Automatic Restoration of Diacritics for Speech Data Sets

Sara Shatnawi[1], **Sawsan Alqahtani**[2], **Hanan Aldarmaki**[1]
[1]Mohamed bin Zayed University of Artificial Intelligence
[2]Princess Nourah Bint Abdulrahman University
[1]{sara.shatnawi;hanan.aldarmaki}@mbzuai.ac.ae, [2]saalqhtani@pnu.edu.sa

Abstract

Automatic text-based diacritic restoration models generally have high diacritic error rates when applied to speech transcripts as a result of domain and style shifts in spoken language. In this work, we explore the possibility of improving the performance of automatic diacritic restoration when applied to speech data by utilizing parallel spoken utterances. In particular, we use the pre-trained Whisper ASR model fine-tuned on relatively small amounts of diacritized Arabic speech data to produce rough diacritized transcripts for the speech utterances, which we then use as an additional input for diacritic restoration models. The proposed framework consistently improves diacritic restoration performance compared to text-only baselines. Our results highlight the inadequacy of current text-based diacritic restoration models for speech data sets and provide a new baseline for speech-based diacritic restoration.

1 Introduction

The Arabic script consists of primary alphabetic characters and secondary diacritics. The alphabetic characters represent consonants and long vowels, while diacritics include short vowels, consonant doubling, and *nunation*, which is an additional 'n' sound that indicates indefinite nouns in standard and classical Arabic. Since these diacritics are peripheral rather than main alphabetical characters, proficient Arabic language users typically omit them. As a result, most text resources, including speech transcriptions, are non-diacritized, and currently available Arabic datasets are heavily under-specified for pronunciation. Contextual information provides a basis for implicitly filling out the missing information for proficient speakers, but new learners, nonnative speakers, as well as low-resource speech recognition and synthesis models, often struggle to identify the correct sense and pronunciation due to the lack of diacritics. Consequently, many tasks aimed at achieving higher performance necessitate diacritizing the script as a preliminary step. To address these challenges, text-based diacritic restoration models (i.e., models that take undiacritized text as input and produce diacritized output) have been employed to address the issue of missing diacritics for text and speech applications (Fashwan and Alansary, 2016; Fadel et al., 2019a,b; Al-Thubaity et al., 2020; AlKhamissi et al., 2020; Obeid et al., 2022). In speech applications, such as Automatic Speech Recognition (ASR) and Text-to-Speech synthesis (TTS), text-based diacritic restoration models have been employed in various ways. Some previous works employed automatic text-based diacritizers to restore the diacritics in speech transcriptions and train a diacritized ASR system (Al Hanai and Glass, 2014; Abed et al., 2019). Alternatively, ASR models can be trained without diacritics, and text post-processing can be employed afterwards to restore the diacritics (Aldarmaki and Ghannam, 2023). In TTS, datasets are curated carefully for phonetic coverage, so the training speech transcriptions are carefully annotated and manually diacritized. For instance, the Classical Arabic Text-To-Speech corpus (ClArTTS) (Kulkarni et al., 2023) was extracted from a recorded classical Arabic audiobook and was manually diacritized and verified to ensure consistency. The reliance on manual diacritization means that all available datasets for TTS are relatively small, making TTS a low-resource application in Arabic. Automatic diacritization is often used in deployed TTS systems to pre-process the text before synthesis.

Uni-modal text-based diacritic restoration models may not be optimal for speech applications (Aldarmaki and Ghannam, 2023). Speech utterances are typically less structured than text and may have unusual grammar, repetitions, or missing context, which results in domain and style shifts that lead to poor generalization of these models when applied on speech transcripts. Further, in ASR applications,

Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), pages 4166–4176

the output may contain transcription errors and misspellings that further sabotage the text diacritization models used in post-processing. The existence of paired text and speech data presents an opportunity for incorporating an additional modality for disambiguation and diacritic restoration. As shown in Aldarmaki and Ghannam (2023), ASR models trained to directly produce diacritics outperform text-based diacritizers applied on ASR outputs by a large margin. Given the existence of large speech datasets that contain paired undiacritized texts with speech audios, we explore the potential to improve the performance of automatic diacritic restoration for these datasets.[1] This kind of automatic diacritization could potentially enable the development of large diacritized speech corpora for both ASR and TTS. In particular, we look into whether the speech signal could facilitate more robust diacritization for speech-based datasets compared to text-only diacritic restoration models. To that end, we propose a diacritic restoration framework that incorporates a pre-trained diacritized speech recognition model. Our experiments show that the framework improves performance compared to an equivalent text-only model, which presents a promising direction for speech-based diacritic restoration. Our findings can be summarized as follows:

1. The proposed diacritic restoration framework results in lower diacritic error rates for read Classical Arabic speech compared to all text-only diacritizers, resulting in a 45% relative reduction in diacritic error rate compared to the best-performing baseline.

