Advancing Arabic Diacritization: Improved Datasets, Benchmarking, and State-of-the-Art Models

Anonymous ACL submission

Abstract

Arabic diacritics, similar to short vowels in English, provide phonetic and grammatical information but are typically omitted in written Arabic, leading to ambiguity. Diacritization (aka diacritic restoration or vowelization) is essential for natural language processing. This paper advances Arabic diacritization through the following contributions: first, we propose a methodology to analyze and refine a large diacritized corpus to improve training quality. Second, we introduce WikiNews-2024, a multi-reference evaluation methodology with an updated version of the standard benchmark "WikiNews-2014". In addition, we propose a BiLSTM-based model that achieves state-of-the-art results with 3.12% and 2.70% WER on WikiNews-2014 and WikiNews-2024. Moreover, we develop a model that preserves user-specified diacritics while maintaining accuracy. Lastly, we demonstrate that augmenting training data enhances performance in low-resource settings.

1 Introduction

Arabic, as one of the most widely spoken languages in the world, presents unique challenges for natural language processing (NLP) due to its rich morphology, complex syntactic structures, and the lack of explicit diacritics in written text. Unlike many other languages, Modern Standard Arabic (MSA) is typically written without diacritics—small marks placed above or below letters to indicate short vowels and other phonetic information. The absence of diacritics introduces significant ambiguity, as many words with identical consonantal structures (roots) can have multiple meanings depending on the context. For instance, the unspecified word (" علم" Elm)[1] could mean "science" (عِلْم E**i**lm), "flag" (عَلَم E**a**l**a**m), or "he knew" (عَلِمَ E**a**l**i**m**a**), depending on its diacritization and contexts. A description of Arabic diacritics is presented in Appendix A.

Arabic diacritization can be likened to restoring short vowels in English text written without them. For instance, given the input "Ths sntnc s hrd t undrstnd wtht vwls.", the diacritized output would be "This sentence is hard to understand without vowels."—resolving ambiguity through phonetic and grammatical cues. Diacritization is crucial for a variety of NLP tasks. It enhances machine translation (Thompson and Alshehri, 2021), speech recognition (Aldarmaki and Ghannam, 2023), and text-to-speech systems (Ungurean et al., 2008; Halabi, 2016) by providing accurate phonetic and grammatical information. Moreover, diacritization is essential in educational tools designed for non-native learners of Arabic and in preserving religious and classical texts, where correct pronunciation and interpretation are critical. In real-life applications, diacritization improves the accessibility and usability of Arabic content for voice assistants and automated reading systems.

This work makes several contributions to the field of Arabic diacritization. **I)** Through detailed analysis of a large-scale diacritized dataset, we identify and address key quality and consistency challenges, establishing methodological improvements that enhance the reliability of training data for diacritization systems. **II)** We develop a novel evaluation methodology that accounts for the richness of the Arabic morphology and the inherent ambiguity in Arabic text by introducing a multi-reference version of the standard WikiNews benchmark (from 2014), providing a more nuanced and accurate assessment of system performance. **III)** Furthermore, we release WikiNews-2024, a new benchmark meticulously reviewed by two expert linguists, establishing a recent reliable standard for evaluating diacritization systems. **IV)** Our research explores various neural network architec-

[1]We use Buckwalter Transliteration.

tures and introduces a BiLSTM-based architecture that achieves state-of-the-art performance on the WikiNews benchmark, significantly advancing the accuracy of Arabic diacritization. The model is robust to the new benchmark Wikinews-2024. **V)** We also introduce a diacritization model that preserves any user-provided diacritical marks while achieving near state-of-the-art accuracy. This makes it especially useful in real-world settings where inputs are often partially diacritized—for example, keeping user-specified diacritics on domain-specific terms or foreign named entities. **VI)** Finally, we demonstrate that when training data is scarce, model performance can be enhanced by diacritizing high-quality data and using it to augment the training set. We make our codes and benchmarks publicly available.

2 Related Work

Arabic diacritization has been extensively studied, with early methods relying on Hidden Markov Models (Gal, 2002; Elshafei et al., 2006) and acoustic-morphological hybrid approaches (Vergyri and Kirchhoff, 2004). Finite-state transducers (Nelken and Shieber, 2005) and maximum entropy model (Zitouni et al., 2006) were later introduced, improving accuracy using lexical and syntactic features. Morphologically aware models (Pasha et al., 2014; Rashwan et al., 2015); incorporated POS tagging and n-gram language models.

The advent of deep learning brought significant advancements in Arabic diacritization. Abandah et al. (2015) explored the use of recurrent neural networks (RNNs), demonstrating improved performance over traditional methods. Building upon this, Belinkov and Glass (2015) employed a long short-term memory (LSTM) network, achieving state-of-the-art results by capturing long-range dependencies in Arabic text. More recently, BERT-based and transformer models have been investigated for diacritization tasks. Zalmout and Habash (2019) used multitask learning and neural language models for lemmatization and diacritization.

