), in ATD, comparing it against other methodologies like the Bi-LSTM model. We specifically focus on the implementation of the BERT model, leveraging Arabic bidirectional encoder representations from transformers (AraBERT)v2, a BERT variant optimized for Arabic, to enhance text diacritization. Through meticulous data preparation, fine-tuning of the model on a comprehensive Arabic dataset, and rigorous evaluation, we unveil the capabilities of transformer-based models in mastering the intricacies of ATD. Our results demonstrate a significant leap in accuracy and efficiency, with AraBERTv2 outperforming traditional models, marking a promising advance in the field. Our objective is to shed light on these model's capabilities in diacritization, highlighting strengths, weaknesses, and avenues for future research. Following an overview of existing diacritization tools with an emphasis on transformer models, we detail the implementation of these models for ATD, outline our experimental framework, and discuss our findings, concluding with insights drawn from our study.

The remainder of this paper is organized as follows: in section 2, we review existing diacritization tools and categorize them, with a specific focus on transformers models. In section 3, we explain how to implement transformers for ATD. Section 4 outlines the experimental setup and presents the results we obtained. Finally, we draw our conclusions in section 5.

2. RELATED WORKS

Transformers are a type of neural network, that recently emerged as a powerful tool for NLP tasks. They are used to model the relationship between the input text and its corresponding output, once the model is trained, it can learn the patterns and relationships between the input text and the corresponding diacritization in the case of ATD. While using transformers for downstream tasks, feature-based and fine-tuning are two main strategies [6].

In the feature-based approach: the pre-trained model is used to extract a fixed set of features from the input text, which are then used as input to a separate model for the downstream task. In the case of ATD, the pre-trained model may be used to extract embeddings or contextualized representations for each character in the input text, which are then fed into a separate model for diacritization. In the Fine-tuning approach: the pre-trained model is trained on a downstream task using task-specific data. The pre-trained model's parameters are updated during this fine-tuning process to improve its performance on the downstream task. This survey categorizes related works into two groups, Transformers for Arabic NLP and transformers for ATD. In each category, the techniques are organized in chronological order to illustrate a clear trajectory of research development within Arabic NLP.

2.1. Transformers for Arabic natural language processing

In this subsection, we explore key developments in Transformer models for Arabic NLP, from multilingual to Arabic-specific models like AraBERT. This review establishes the technological backdrop against which our study introduces enhancements in ATD, leveraging the strengths of these models. Lan *et al.* [7] present a study on the performance of multilingual pre-trained transformers mBERT and XLM-RoBERTa on Arabic information extraction tasks. It introduces GigaBERT, a customized bilingual BERT model designed for Arabic NLP and English-to-Arabic zero-shot transfer learning. The study demonstrates that GigaBERT significantly outperforms existing models in both supervised and zero-shot settings across four information extraction tasks: named entity recognition, part-of-speech tagging, argument role labeling, and relation extraction.

Antoun *et al.* [8] introduce AraBERT, a BERT-based model pre-trained specifically for the Arabic language to improve performance on Arabic natural language understanding (NLU) tasks. The methodology involves pre-training BERT on a large-scale Arabic corpus, then fine-tuning it on downstream tasks like sentiment analysis, named entity recognition, and question answering. The study demonstrates AraBERT's superior performance compared to multilingual BERT and other state-of-the-art approaches, attributing success to its training on a large and diverse Arabic dataset, which includes both MSA and dialectal Arabic.

Antoun *et al.* [9] presents ARAGPT2, the first advanced Arabic language generation model based on transformer architecture, specifically designed for Arabic text generation tasks. The methodology includes pre-training ARAGPT2 on a large and diverse Arabic text corpus, comprising 77 GB of data from various sources, including internet text and news articles. The model comes in several sizes, with the largest, ARAGPT2-MEGA, having 1.46 billion parameters, making it the most extensive Arabic language model available. This model has demonstrated success in tasks such as synthetic news generation and zero-shot question answering. The study also includes the development and release of an automatic discriminator model to detect text generated by ARAGPT2, boasting 98% accuracy.

Mageed *et al.* [10] presents two models, ARBERT and MARBERT, aimed at enhancing Arabic NLP. These models are built to address the limitations of multilingual models, especially in handling diverse Arabic dialects and the challenges associated with non-English data's size and diversity in pre-training. The paper introduces ARLUE, a benchmark for evaluating Arabic language understanding across multiple dialects using 42 datasets spanning six task clusters. The models achieve state-of-the-art results on the majority of tasks, outperforming existing models, including those significantly larger in size. The paper highlights the importance of developing efficient, powerful models for diverse language varieties, specifically for Arabic in this case, and demonstrates the effectiveness of ARBERT and MARBERT through comprehensive evaluations.