2. We experiment with both Transformer and LSTM-based architectures with similar scales. While both result in lower error rates compared to the text-only baselines, the LSTM model results in overall better performance.

3. The performance of the proposed framework partially depends on the performance of the ASR model used to produce the provisional diacritics. Since diacritized datasets are limited for Modern Standard Arabic (MSA) and Dialectal Arabic (DA), more work is needed to improve performance for these variants.

[1] As an example, the QASR data set contain ∼2000 hours of transcribed Arabic speech data, mostly undiacritized (Mubarak et al., 2021).

2 Related Work

2.1 Text-based diacritic restoration

Early approaches for Arabic diacritic restoration mainly relied on morphological rules. For example, Fashwan and Alansary (2016) proposed a rule-based approach to address the case ending diacritization problem in Modern Standard Arabic text. This system relied on morphological and syntactic analyses, taking into account the part-of-speech of each word and its position within the sentence. Morphological analysis has been used as the basis for diacritization in several models, such as MADAMIRA (Pasha et al., 2014) and the recent Camelira multi-dialectal morphological disambiguator (Obeid et al., 2022). In these systems, diacritic restoration is a result of complete morphological analysis and disambiguation, rather than a stand-alone objective. In recent years, researchers have investigated different neural network-based architectures for stand-alone Arabic diacritization systems. These methods do not rely on morphological analyzers, dictionaries, or feature engineering, but rather use sequence tagging frameworks, leveraging their ability to capture patterns in Arabic text implicitly through end-to-end training. Architectures include feed-forward networks (Fadel et al., 2019b), recurrent neural networks (RNNs) (Abandah and Abdel-Karim, 2020; Al-Thubaity et al., 2020; Fadel et al., 2019b), convolutional neural networks (CNNs) (Alqahtani et al., 2019), and bidirectional LSTM networks possibly followed by Conditional Random Fields (CRF) (Al-Thubaity et al., 2020). Some of the most commonly used toolkits and APIs for Arabic diacritization, including Farasa[2], ALI_Soft[3], Shakkala[4], Mishkal[5], and Camelira[6], do not clearly disclose model details or training data, but are often used in practical applications for their convenience.

2.2 Speech-based diacritic restoration

To the best of our knowledge, the use of speech data in automatic diacritization has rarely been explored in previous research. The work most closely related to ours in terms of problem formulation is Vergyri and Kirchhoff (2004), where they explore the use of acoustic and morphological information

[2] farasa.qcri.org/diacritization
[3] ali-soft.com
[4] github.com/Barqawiz/Shakkala
[5] github.com/linuxscout/mishkal
[6] camelira.abudhabi.nyu.edu

to automatically restore diacritics in dialectal Arabic speech. The proposed approach employs the EM algorithm to automatically optimize the best diacritic combination using either a morphological analyzer for generating all possible diacritics in an utterance, or using all possible diacritization without morphological constraints. The resulting diacritizations were used to construct a word pronunciation network for the acoustic model. The model achieved roughly 11% and 23% DER with and without morphological analysis, respectively, on the Egyptian CallHome corpus that contains diacritized transcripts in a romanized form. Using diacritics to train speech recognition or synthesis systems is desirable, but this requires manually diacritized training data to learn accurate mappings between acoustics and vowels. Aldarmaki and Ghannam (2023) demonstrated through controlled experiments that speech recognition models trained with manually diacritized data sets result in much higher diacritic recognition accuracy compared to text-based diacritic restoration models that are used before training (by diacritizing training speech transcripts) or after inference. This study underscores the importance of optimizing diacritic restoration performance for speech data as the text-based models were shown to have poor generalization in the speech domain.

3 Proposed Framework

Diacritic restoration is a sequence labeling task: the input is a sequence of Arabic characters without diacritics, and the output is the target diacritic for each input character, or 'no diacritic' if there are none. In our problem setting, we have a dataset that consists of speech utterances along with their undiacritized transcripts. Rather than applying a text-based diacritizer on the text transcripts in isolation, we propose a diacritic restoration framework that incorporates both speech utterances and their text transcripts to produce more accurate diacritics. The proposed model is illustrated in Figure 1. For the speech modality, we utilize a pre-trained ASR model to produce provisional diacritized transcripts [7]. The undiacritized text transcript and ASR hypothesis are fed into two separate sequence encoders of identical configuration.