Notably, Darwish et al. (2017) proposed a diacritization system that combines statistical methods with linguistic rules, achieving high accuracy and efficiency. Their approach involves a two-step process: The first step is guessing the diacritics of the core words, followed by determining their case endings. This method demonstrated significant improvements in both speed and accuracy compared to previous models. Further advancements were made by Mubarak et al. (2019b), who introduced a sequence-to-sequence modeling approach for Arabic diacritization. Their model leverages the capabilities of neural networks to predict diacritics, resulting in highly effective performance. Later, they showed the effectiveness of their approach on four varieties of Arabic (Mubarak et al., 2019a). This approach addresses some of the limitations of earlier methods by better capturing the sequential nature of language. Moreover, Alqahtani et al. (2020) used multi-task learning to jointly optimize diacritization with segmentation, and POS tagging for Arabic Treebank (Maamouri et al., 2010). Elmallah et al. (2024) used a Recurrent Neural Network-based architecture combined with morphological segmentation, but the need for highly accurate segmenters for all varieties of Arabic may limit the applicability of this approach.

On the standard WikiNews benchmark, Mubarak et al. (2019a) showed that their system achieved the best result with a Word Error Rate% (WER) of 04.49, followed by Microsoft ATKS (Said et al., 2013), Farasa (Darwish et al., 2017), RDI (Rashwan et al., 2015), MADAMIRA (Pasha et al., 2014), and MIT (Belinkov and Glass, 2015) with WER of 12.25, 12.76, 15.95, 19.02, and 30.50 in order.

Despite these advancements, challenges remain in handling ambiguity, domain generalization, and annotation inconsistencies. Our work enhances diacritization by leveraging a BiLSTM-based model trained on a diverse dataset and introducing a human-in-the-loop correction mechanism to improve training and evaluation reliability.

3 Methodology

3.1 Dataset Quality Improvement

Diacritizing Arabic texts is a complex and labor-intensive task that is prone to errors and presents numerous challenges, including consistency across the whole corpus, the correct diacritization of foreign names, handling ambiguities in sentence structure, and addressing visual errors that may be difficult for humans to detect.

We used a comprehensive MSA corpus comprising approximately 129K sentences and 4.7M words to train our diacritization models. The training dataset was obtained through a licensed commercial provider under terms that prevent redistribution. The training corpus encompasses diverse topics, including news articles, scientific content, political discourse, and sports coverage.

To systematically analyze and identify quality and consistency issues in the dataset, we used a frequency-based analysis approach. For every unique word stem in the corpus, we mapped the different sequences of diacritics assigned to the stem and their frequencies. We focused the analysis on frequently occurring words by setting a minimum occurrence threshold. We chose a threshold of 50 occurrences as it provided a reasonable compromise between the number of words to analyze and the overall coverage of all the words in the dataset.

Usually, the diacritical marks on the last letter of an Arabic word indicate the word's grammatical role and as such would depend on the context in which the word is used in a sentence. Therefore, we considered the remaining diacritics (core-word diacritics) to assess the diacritization consistency.

To detect potential errors, we identified words with a dominant diacritical form by applying a 95% frequency threshold. Deviations from this form were inspected, revealing inconsistencies in the dataset. A linguist reviewed and corrected doubtful cases to ensure accuracy. Examples of the quality issues we uncovered include: *(i)* inconsistent diacritization of words beginning with *alif-lam*, the definite article in Arabic. (e.g. الشمس (Al$ms) might appear as الشَّمْس (Al$∼amos) or الشَمْس (Al$amos)), and *(ii)* missing diacritics on word-initial positions, especially for words beginning with *alif* (e.g. إِخْتَلَفُوا (AixotalafuwA) and اخْتَلَفُوا (AxotalafuwA)). These issues were fixed by:

- Correcting the diacritization of *alif-lam* by adding a *sukun* to the *lam* if followed by a *moon letter* or adding a *shadda* to the following letter if followed by a *sun letter*.[2]
- Reviewing words with a missing diacritic in the first letter and adding the correct diacritic

Note that this frequency-based analysis approach is dataset-agnostic. Thus, it can be used on any diacritized dataset to detect similar issues and correct them to improve dataset quality. Table 4 shows the effect of the data cleaning on the model performance.

3.2 Representation Unit

The diacritization relies on both character- and word-level context, making character-based representation a natural choice. Each input sentence is encoded as two sequences: one consisting of the characters (including whitespace) and another of the corresponding diacritics. Diacritics are mapped to numerical values (e.g., *kasra* = 2), while non-diacritized characters and whitespace are assigned 0. For example, the phrase صَبَاحُ الْخَيْرِ (SabAHu Aloxayori-Good Morning) is represented as ([ص, ب, ا, ح, " ", ا, ل, خ, ي, ر], [1, 1, 0, 3, 0, 0, 4, 1, 4, 2]); i.e., consonant letters (S,b,A,H, " ", A,l,x,y,r) followed by their diacritic codes.