Toyin *et al.* [11] introduces a pre-trained model designed to support Arabic language processing in both text and speech forms. This model, based on the unified-modal framework SpeechT5, focuses initially on MSA with future plans to encompass dialectal and code-switched Arabic. ArTST is trained from scratch on MSA speech and text data and is fine-tuned for automatic speech recognition (ASR), TTS, and spoken dialect identification, demonstrating competitive or superior performance across these tasks. The research contributes by releasing the pre-trained model and fine-tuned ASR and TTS models for research purposes, showcasing the model's generalization capabilities, particularly in low-resource TTS tasks.

Beheitt and Hmida [12] outlines a study on utilizing generative pre-trained transformer 2 (GPT-2) for generating Arabic poetry. The authors, explore the challenge of automatic poetry generation in Arabic, a task complicated by the language's rich grammatical structure. They trained a customized GPT-2 model on over 1 million Arabic news articles before fine-tuning it on Arabic poetry. The model demonstrated effectiveness through both automatic and human evaluations, outperforming existing models in generating coherent and stylistically appropriate Arabic poetry.

Alruqi and Alzahrani [13] focuses on enhancing Arabic chatbot capabilities using transfer learning and transformer models such as AraBERT, CAMeLBERT, AraElectra-SQuAD, and AraElectra (generator/discriminator). It explores the development and evaluation of an Arabic chatbot for efficient and accurate question-answering tasks. The study employs two datasets for testing and demonstrates the potential of combining transformer architecture with extractive chatbot methods to provide contextually relevant answers in Arabic, showcasing significant improvements in chatbot performance.

Sakr and Torki [14] focuses on the challenge of adding punctuation to Arabic text, which is crucial for readability and clarity in various applications like ASR and MT systems. It introduces the AraPunc dataset, derived from the Tashkeela corpus, and involves training transformer-based language models for punctuation restoration. The study finds XLM-RoBERTa to outperform other models on the AraPunc test set, demonstrating the effectiveness of transformer-based approaches in addressing the punctuation restoration task in Arabic. The paper contributes to Arabic NLP by providing a new dataset and insights into model performance for punctuation restoration.

Chennafi *et al.* [15] focus on Arabic aspect-based sentiment analysis (ABSA), particularly on aspect term polarity and aspect category polarity tasks. They introduce a Seq2Seq model for dialect normalization as a preprocessing step to improve ABSA classification by reducing out-of-vocabulary (OOV) words. Utilizing pre-trained transformer models like BERT, the study presents new approaches to handling dialectal Arabic and MSA for ABSA tasks. The research demonstrates improved performance over existing models, highlighting the effectiveness of combining dialect normalization with transformer-based models for Arabic sentiment analysis.

Abdurahimov [16] explores using BERT for MT between English and Arabic. It highlights the advancements in MT, focusing on neural MT's superiority over rule-based and statistical approaches. The study emphasizes Arabic's morphological complexity, presenting challenges for MT due to diacritization and the absence of short vowels in writing. By preprocessing data and employing byte-pair encoding (BPE) tokenization, the research demonstrates significant improvements in translation quality, showcasing BERT's potential to enhance Arabic-English MT performance.

While the introduction of multilingual models like mBERT and XLM-RoBERTa marked significant progress in Arabic NLP, specialized models such as GigaBERT and AraBERT have demonstrated superior performance due to their focused training on Arabic corpora. AraBERT, in particular, underscores the advantage of tailoring models to the linguistic characteristics of Arabic, including its dialectal diversity. However, the evolution from multilingual to language-specific models reveals a gap in the exploration of fine-grained dialectal nuances and their impact on NLP tasks. Our study seeks to address this by evaluating the AraBERT model’s adaptability and performance across a spectrum of dialects in the context of ATD, a dimension less explored in previous work.

2.2. Transformers for Arabic text diacritization

Here, we examine transformative approaches in ATD using transformer models. By reviewing pioneering works from ByT5 to BERT-based techniques, we set the stage for our comparative analysis, aiming to further advance ATD's effectiveness with AraBERTv2. Rfooh *et al.* [17] discuss a novel approach for ATD using pre-trained language models. They investigate the use of token-free pre-trained multilingual models (specifically, ByT5) to learn how to predict and insert missing diacritics in Arabic text. Diacritization is a complex task that requires an understanding of sentence semantics and the morphological structure of tokens. By leveraging pre-trained models, the authors are able to achieve state-of-the-art results with minimal training and no feature engineering, reducing word error rate (WER) by 40%.