To fuse information from the text and speech modalities, we apply cross-attention at the final

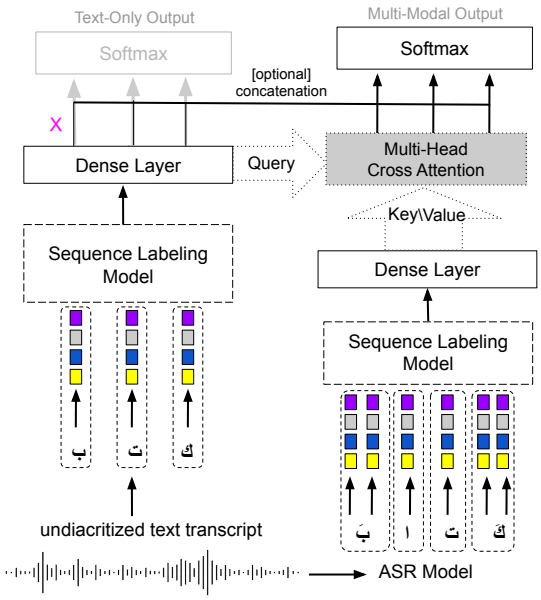

Figure 1: The proposed diacritic restoration model takes speech utterances and their undiacritized transcripts as input, and produces diacritized text. **Left**: text-only diacritic restoration, which can be any sequence labeling model. **Full figure:** Proposed framework, which includes a speech recognition model pre-trained to produce diacritized hypotheses, and a cross-attention mechanism to fuse the two modalities.

layer as follows: we use the outputs of the final dense layer of the text encoder (corresponding to the undiacritized text on the left side in Figure 1) as query vectors, and the outputs from the speech side as key and value vectors. Note that ASR predictions are longer than the raw text due to the presence of diacritics and other ASR errors, but the cross-attention mechanism ensures that the final output matches the length of the original undiacritized text. The outputs of cross-attention can be used directly as inputs to the final linear layer with softmax activation for sequence labeling. Note that in this configuration, the prediction relies heavily on the representations from the speech side which contributes the value vectors. To increase the contribution from the text modality, the output of cross-attention can be concatenated with the output of the text encoder (denoted as X in the figure).

Notation: For the rest of the paper, we will refer to this proposed framework as `Text+ASR`; diacritic restoration models that only rely on the undiacritized text transcripts (as is the case for all the existing baselines) will be referred to as `Text-Only`.

[7] 'provisional' because they contain ASR errors that distort both consonants and vowels.

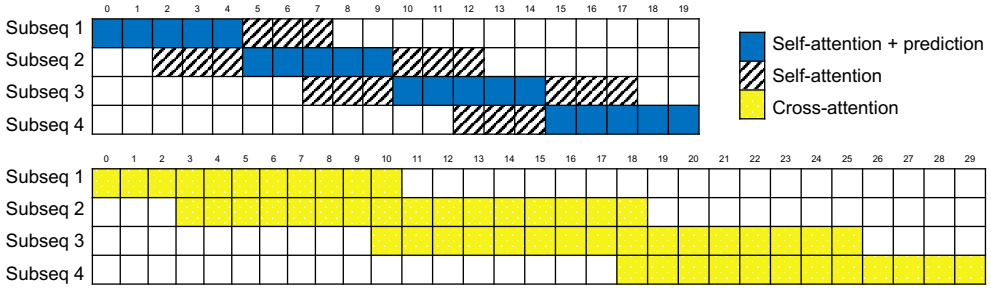

Figure 2: **Top**: Basic transformer self attention and prediction regions with sliding window mechanism for inference. **Bottom:** Cross-attention region from ASR prediction used in each sub-sequence.

3.1 Sequence Encoder Architecture

The proposed `Text+ASR` model is conceptually agnostic to the backbone architecture used for speech and text encoders. In this work, we investigated two different architectures to evaluate the efficacy of different sequential models on the proposed approach: Transformer (Vaswani et al., 2017) and bidirectional LSTM (Hochreiter and Schmidhuber, 1997; Schuster and Paliwal, 1997). Note that we use the same architecture for both encoders.