We also introduce a **variable-length representation** that combines both character and word-level representations. We used the list of words with a dominant diacritical form collected from the dataset analysis to create either fully or partially diacritized word tokens. For example, instead of representing the word ذَلِكَ (∗alika-"that") as [ذ, ل, ك] (∗,l,k; only the consonants), it would be represented as [ذَلِكَ] (∗alika; fully diacritized) as the diacritization of this word dominantly does not depend on the context. Any letters within a word with ambiguous diacritization are assigned as separate tokens, allowing partially diacritized words to appear in the input. For example, words with a dominant core word diacritic form are split into two tokens, one representing the core word and one token for the last letter which typically carries the grammatical case ending marker. As an example, the string وَأَشَارَ إِلَى أَن أُسْلُوب ([He] indicated that the manner ...) is represented as [وَأَشَارَ, ' ', إِلَى, ' ', أَ, ن, ' ', أُسْلُو, ب] where أَن is split as the diacritization of the letter *nun* (ن) is ambiguous but the initial word وَأَشَارَ (WaAshaRa [He indicated]) is represented as one token only since this is its dominant diacritized form.

The variable-length representation has several benefits, such as reducing the number of tokens required to represent inputs and thus making more effective use of shorter context windows. Moreover, partially diacritized tokens should reduce errors on words with a dominant diacritic form, as it removes the guesswork. More importantly, it allows us to preserve the diacritics present in the input text, a capability often ignored in previous works on Arabic diacritization.

[2]For further explanation on sun and moon letters: https://en.wikipedia.org/wiki/Sun_and_moon_letters

We prefer the variable-length representation in preserving user-input diacritics over post-processing as it avoids errors generated by substituting diacritics in the model output. For example, consider the case when the user inputs يُكتب (y**u**ktb - is being written); other models usually ignore the input diacritics and might output يَكْتُب (y**a**kt**u**b - he writes). If we post-process and replace the first *fatha* with a *damma*, we get يُكْتُب (*y**u**kt**u**b), which is not a valid Arabic word.

3.3 Models

To test a model, an input sentence without diacritics is given to the model, and its output is compared with a reference text that is fully diacritized. We tested several neural network models with different configurations to determine the best model for diacritization. Previous work in the field has shown the effectiveness of bidirectional long-short-term memory (Bi-LSTM) models. We experimented with different configurations of Bi-LSTM models with varying numbers of layers and context window sizes and investigated using a CRF layer for decoding. We also tested two transformer models, one built from scratch and one fine-tuned from AraBERT (Antoun et al., 2020), as transformers have shown exemplary performance in various Arabic NLP tasks in different domains.

The Bi-LSTM models consist of an embedding layer, followed by a sequence of Bi-LSTM layers, then three dense layers, followed by an output layer. We set the embedding dimension to 20 and the Bi-LSTM hidden dimension to 512. The first two dense layers had an output dimension of 512, and the last one had an output dimension of 256. The choice for dimension sizes is inspired by previous works (Elmallah et al., 2024). The output layer has an output dimension of 15, representing the 14 possible diacritics and the null diacritic. We tested configurations consisting of 2, 4, and 6 Bi-LSTM layers. All models were trained using the Adam optimizer with a learning rate of 0.001, $\beta_1 = 0.9$ and $\beta_2 = 0.999$. To avoid overfitting, we added dropout layers between the Bi-LSTM layers with a dropout probability of 20% and used early stopping with a patience of 3.

The transformer model (built from scratch) consisted of 4 encoder layers, each with a 4-head attention mechanism and a linear output layer. The embedding dimension was set to 512, and the dropout probability was set to 15%. Both transformer models were trained with the AdamW optimizer with a learning rate of 0.001 $\beta_1 = 0.9$ and $\beta_2 = 0.999$. All models were trained with a batch size of 64.

3.4 Context Window

Arabic diacritization is sensitive to word context. Therefore, a larger context window would allow models to capture long-term dependencies, especially in cases of conjunction/coordination. However, using a character-level representation for input sentences could lead to large token sequences that significantly increase model training and inference time. We employed a simple solution of setting a fixed context size to split the input sentences. The choice of context window size was inspired by analyzing the sentence lengths within our training corpus. In our case, we found 250 tokens to be an appropriate context window size as the average sentence within the corpus consisted of 209 characters, and a context size of 250 meant that 80% of the sentences would not require splitting. When using the partially diacritized tokens representation, a context window size of 250 meant that 92% of the sentences would not require splitting. We tested all the models with a context window of 250 and a context window of 100 to determine the effect of the increase in context window size.

While a larger context window enhances the model's ability to capture long-range dependencies, our empirical analysis indicates that short-range context is typically sufficient for accurate diacritization. This finding is in line with previous research on the topic (Mubarak et al., 2019b). Given this observation, we opted not to implement a sliding window approach, which would substantially increase inference time without proportional performance gains. This decision was further validated by our model, which achieved state-of-the-art results without employing sliding window techniques. Future research directions could explore integrating sliding window mechanisms with our architecture, potentially offering insights into the trade-offs between computational efficiency and marginal improvements in diacritization accuracy.

4 Evaluation

4.1 Multi-Reference Diacritization

We introduce the concept of *multi-reference diacritization* for the first time, addressing a crucial gap in previous works and benchmarks. According to Arabic lexicons, many Arabic words have multiple valid diacritized forms. For example, the word الجمعة (Friday) can be diacritized and pronounced as الجُمْعةُ ، الجُمُعةُ ، الجُمَعَةُ (“ju**mo**Ea,” “ju**mu**Ea,” or “ju**ma**Ea”) [3]. Similarly, foreign names often have multiple acceptable diacritizations; for instance, المكسيك (Mexico) appears as المَكْسِيك، المِكْسِيك (“Al**ma**ksyk” or “Al**mi**ksyk”).