Aldarmaki and Ghannam [18] evaluates diacritic recognition in Arabic ASR, finding that ASR diacritization outperforms text-based methods, especially when models are fine-tuned with manually diacritized transcripts. This is significant for enhancing ASR systems's accuracy in Arabic, a language where diacritics play a crucial role in meaning. The results suggest potential improvements in ASR applications by incorporating diacritics directly into the system rather than relying on post-processing text diacritization.

Shatnawi *et al.* [19] utilizes the classical Arabic text-to-speech (ClArTTS) Corpus for in-domain data, developed specifically for TTS synthesis with manually diacritized and verified text transcriptions. Out-of-domain testing employed clean subsets from the question answering in Arabic (QASR) dataset, manually diacritized for TTS purposes. Additionally, the cleaned Tashkeela Corpus, comprising 2.3 M words across 55 K lines from classical Arabic literature and religious texts, was also used for the basic model training. Nazih and Hifny [20] present a novel approach to hand Arabic syntactic diacritic restoration using a BERT tagger. The BERT tagger is based on self-attention networks, which run in parallel and are able to model long-term dependencies better than LSTM taggers. The proposed approach achieves a new state-of-the-art absolute case-ending error rate (CEER) without using handcrafted features, on the standard Arabic treebank corpus.

To address the ATD problem, Mijlad and Younoussi [21] propose a letter-based encoder-decoder model that utilizes previous deep-learning attention models, specifically the Luong attention model. During training, the models experienced unstable loss. The results show that the model using local predictive attention achieved the best word and letter error rates, with a diacritic error rate (DER) of approximately 26.80% on the test data. However, the authors note that there is room for improvement in future work.

The shift from general NLP applications to ATD-specific tasks using transformers illustrates a focused attempt to tackle Arabic's diacritization challenges. ByT5's approach to predicting and inserting missing diacritics showcased the potential of leveraging pre-trained models for language-specific tasks. However, while these models have achieved state-of-the-art results, they often require substantial computational resources and extensive training data, which may not be readily available for all dialects or text genres. Our research contributes by comparing the efficiency and effectiveness of these Transformer models in a diacritization context with limited resources, highlighting potential areas for optimization and improved performance in low-resource settings [22].

The Table 2 provides a synthesized overview of recent studies employing transformer models for ATD. These studies utilize various datasets, showcasing the versatility and effectiveness of transformer-based models across different Arabic text corpora. The performance of each model is evaluated using two primary metrics: DER and WER, which collectively offer insights into the models' accuracy in diacritizing Arabic text.

Table 2. Overview of recent studies using transformer models

Reference	DER (%)	WER (%)	Model	Dataset
17	0.74	2.49	ByT5	Tashkeela corpus
18	1.8	6.4	D2 (BERT)	Classical Arabic TTS corpus (ClArTTS) Arabic Speech Corpus
19	2.71	3.17	Whisper-medium (BERT)	Classical Arabic TTS corpus (ClArTTS) Tashkeela Corpus

3. METHODOLOGY: USING TRANSFORMERS FOR ARABIC TEXT DIACRITIZATION
3.1. Overview of transformer architecture

The adoption of transformer models marks a significant advancement in NLP, particularly for languages with intricate morphological features like Arabic. The transformer's unique architecture, surpassing conventional neural models such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs), excels in identifying long-range dependencies and nuanced linguistic patterns crucial for accurate ATD. Its parallel processing capability significantly reduces training time, a critical advantage for processing extensive datasets required for ATD tasks [23].

3.1.1. Transformer model categorization

Transformer models are categorized into three categories: auto-encoding, auto-regressive, and sequence-to-sequence models. Auto-encoding: are designed to reconstruct the input sequence, and can be used for tasks such as denoising or compression. These models typically consist of an encoder network that maps the input sequence to a hidden representation, and a decoder network that reconstructs the input sequence from the hidden representation. Náplava *et al.* [24] proposed a transformer-based model that performs classification of each transformation, described by a diacritic sign to be applied and its position in a word. The model takes non-diacritized text as input and outputs the corresponding diacritized text. Similarly, Dang and Nguyen [25] proposed a transformer-based model with attention masking removed. This model also performs classification of output diacritic mark categories for each input character. The model is trained using non-diacritizd text as input and the corresponding diacritized text as output.

Auto-regressive models: generate output tokens one at a time based on previously generated tokens. These models are highly effective for tasks like language modeling or text generation because they build sequences in a step-by-step manner. Typically, auto-regressive models employ a single decoder network that processes each token sequentially, predicting the next token based on prior outputs. This sequential token generation allows the model to maintain context throughout the output process, ensuring coherent and contextually relevant. The Figure 1 illustrates the progression of transformer-based models from 2018 to 2021, highlighting key developments like GPT, BERT, and GPT-3. These advancements have significantly influenced the evolution of NLP, especially in tasks requiring sequence generation, such as auto-regressive modeling.

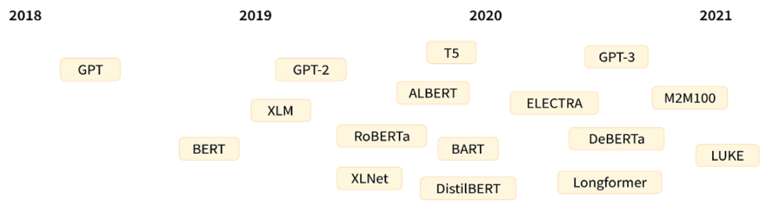


Figure 1. The history of transformer models

Sequence-to-sequence models: Are designed to map an input sequence to an output sequence, and can be used for tasks such as MT or summarization. These models typically consist of an encoder network that maps the input sequence to a hidden representation, and a decoder network that generates the output sequence based on the hidden representation. The encoder and decoder can be trained jointly to optimize the output sequence [22].

In the diagram in Figure 2, the input is passed through an embedding layer to convert the words into continuous representations. The positional encoding is added to capture the positional information of the words.

The transformer encoder consists of two blocks, each containing a self-attention layer followed by a feed-forward layer and layer normalization. The transformer decoder consists of three blocks, each containing a cross-attention layer with the encoder's output, a self-attention layer, a feed-forward layer, and layer normalization. Finally, the output represents the diacritized version of the input text.

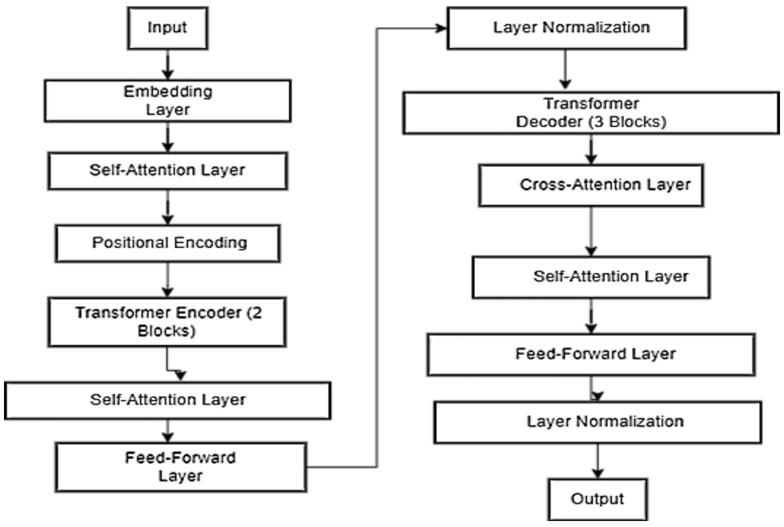


Figure 2. The transformer's basic architecture for ATD

Before feeding the data into the model, the first step is preparing the training data of fully diacritized Arabic text. The second step is input representation. The Arabic text is tokenized into individual words or characters, breaking it down into manageable units for the model. Each token is then mapped to a vector representation using pre-trained word embeddings or character embeddings. This vector representation captures the semantic and contextual information of the tokens, enabling the model to understand the input text. After input representation, the core components of the Transformer architecture come into play. Self-attention allows the model to weigh the importance of each token in the input sequence relative to all other tokens. Multi-head attention is applied to the output of the self-attention layer. It enables the model to attend to different aspects or subspaces of the input text, providing a richer understanding and capturing diverse patterns and dependencies. The output of the multi-head attention layer is then passed through feedforward networks, which apply non-linear transformations to each element in the sequence. These networks enable the model to learn complex relationships and patterns in the data. Finally, the output layer of the model is responsible for predicting the most likely diacritic marks for each token in the input sequence. This allows the model to generate accurate diacritization predictions.

3.2. Implementing transformers for Arabic text diacritization

Our ATD methodology follows a structured, multi-phase approach, starting from data collection and preparation, leading to the final generation of diacritized output. Following thorough preprocessing to enhance data quality, this augmented dataset is utilized in advanced transformer models. Finally, the system generates diacritized Arabic text, which is systematically evaluated against reference datasets to ensure accuracy and optimize performance.