Furthermore, Arabic grammar prescribes grammatical case-ending markers for foreign named entities to indicate their syntactic roles. For example, nominative case (subject position) requires a final *damma*, while accusative and genitive cases (object or noun-preposition constructions) require a final *fatha*. Thus, the sentence زار دونالد ترامب باريس (Donald Trump visited Paris) should be diacritized as زارَ دونالدُ ترامبَ باريسَ (Donald**u** Trump**a** visited Paris**a**). However, contemporary usage often omits these markers, as they can affect pronunciation or feel unnatural. A renowned Arabic scholar (https://shamela.ws/book/20642/530#p1) states that “a foreign name originally written in Latin letters should follow its pronunciation in the source language.” To accommodate both traditional and contemporary conventions, we introduce an alternative approach in our benchmarks, including both grammatical case endings and *sukun* (null diacritic) for foreign named entities. For instance, in the example above, we modify the reference to زار دونالدْ / دونالدُ ترامبْ / ترامبَ باريسْ / باريسَ (Donal**d**/Donal**du** Trum**p**/Trum**pa** visited Pari**s**/Pari**sa**..)

Additionally, in Arabic phonetics, when a word ending in *sukun* is followed by a word beginning with a *plain alif*, the *sukun* is typically replaced by either a *fatha* or *kasra*. For example, the phrase مِنْ اَلْبَيْتِ (“mi**n** albayti” – from home) can alternatively appear as مِنَ الْبَيْتِ (“mi**na** albayti”). This phenomenon, known as اِلتقاء الساكنيْن (Consonant Clustering), is a fundamental aspect of Arabic phonology. To account for this variation, both diacritic forms are included in our benchmarks.

Moreover, in some cases, syntactic ambiguity allows a word to take multiple valid grammatical case endings. For example, in the sentence دمر المقرات في المدينة كلها (“it destroyed the headquarters in the city *[in all of it/all of them]*), the adverb كلها can be diacritized as كلِّها (kull**i**ha) or كلَّها (kull**a**ha), depending on its referent within the sentence, and both forms are valid. The modified benchmarks include common modern usage cases.

The first diacritization form is ensured to be the most frequently used, making it suitable for single-reference evaluation when needed. The linguists were asked to follow these instructions carefully.

4.2 Standard Benchmarking

The authors in Darwish et al. (2017) introduced and publicly released the **WikiNews-2014** dataset as a benchmark for Arabic diacritization. This dataset comprises 70 WikiNews articles (18,300 words) from 2013 and 2014, covering diverse topics such as politics, economics, health, science and technology, and sports. It was created to address the limitations of existing benchmarks, such as the Arabic Treebank (Maamouri et al., 2010), which suffer from availability issues and annotation errors.

WikiNews-2014 has since become a de facto standard for diacritization evaluation. However, it does not account for multi-reference diacritization and has become outdated, as many new names and events have emerged since its release. To enhance its usability, we augmented the dataset with multi-reference annotations and modified the standard scoring script provided with it to accept any of the valid diactizations. Additionally, we curated a new dataset, **WikiNews-2024**, by collecting 35 news articles (more than 10K words) from WikiNews[4] in 2024, spanning various topics. Two expert linguists reviewed the dataset individually to correct spelling and grammar errors, fully diacritize the text, and incorporate multi-reference annotations where applicable. Any discrepancies between the linguists were resolved during the final revision stage. Notably, 785 words in WikiNews-2024 (7%) exhibit multiple valid diacritized forms. The modified WikiNews-2014, the updated evaluation program, and the newly created WikiNews-2024 dataset will be made available for research purposes.

[3]Arabic lexicon: https://www.almaany.com

[4]https://ar.wikinews.org/

	Character Representation				Variable-len Representation			
Context Size	100		250		100		250	
Model	WER	DER	WER	DER	WER	DER	WER	DER
BiLSTM (2)	4.92%	1.24%	3.99%	1.02%	4.62%	1.20%	4.36%	1.17%
BiLSTM (4)	4.86%	1.24%	**3.74%**	**0.96%**	4.46%	1.17%	4.10%	1.06%
BiLSTM (6)	4.92%	1.24%	4.43%	1.13%	4.78%	1.22%	4.90%	1.26%
Transformer-A	7.13%	1.69%	5.16%	1.35%	5.44%	1.39%	5.21%	1.36%
Transformer-S	6.01%	1.52%	4.45%	1.15%	5.17%	1.34%	4.86%	1.27%

Table 1: Single-Reference Evaluation Results on WikiNews-2014. The numbers in brackets indicate the number of BiLSTM layers. Transformer-A refers to the transformer trained on top of AraBERT, and Transformer-S refers to the transformer trained from scratch.

	Character Representation				Variable-len Representation			
Context Size	100		250		100		250	
Model	WER	DER	WER	DER	WER	DER	WER	DER
BiLSTM (2)	4.75%	1.28%	3.61%	1.01%	4.14%	1.18%	4.05%	1.14%
BiLSTM (4)	4.40%	1.20%	**3.27%**	**0.93%**	3.99%	1.14%	3.62%	1.02%
BiLSTM (6)	4.45%	1.21%	3.90%	1.09%	4.29%	1.21%	4.33%	1.22%
Transformer-A	6.82%	1.63%	4.71%	1.32%	4.95%	1.33%	4.73%	1.33%
Transformer-S	5.51%	1.50%	3.97%	1.12%	4.70%	1.31%	4.38%	1.24%

Table 2: Multi-Reference Evaluation Results on WikiNews-2014

5 Experiments

As shown in section 3.1, a corpus comprising around 129K sentences and 4.7M words was used for training. Applying the character representation to the training dataset resulted in around 27M tokens, while the variable-length representation resulted in around 17M tokens. For each experiment, 95% of the training data was used for training and 5% for validation. All models were trained with early stopping with a patience of 5. Details of the experimental setup are presented in Appendix B.

Each model specified in section 3.3 was tested on the WikiNews-2014. For each model, we varied the context window size to 100 or 250 (as explained in section 3.4) and tested both character-level and variable-length representations. We calculated the Word Error Rate (WER), which represents the percentage of words with at least one diacritic mistake, and the Diacritic Error Rate (DER), which represents the percentage of characters with a diacritic mistake. We report both single-reference and multi-reference evaluation results. The single-reference results are reported in Table 1. The multi-reference results are reported in Table 2.

Comparison of Tables 1 and 2 shows a consistent 0.4-0.5% WER improvement when using the multi-reference evaluation. This confirms the need for multi-reference evaluation to address the issue of words with multiple valid diacritizations that has been completely ignored by previous work.

The results indicate that the BiLSTM models consistently outperformed both transformer models. The BiLSTM model with 4 layers consistently outperformed the other models across all experiments, hinting that the 4-layer configuration might achieve the desired model complexity without overfitting. We experimented with a CRF layer, but it did not lead to any improvements. Moreover, the transformer built from scratch consistently outperformed the transformer based on the AraBERT embeddings. This performance gap could potentially be due to the distribution shift as the AraBERT models were primarily trained on a subword level rather than a character level (Antoun et al., 2020).

Tables 1 and 2 show that the variable-length representation led to more accurate models than the character representation for shorter context windows. This could be attributed to the variable-length representation's ability to compact more information into a shorter token sequence.

The character representation yielded the best model performance for the longer context windows. We hypothesized that this is because the model can learn inter-character dependencies that are not as apparent when using the variable-length representation, as the latter representation could combine sequences of characters into one token. We tested this hypothesis by training the 4-layer BiLSTM model using a combination of the character and

Exp	Single-ref.		Multi-ref.	
Var-len rep.	4.10%	1.06%	3.62%	1.02%
Char + Var-len rep.	**3.90%**	**1.01%**	**3.42%**	**0.98%**

Table 3: Effect of combining character-level representation with variable-length representation on model accuracy (WER and DER)

Exp	WER	DER
Before Quality Improv.	3.91%	1.00%
After Quality Improv.	**3.74%**	**0.96%**

Table 4: Effect of proposed data quality improvement techniques on 4-layer BiLSTM model performance

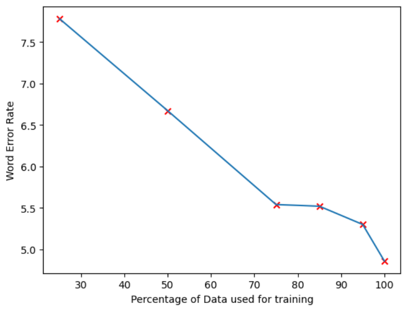

Figure 1: Effect of training dataset size on WER

variable-length representations. We then evaluated the model using the variable-length representation to tokenize the evaluation data.

Table 3 shows that combining both representations leads to improved diacritization accuracy relative to only using the variable-length representation while providing the ability to preserve user-provided diacritics. This result is notable as the model achieves a lower WER than the SOTA models while preserving user-provided diacritics.

We trained the best-performing model configuration, 4-layer BiLSTM, on the original dataset to validate the quality improvement discussed in section 3.1. Table 4 shows a 4.5% relative gain in accuracy due to the data quality improvement, highlighting the need for high-quality diacritized data for training. We also tested the effect of the training data size on WER to determine if there were potential gains from using a larger dataset. We trained the best-performing BiLSTM model on varying percentages of the training dataset. Fig. 1 shows a consistent benefit to using a larger dataset. This finding motivated our idea to augment the training dataset by using the best-performing diacritizer to diacritize different sources and add them to the training data. We performed two data augmentation experiments. In the first experiment, we diacritized 4.5 million words (a size similar to our initial dataset) from the Arabic Wikipedia and added it to the training set. In the second experiment, we diacritized approximately 4.6 million words from an in-house curated, high-quality news source dataset. Table 5 shows that data augmentation improves performance, even with automatically diacritized data. The model achieved relative improvements of 3.7% and 4.6% when augmented with Arabic Wikipedia and high-quality in-house data, respectively. Table 6 highlights that our model outperforms the SOTA models for this task on the WikiNews-2014 dataset. The prompt used for GPT-4o is presented in Appendix C. Note that we compared the model to the RNN model without segmentation (Elmallah et al., 2024) as the other RNN model requires the use of a highly accurate segmentation model to preprocess the data, which is an extra requirement that would significantly decrease model speed as well as make it challenging to extend the usability of the model to other dialects or contexts.