3.2.1. Data preparation

The initial and critical phase in our ATD approach is the meticulous preparation and curation of the training dataset, which is vital for the successful training of our models. This involves:

– Input corpus: Selecting the Tashkeela corpus as our primary data source due to its comprehensive collection of fully diacritized Arabic texts, which provides a wide coverage of topics and styles.
– Data augmentation: Using the dataset in [26], it represents an enhanced form of the Tashkeela corpus, augmented with additional texts from medical and other domain-specific literature from the Shamela library. This expansion incorporates a broad array of terminologies and contexts.
– Annotation quality: Each text included in our dataset has been verified for diacritic accuracy to ensure high-quality training data, which is crucial for the model to learn correct diacritization.

- Preprocessing: Cleaning the dataset to remove any inconsistencies or non-textual elements that may impede the model's learning process. This includes the normalization of white spaces, punctuation, and the unification of character representations.

3.2.2. Input representation

For the Input Representation stage, our methodology is crafted to maintain the linguistic integrity of the Arabic text while converting it into a machine-readable format:

- Tokenization: Breaking down the Arabic text into smaller units such as words or subwords. Considering the morphological richness of Arabic, we use a tokenization approach that can accurately handle prefixes, suffixes, and infixes common in Arabic script.
- Embeddings: Employing pre-trained word embeddings that have been trained on large and diverse Arabic corpora. These embeddings have encoded within them an understanding of Arabic word usage patterns and contexts.
- Contextual embeddings: In cases where we employ models like BERT, the embeddings are contextual, meaning that the representation of a word can change based on the surrounding words, providing a more nuanced understanding of language.
- Semantic and syntactical nuances: Special attention is paid to the way diacritics change the meaning of words in Arabic. The embeddings capture these nuances, enabling the model to predict diacritics that are semantically and syntactically appropriate.
- Vector mapping: We map each token to a high-dimensional space where semantic relationships and syntactic structures are encoded as distances and directions in this space, enabling the model to perform complex linguistic predictions based on this geometric representation.

3.2.3. Model architecture and training

The architecture and training methodology of our transformer-based model are crucial for achieving accurate results. Utilizing an encoder-decoder framework, the encoder processes input tokens through self-attention mechanisms that capture the contextual relationships inherent in Arabic text. The decoder then employs a cross-attention process to generate diacritic marks, ensuring coherence and precision in the output.

- Transformer encoder

Input token processing: The encoder starts by converting the tokenized input into vectors using the embedding layer. This step captures important information about each token and its position in the sequence. By combining both the token and its position, the model is better equipped to understand the grammatical structure of Arabic text.

Self-attention mechanism: We implement multi-head self-attention within the encoder. This mechanism allows the model to consider each token in the context of every other token in the sequence, assigning relevance through learned attention weights. This process is crucial for recognizing the grammatical structure and context-dependent meanings of Arabic words and phrases.

Layer normalization and feed-forward network: We use layer normalization after the self-attention layers to stabilize the activations and speed up the training process. This helps keep the activations consistent, which can improve our model's performance. Following this, we pass the processed data to the decoder, allowing the model to maintain the learned relationships between tokens for effective diacritic prediction in Arabic text

- Transformer decoder

Cross-attention mechanism: In the decoding phase, the cross-attention mechanism takes the encoder output and the previously generated tokens. This mechanism allows the model to focus on relevant parts of the input sequence when predicting each subsequent token, which is crucial for maintaining coherence in diacritic generation. By effectively leveraging this attention, we enhance the quality of the generated output.

Prediction of diacritics: The decoder utilizes a similar multi-head attention and feed-forward structure as the encoder but is tailored to generate the output sequence. It predicts the diacritic marks sequentially, combining information from both the input and the parts of the output sequence generated so far. This method ensures that the predictions are contextually relevant, improving the overall accuracy of the diacritic generation process.

- Output generation

Final diacritic prediction: The decoder's final layer is a linear transformation followed by a softmax function. Which outputs a probability distribution over possible diacritic marks for each token. By selecting the diacritic with the highest probability, we produce a diacritized version of the input text, ensuring that the output is both accurate and contextually appropriate.

Training objective: We employ a categorical cross-entropy loss function to guide the training process. This loss function is minimized during training to fine-tune the model's parameters, making it more effective. Each diacritic mark is treated as a separate class, ensuring that the model learns to distinguish between them accurately.

Optimization and learning rate: We use the Adam optimizer with a customized learning rate scheduler. Initially, the learning rate is warmed up over the first few epochs, which helps stabilize the training. It then decays according to a pre-defined schedule, a method proven to enhance the training stability of transformer models.