We tested the 4-layer BiLSTM, the best model, on the new WikiNews-2024 benchmark. Table 7 shows that we achieved a 2.70% WER on the new dataset, demonstrating the model's robustness in dealing with unseen data. Table 8 shows the time taken to diacritize WikiNews-2014 for different models. The results indicate that our model is about 3 times faster than Farasa. It is also interesting to note that the model that preserves user-input diacritics is 27% faster than the model using the character representation.

Exp	Single-ref.		Multi-ref.	
	WER	DER	WER	DER
No Augmnt.	3.74%	0.96%	3.27%	0.93%
Aug (Wikipedia)	3.63%	0.93%	3.15%	0.90%
Aug (HQ in-house)	**3.61%**	**0.93%**	**3.12%**	**0.90%**

Table 5: Effect of data augmentation on diacritizer accuracy. Wiki refers to augmentation using Arabic Wikipedia articles. HQ in-house refers to high-quality in-house curated dataset from diverse news sources.

Model	Time (s)
GPT-4o	269
Farasa	9.1
Char. BiLSTM	3.3
Char + Var-len BiLSTM	**2.4**

Table 8: Time taken to diacritize WikiNews-2014

As part of this work, we release a fully diacritized corpus comprising 5 million words (ex-

Model	WER	DER
Our System	**3.61%**	**0.93%**
Farasa (Darwish et al., 2017)	11.50%	3.06%
Seq2Seq Model (Mubarak et al., 2019b)	4.49%	1.21%
RNN (w\o segmentation) (Elmallah et al., 2024)	5.70%	1.40%
GPT-4o	8.55%	3.31%

Table 6: Comparison to other full-diacritization systems (Single Reference)

Dataset	WER	DER
WikiNews-2014	3.12%	0.90%
WikiNews-2024	2.70%	0.94%

Table 7: Our best model's performance on both WikiNews datasets (Multi-reference)

ceeding our training corpus of 4.7M words), generated by our best-performing model (achieving 3.61% WER) on randomly selected Wikipedia articles. This resource, which represents the largest publicly available diacritized corpus to our knowledge, is made available to facilitate future research. Researchers may utilize this corpus directly or employ linguistic experts to refine its minimal errors for specialized applications.

6 Error Analysis

We conducted an error analysis on all WikiNews-2024 errors produced by our best model (n=243). Table 9 presents the most common error types, which account for 94% of all errors. The "POS Error" type occurs when the diacritizer assigns an incorrect part of speech, such as misidentifying passive vs. active tense. The "HUM-MRef" type arises when the reference contains only one diacritization form, whereas multiple forms are acceptable. The "HUM-Error" type corresponds to human annotation mistakes in assigning the correct diacritization within a given context; both "HUM-MRef" and "HUM-Error" will be corrected in the final released version. In the "Noun-Adj" type, Arabic grammar requires adjectives to match their reference noun in grammatical markers, but in complex cases (e.g., Noun Noun Adjective), the adjective can refer to any noun. The "CONJ" type pertains to cases where a word following a conjunction should conform to the structure of the preceding phrase, for example, the word "safety" should have the same grammatical diacritic mark as the word "speed" in the shown example. The "NE" type indicates errors in predicting diacritics for named entities, particularly foreign ones that are not seen in all the training data.[5] Finally, Arabic's relatively free word order allows the object to precede the subject, and the "SBJ-OBJ" type captures instances where the diacritizer failed to assign the correct grammatical markers distinguishing subject and object roles.

Type	%	System Output (Reference)-English
POS Error	20	إذا رَسَمَ الإمبراطور (رُسِمَ) **r**a**s**a**m**a (r**u**si**m**a) - draws (was drawn)
HUM-MRef	16	طاقِم الطائرة (طاقَم) TAq**i**m (TAq**a**m) - plane crew
Noun-Adj	14	وزيرُ الأمنِ القوميُّ (القوميِّ) Alqawmiy**u** (Alqawmiy**i**) - Minister of National Security
HUM-Error	12	في عامِ 2020 (عامَ) EAm**i** (EAm**a**) - in the year 2020
CONJ	12	سرعةُ التطويرِ والسلامةِ (والسلامةُ) - speed of development and safety
NE	12	نيوهامشِيرْ (نيوهامشَيَرْ) - New Hampshire
SBJ-OBJ	7	نفذت الضربةُ سفينةَ (الضربةَ، سفينةُ) "Carried-out the-strike a-ship" A ship carried out the strike.