Regularization techniques: To prevent overfitting, we implement dropout within the attention and feed-forward layers. Additionally, we use weight decay as part of the optimization strategy. These regularization techniques are crucial for improving the model's ability to generalize to new, unseen data, ultimately enhancing its performance.

3.2.4. Experimental setup and system evaluation

In our experiment, we utilized a Tesla® V100 GPU within the Google Colab Pro+ environment for training and testing. We measured diacritization accuracy with the DER, which helped us compare our fine-tuned AraBERTv2 model to a baseline Bi-LSTM model. We also conducted an error analysis to find areas where the model could improve.

– Computational resources

Our experiments are facilitated by the high computational power of a Tesla® V100 GPU, housed within the Google Colab Pro+ environment. This setup provides us with 32 GB of high-bandwidth GPU memory, which significantly accelerates the training and inference phases of our models. The enhanced performance allows us to efficiently handle large datasets and complex computations.

Software framework: We utilize the 'simple transformers' library, a Python-based toolset that simplifies the use of Transformer models. This library offers convenient functionalities for model training, evaluation, and fine-tuning. Which allows us to focus more on our experiments

Baseline model comparison: The fine-tuned AraBERTv2 model is compared against a baseline Bi-LSTM model, which represents a traditional approach to sequence modeling tasks. The Bi-LSTM architecture is known for its ability to effectively process sequential data, making it a strong point of comparison. This analysis highlights the advancements achieved through Transformer-based models, demonstrating their effectiveness in diacritization tasks.

– System evaluation

DER is employed as the principal evaluation metric to quantify the accuracy of diacritization. DER measures the percentage of characters in the text that are incorrectly diacritized by the model. By utilizing this metric, we can effectively assess the performance of our models and identify areas for improvement in diacritization accuracy.

Metric calculation: We compute DER by aligning the predicted diacriticized text with the ground truth and counting the mismatches. The formula as in (1):

$$DER = \frac{\mathrm{Dw}}{\mathrm{Dw+Dc}} \times 100 \tag{1}$$

Case-ending variations: We perform evaluations both including and excluding case endings (CE). By excluding CE, we obtain a measure that reflects the core diacritic accuracy. Allowing us to focus on the essential diacritization without the complexities introduced by grammatical case endings.

Error analysis: In addition to quantitative metric, an error analysis is conducted to understand the types of errors made by the model. This analysis provides valuable insights into specific areas. Where the model may require further refinement.

4. RESULTS AND DISCUSSION

In our investigation of ATD, we conducted a comparative analysis utilizing two prominent models: AraBERT, a BERT variant tailored for Arabic, and a traditional Bi-LSTM architecture. The dataset employed for this experiment, as utilized in [26], was partitioned into an 80-20 split, serving as our training and development sets, respectively. The evaluation focused on the DER, the AraBERT model demonstrated superior performance, achieving a remarkably low DER of 0.81% alongside a high accuracy rate of 98.15%. Conversely, the Bi-LSTM model, while effective, presented a DER of 1.02% and an accuracy rate of 93.88%. The comparative outcomes not only reinforce the robustness of the AraBERT model but also signify its superior capability in deciphering and applying the complex rules of Arabic diacritization.

To gain further insight into AraBERT's superior performance, we conducted a post-hoc analysis to identify factors contributing to its effectiveness. We hypothesize that the following aspects may play a role:

– Contextual understanding: AraBERT's transformer-based structure, adept at capturing contextual nuances, may offer a deeper understanding of Arabic's morphological complexity, especially when predicting diacritics which are heavily context-dependent.

- Model architecture: The attention mechanisms within AraBERT likely provide an edge over the sequential processing of the Bi-LSTM, especially for the Arabic language, where diacritics can influence and be influenced by non-adjacent characters in a word.
- Training efficiency: The efficient parallelization in the training of AraBERT might lead to more effective learning patterns, particularly when handling the voluminous and diverse dataset provided.
- Future research should concentrate on optimizing these models further, exploring the intricacies that allow AraBERT to outshine its counterparts. Potential avenues for enhancement include:
- Advanced regularization techniques: To mitigate overfitting and improve the model's generalization abilities on unseen data.
- Ablation studies: To dissect the contribution of different model components and attention heads to the overall performance.

These results, illustrated in Table 3, benchmark our AraBERT model's performance against previous studies, providing a clear and quantified insight into its state-of-the-art capability in the realm of ATD. The findings from our experiments underscore the potential of transformer-based models in effectively addressing the linguistic challenges presented by Arabic diacritization. The exploration of underlying factors contributing to the success of these models not only enhances our current understanding but also paves the way for future breakthroughs in Arabic NLP.