Table 9: Common Error Types

7 Conclusion

The paper introduces a comprehensive approach to Arabic diacritization that improves dataset quality and evaluation methods. Key contributions include a novel BiLSTM model with a variable-length representation that outperforms existing systems on both WikiNews-2014 and WikiNews-2024 benchmarks, as well as effective data augmentation techniques for scenarios with limited training data. Additionally, the work presents a variant model that preserves user-specified diacritics without sacrificing accuracy. Future directions include exploring sliding window techniques, extending the approach to Arabic dialects, and leveraging transfer learning, with all code and benchmarks made publicly available to foster further research in the field.

[5]We plan to add more fully diacritized named entities from Wikipedia and other sources to the training data.

Limitations

Despite the advancements in Arabic diacritization, several challenges remain. First, ambiguity in diacritization persists, as multiple valid diacritical forms exist for many words depending on syntactic and semantic context. While our multi-reference evaluation mitigates this issue, it does not fully resolve the inherent uncertainty in diacritization.

Second, domain generalization remains a challenge. Our model performs well on benchmark datasets, but its accuracy may degrade when applied to out-of-domain texts, such as technical or scientific writing, which often exhibit unique linguistic patterns and new terms.

Third, annotation inconsistencies in training data impact model performance. Despite our efforts to refine the dataset using a human-in-the-loop approach, errors in existing corpora and variations in human annotation guidelines can still introduce noise into the training process.

Finally, computational efficiency is a concern. While deep learning models achieve high accuracy, they require significant computational resources, making real-time diacritization on low-resource devices challenging. Future work should explore more efficient architectures and compression techniques to enable wider adoption.

Addressing these limitations will be crucial for improving the robustness and applicability of Arabic diacritization models across diverse contexts.

Ethical Considerations

Arabic diacritization plays a crucial role in enhancing text readability, speech synthesis, and NLP applications. However, several ethical considerations must be addressed when developing and deploying diacritization models. I) Bias and Representation: Diacritization models are trained on specific datasets, which may not fully represent the diversity of Arabic writing styles, or domains. Bias in training data can lead to models favoring certain linguistic patterns over others, potentially disadvantaging underrepresented topics or genres. Ensuring diverse and balanced training data is essential to mitigate this issue. II) Misinformation and Misinterpretation: Incorrect diacritization can alter word meanings, leading to misinterpretation in critical contexts such as legal, religious, or medical texts. This risk underscores the need for rigorous evaluation and human oversight when deploying diacritization models in sensitive applications. III) Ethical Use in Automated Systems: Diacritization can enhance AI-driven applications such as automated content moderation, speech recognition, and machine translation. However, misuse—such as manipulating diacritization to evade content moderation or spread misinformation—must be considered. Developers should implement safeguards to detect and prevent such manipulations.

Addressing these ethical considerations is essential to ensure fairness, accuracy, and responsible deployment of Arabic diacritization models in real-world applications.

It's worth mentioning that the annotation was conducted by two expert linguists from Egypt (ages 36 and 52), both with extensive training. The linguists were selected from the professional network of one of the authors and had prior experience in similar annotation tasks. To ensure high-quality annotations, rigorous quality control sessions were implemented. The annotators were compensated at an average rate of $7 per hour for data creation and quality rounds, which exceeds typical pay rates for similar tasks, as verified through surveys on Glassdoor.com, Bayt.com, and direct feedback from the linguists regarding fair compensation.

Additionally, ChatGPT was used to refine certain sections, particularly the "Limitations" and "Ethical Considerations" sections, but not for generating content from scratch.

References

Gheith A Abandah, Alex Graves, Balkees Al-Shagoor, Alaa Arabiyat, Fuad Jamour, and Majid Al-Taee. 2015. Automatic diacritization of arabic text using recurrent neural networks. *International Journal on Document Analysis and Recognition (IJDAR)*, 18:183–197.

Hanan Aldarmaki and Ahmad Ghannam. 2023. Diacritic recognition performance in arabic asr. *arXiv preprint arXiv:2302.14022*.

Sawsan Alqahtani, Ajay Mishra, and Mona Diab. 2020. A multitask learning approach for diacritic restoration. *arXiv preprint arXiv:2006.04016*.

Wissam Antoun, Fady Baly, and Hazem Hajj. 2020. Arabert: Transformer-based model for arabic language understanding. *arXiv preprint arXiv:2003.00104*.

Yonatan Belinkov and James Glass. 2015. Arabic diacritization with recurrent neural networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2281–2285.

Kareem Darwish, Hamdy Mubarak, and Ahmed Abdelali. 2017. Arabic diacritization: Stats, rules, and hacks. In *Proceedings of the third Arabic natural language processing workshop*, pages 9–17.

Muhammad Morsy Elmallah, Mahmoud Reda, Kareem Darwish, Abdelrahman El-Sheikh, Ashraf Hatim Elneima, Murtadha Aljubran, Nouf Alsaeed, Reem Mohammed, and Mohamed Al-Badrashiny. 2024. Arabic diacritization using morphologically informed character-level model. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 1446–1454.

Moustafa Elshafei, Husni Al-Muhtaseb, and Mansour Alghamdi. 2006. Statistical methods for automatic diacritization of arabic text. In *The Saudi 18th National Computer Conference. Riyadh*, volume 18, pages 301–306.

Ya'akov Gal. 2002. An hmm approach to vowel restoration in arabic and hebrew. In *Proceedings of the ACL-02 workshop on Computational approaches to semitic languages*.