Table 3. Comparisons of experimental results (i.e., DER, accuracy) between previous studies and our AraBERT model

Diacritic	DER (%)	ACC (%)
Ours AraBERT	0.81	98.15
Ours Bi_LSTM	1.02	93.88

5. CONCLUSION

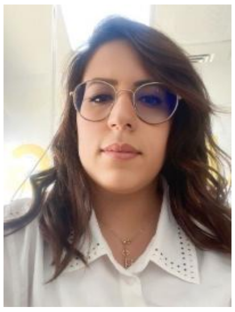
The Arabic language, with its intricate nuances and unique characteristics, presents both challenges and opportunities in the realm of NLP. Our exploration into Arabic diacritization, especially using advanced models like BERT and Bi-LSTM, underscores the pivotal role of machine learning in enhancing linguistic processes. While traditional rule-based methods have their merits, the advent of Transformer-based models, exemplified by BERT, heralds a new era in diacritization research. These models not only capture the rich contextual intricacies of Arabic but also offer scalable solutions for diverse and voluminous texts. As we move forward, continuous refinement of these techniques, coupled with the integration of emerging technologies, will undoubtedly pave the way for more accurate, efficient, and comprehensive Arabic text processing systems.

REFERENCES

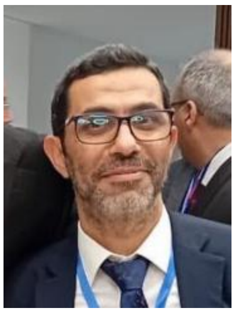
[1] K. Darwish, A. Abdelali, H. Mubarak, and M. Eldesouki, “Arabic diacritic recovery using a feature-rich bilstm model,” *ACM Transactions on Asian and Low-Resource Language Information Processing*, vol. 20, no. 2, 2021, doi: 10.1145/3434235.
[2] A. Fadel, I. Tuffaha, B. Al-Jawarneh, and M. Al-Ayyoub, “Arabic text diacritization using deep neural networks,” *2nd International Conference on Computer Applications and Information Security, ICCAIS 2019*, 2019, doi: 10.1109/CAIS.2019.8769512.
[3] and M. A. M. Elshafei, H. Al-Muhtaseb, “Statistical methods for automatic diacritization of Arabic text,” *Proceedings 18th National computer Conference*, 2006.
[4] H. Abbad and S. Xiong, “Multi-components system for automatic arabic diacritization,” *Advances in Information Retrieval*, pp. 341–355, 2020, doi: 10.1007/978-3-030-45439-5_23.
[5] 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,” *International Journal on Document Analysis and Recognition*, vol. 18, no. 2, pp. 183–197, 2015, doi: 10.1007/s10032-015-0242-2.
[6] J. Devlin, M. W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of deep bidirectional transformers for language understanding,” *NAACL HLT 2019 - 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference*, vol. 1, pp. 4171–4186, 2019.
[7] W. Lan, Y. Chen, W. Xu, and A. Ritter, “An empirical study of pre-trained transformers for Arabic information extraction,” *EMNLP 2020 - 2020 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference*, pp. 4727–4734, 2020, doi: 10.18653/v1/2020.emnlp-main.382.
[8] W. Antoun, F. Baly, and H. Hajj, “AraBERT: Transformer-based model for Arabic language understanding,” *arXiv-Computer Science*, pp. 1-7, 2020.
[9] W. Antoun, F. Baly, and H. Hajj, “ARAGPT2: pre-trained transformer for Arabic language generation,” *WANLP 2021 - 6th Arabic Natural Language Processing Workshop, Proceedings of the Workshop*, pp. 196–207, 2021.
[10] M. A. -Mageed, A. R. Elmadany, and E. M. B. Nagoudi, “ARBERT & MARBERT: Deep bidirectional transformers for Arabic,” *ACL-IJCNLP 2021 - 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*, pp. 7088–7105, 2021, doi: 10.18653/v1/2021.acl-long.551.
[11] H. O. Toyin, A. Djanibekov, A. Kulkarni, and H. Aldarmaki, “ArTST: Arabic text and speech transformer,” *ArabicNLP 2023 - 1st Arabic Natural Language Processing Conference*, pp. 41–51, 2023, doi: 10.18653/v1/2023.arabicnlp-1.5.