Nawar Halabi. 2016. *Modern standard Arabic phonetics for speech synthesis*. Ph.D. thesis, University of Southampton.

Mohamed Maamouri et al. 2010. *Arabic Treebank: Part 3 v3.2*. Linguistic Data Consortium, Philadelphia, PA. LDC2010T08, Web Download.

Hamdy Mubarak, Ahmed Abdelali, Kareem Darwish, Mohamed Eldesouki, Younes Samih, and Hassan Sajjad. 2019a. A system for diacritizing four varieties of arabic. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP): System Demonstrations*, pages 217–222.

Hamdy Mubarak, Ahmed Abdelali, Hassan Sajjad, Younes Samih, and Kareem Darwish. 2019b. 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)*, pages 2390–2395.

Rani Nelken and Stuart Shieber. 2005. Arabic diacritization using weighted finite-state transducers. In *Proceedings of the 2005 ACL Workshop on Computational Approaches to Semitic Languages*. Association for Computational Linguistics.

Arfath Pasha, Mohamed Al-Badrashiny, Mona T Diab, Ahmed El Kholy, Ramy Eskander, Nizar Habash, Manoj Pooleery, Owen Rambow, and Ryan Roth. 2014. Madamira: A fast, comprehensive tool for morphological analysis and disambiguation of arabic. In *Lrec*, volume 14, pages 1094–1101.

Mohsen AA Rashwan, Ahmad A Al Sallab, Hazem M Raafat, and Ahmed Rafea. 2015. Deep learning framework with confused sub-set resolution architecture for automatic arabic diacritization. *IEEE/ACM Transactions on Audio, Speech, And Language Processing*, 23(3):505–516.

Ahmed Said, Mohamed El-Sharqwi, Achraf Chalabi, and Eslam Kamal. 2013. A hybrid approach for arabic diacritization. In *International Conference on Application of Natural Language to Information Systems*, pages 53–64. Springer.

Brian Thompson and Ali Alshehri. 2021. Improving arabic diacritization by learning to diacritize and translate. *arXiv preprint arXiv:2109.14150*.

Catalin Ungurean, Dragos Burileanu, Vladimir Popescu, Cristian Negrescu, and Aurelian Dervis. 2008. Automatic diacritic restoration for a tts-based e-mail reader application. *UPB Scientific Bulletin, Series C*, 70(4):3–12.

Dimitra Vergyri and Katrin Kirchhoff. 2004. Automatic diacritization of arabic for acoustic modeling in speech recognition. In *Proceedings of the workshop on computational approaches to Arabic script-based languages*, pages 66–73.

Nasser Zalmout and Nizar Habash. 2019. Adversarial multitask learning for joint multi-feature and multi-dialect morphological modeling. *arXiv preprint arXiv:1910.12702*.

Imed Zitouni, Jeffrey Sorensen, and Ruhi Sarikaya. 2006. Maximum entropy based restoration of arabic diacritics. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 577–584.

A Description of Arabic Diacritics

In Arabic script, most vowels are not represented by separate letters. Instead, diacritics (**áarakāt**) indicate short vowels and other phonetic information. These zero-width marks attach above or below consonants. Table 10 lists the diacritics used in Modern Standard Arabic.

The three vowel diacritics—fatha, kasra and damma — represent the short sounds /a/, /i/ and /u/, respectively. The sukun signals the absence of a vowel. Nunation (tanwīn) adds a final "-n" sound to indefinite nouns by combining one of the short-vowel marks with the letter ن (nun, /n/).

The shadda marks consonant doubling (gemination). Unlike other diacritics, the shadda may be used together with any one of the vowel marks; no other combinations are permitted.

Diacritic	**Example (ت+diacritic)**	**Transliteration**
Fatha	تَ	t**a**
Kasra	تِ	t**i**
Damma	تُ	t**u**
Sukun	تْ	t
Shadda	تّ	t**t**
Tanwīn Fatá	تً	t**an**
Tanwīn Kasr	تٍ	t**in**
Tanwīn áamm	تٌ	t**un**

Table 10: List of the diacritics used in Modern Standard Arabic scripts along with an example of them applied to the letter ت (ta')

B Experimental Setup

Experimental Setup. All training and experiments were conducted on a machine with the specifications detailed in Table 11. A total of around 540 GPU hours were utilized across all experiments, with each experiment running for an average of 18 hours on a single GPU.

CPU	Intel(R) Xeon(R) CPU E5-2650 v4
GPU	Tesla P100-16GB
Memory	256GB
OS	Ubuntu 20.04.3 LTS

Table 11: Specification of machine used for training and experiments

C GPT-4o Test Prompt

The following prompt was used to diacritize WikiNews-2014 on GPT-4o (version 2024-08-06):

Task Instructions

SYSTEM: You are an expert in the Arabic language. You will be given a sentence in Arabic and you will be asked to fully diacritize it. Provide the diacritized version of the sentence in json format. Do not add or modify any of the punctuation or spaces in the text (e.g if a sentence does not end in a period, do not add a period). The output should be of the form

```
{'result' : <diacritized_sentence>}
```

USER: <INPUT>