[12] M. E. G. Beheitt and M. B. H. Hmida, “Automatic Arabic poem generation with GPT-2,” *International Conference on Agents and Artificial Intelligence*, vol. 2, pp. 366–374, 2022, doi: 10.5220/0010847100003116.
[13] T. N. Alruqi and S. M. Alzahrani, “Evaluation of an Arabic chatbot based on extractive question-answering transfer learning and language transformers,” *AI*, vol. 4, no. 3, pp. 667–691, 2023, doi: 10.3390/ai4030035.
[14] A. Sakr and M. Torki, “AraPunc: Arabic punctuation restoration using transformers,” *Proceedings of IEEE/ACS International Conference on Computer Systems and Applications, AICCSA*, 2023, doi: 10.1109/AICCSA59173.2023.10479326.
[15] M. E. Chennafi, H. Bedlaoui, A. Dahou, and M. A. A. Al-qaness, “Arabic aspect-based sentiment classification using Seq2Seq dialect normalization and transformers,” *Knowledge*, vol. 2, no. 3, pp. 388–401, 2022, doi: 10.3390/knowledge2030022.
[16] M. Abdurahimov, “Leveraging BERT for English-Arabic machine translation,” *Research Gate*, pp. 1-9, 2023.
[17] B. Al-Rfooh, G. Abandah, and R. Al-Rfou, “Fine-Tashkeel: Fine-tuning byte-level models for accurate Arabic text diacritization,” in *2023 IEEE Jordan International Joint Conference on Electrical Engineering and Information Technology (JEEIT)*, May 2023, pp. 199–204, doi: 10.1109/JEEIT58638.2023.10185725.
[18] H. Aldarmaki and A. Ghannam, “Diacritic recognition performance in Arabic ASR,” *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, pp. 361–365, 2023, doi: 10.21437/Interspeech.2023-2344.
[19] S. Shatnawi, S. Alqahtani, and H. Aldarmaki, “Automatic restoration of diacritics for speech data sets,” *The 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 4166–4176, 2024.
[20] W. Nazih and Y. Hifny, “Arabic syntactic diacritics restoration using BERT models,” *Computational Intelligence and Neuroscience*, vol. 2022, 2022, doi: 10.1155/2022/3214255.
[21] A. Mijlad and Y. E. Younoussi, “The global and local attention for automatic Arabic text diacritization,” *International Journal of Engineering and Applied Physics*, vol. 3, no. 1, pp. 653–662, 2023.
[22] L. Stankevičius, M. Lukoševičius, J. Kapočiūtė-Dzikienė, M. Briedienė, and T. Krilavičius, “Correcting diacritics and typos with a ByT5 transformer model,” *Applied Sciences*, vol. 12, no. 5, 2022, doi: 10.3390/app12052636.
[23] T. Wolf *et al.*, “Transformers: state-of-the-art natural language processing,” *EMNLP 2020 - Conference on Empirical Methods in Natural Language Processing, Proceedings of Systems Demonstrations*, pp. 38–45, 2020, doi: 10.18653/v1/2020.emnlp-demos.6.
[24] J. Náplava, M. Straka, and J. Straková, “Diacritics restoration using BERT with analysis on Czech language,” *Prague Bulletin of Mathematical Linguistics*, vol. 116, no. 1, pp. 27–42, 2021, doi: 10.14712/00326585.013.
[25] T. D. A. Dang and T. T. T. Nguyen, “TDP – a hybrid diacritic restoration with transformer decoder,” *Proceedings of the 34th Pacific Asia Conference on Language, Information and Computation*, pp. 76–83, 202.
[26] Z. Iman, S. Adnan, and E. M. B. Eddine, “Neural network for Arabic text diacritization on a new dataset,” *Proceedings of the 6th International Conference on Big Data and Internet of Things*, pp. 186–199, 2023, doi: 10.1007/978-3-031-28387-1_17.

BIOGRAPHIES OF AUTHORS



Iman Zubeiri received her master’s degree in Computer Science and Networking from Science Faculty of science and technology Fez, Fez in 2014. At present, she is working as a data engeneer at ALTEN morocco. She can be contacted at email: imane.zubeiri@gmail.com.



Adnan Souri is an assistant professor in computer science at Abdelmalek Essaâdi University, Faculty of Sciences Tetouan, Morocco. He is a former teacher of computing at preparatory classes for engineering schools – Tangier. He received his Ph.D. degree in computer science from the Faculty of Sciences Tetouan, Abdelmalek Essaâdi University Morocco. His current research interests include artificial intelligence, artificial neural networks, and algorithms. Their current project is “Arabic language processing”. He can be contacted at email: a.souri@uae.ac.ma.

Badr Eddine El Mohajir is a professor in computer science at Abdelmalek Essaâdi University, Faculty of Sciences Tetouan, Morocco. He received his Ph.D. degree in applied science from the university libre of Bruxelles. His current research interests include information systems and databases, open-source applications, and networking. He can be contacted at email: badreddine@elmohajir.com.