Transformer Model: We utilize the Transformer encoder architecture with position encoding and multi-head self-attention to enable contextual integration across the whole sentence. We employ absolute position embeddings with learned parameters that are optimized along with the rest of the network. We use multiple transformer blocks with multi-head self-attention, followed by a dense layer before the softmax sequence classification.

bi-LSTM Model: We use the same architecture described in Fadel et al. (2019b), which achieves the best diacritic restoration performance on our test set. The model consists of an embedding layer followed by several bidirectional LSTM layers. To prevent overfitting, dropout layers are inserted after each bidirectional LSTM layer. The model employs two layers of time-distributed dense units with ReLU activation before the final output layer.

3.2 Sliding Window Inference

The trained Transformer model with absolute position embeddings can handle sequences up to the maximum length used for training. This kind of position encoding has a poor generalization to longer sequence lengths.[8] Furthermore, given the nature of the task, attention to surrounding characters, as well as corresponding regions in the ASR prediction side, is more useful than global attention across two long sequences. Given these reasons, we employ a sliding window mechanism to handle long sequences at inference time and avoid any potential generalization issues. This mechanism is described concretely in Algorithm 1 and illustrated in Figure 2 with $window = 5$ and $buffer = 3$. Note that this is conceptually similar to the sliding window attention employed in the Longformer model (Beltagy et al., 2020) but we also proportionally handle cross-attention with the longer ASR sequence, and we apply the method only for inference. For the `Text-Only` model, we employ the sliding window inference, but without the ASR part (i.e. only the top part of Figure 2).

Algorithm 1 Sliding Window Inference

Input: $model, origText, asrText, window, buffer$
Output: $predictions$

$len \leftarrow \emptyset$
$len \leftarrow length(origText)$
$r \leftarrow length(asrText)/len$
$start \leftarrow 0$
while $start < len$ **do**
  $safeStart \leftarrow \max(start - buffer, 0)$
  $end \leftarrow start + window + buffer$
  $end \leftarrow \min(end, len)$
  $s1 \leftarrow origText[safeStart : end]$
  $s2 \leftarrow asrText[safeStart * r : end * r]$
  $result \leftarrow model.predict(s1, s2)$
  $end \leftarrow max(start + window, len)$
  $predictions.insert(result[start : end])$
  $start \leftarrow end$
end while
return $predictions$

[8] Most types of absolute and relative positional embeddings have poor length generalization, as shown in (Kazemnejad et al., 2024).

4 Experimental Settings

4.1 Datasets

A commonly-used corpus is the cleaned **Tashkeela** Corpus; this dataset comprises a large collection of Arabic texts mainly from Classical Arabic (CA) literature and religious texts, and a smaller collection in the Modern Standard Arabic (MSA) variety. It consists of 2.3M words spread over 55K lines, which is extracted from the **original Tashkeela** (Zerrouki and Balla, 2017) corpus, which has a total of 75M words. We refer to the latter as Tashkeela (Original) but do not use it for training. No speech audios are available in this dataset so we use it for experiments relevant to the `Text-Only` models. Diacritized speech data sets are rather scarce. We mainly use the CLassical Arabic Text-To-Speech Corpus **CLArTTS** (Kulkarni et al., 2023), which was developed for the purpose of text-to-speech synthesis, so the text transcriptions have been manually diacritized and verified. The corpus includes approximately 12 hours of recorded speech from a single male speaker (10K relatively short utterances), and about 30 minutes held-out for testing. Since the corpus consists of read Classical Arabic speech, it is consistent with the content of the Tashkeela corpus to enable fair comparison with text-based models.

4.2 Model Setup

For the Transformer architecture, we tuned the following hyper-parameters on the ClArTTS validation set: number of transformer blocks, number of attention heads, embedding size, and dropout rate. The reported results are obtained using 128-dimensional token and position embeddings, and 2 Transformer blocks with a feed-forward layer of size 128. We used 4 heads for multi-head attention and 0.2 dropout rate. For the LSTM model, we used 128-dimensional vectors for all layers, which include an embedding layer, two bi-LSTM layers, and two dense layers before the final output. Dropout rate of 0.5 was applied after each bi-LSTM. This is similar to the model used in the Shakkelha model (Fadel et al., 2019b), which is the strongest baseline on the ClArTTS test set. For optimization, we used the Adam optimizer with categorical cross-entropy loss. For sliding-window inference, we used a window size 50 and a buffer of 25 tokens on each side.[9]

All models were implemented using the Keras Python library and trained on one Nvidia A100 GPU with 40GB memory. The total number of parameters is ∼700K for the `Text-Only` Transformer and LSTM models and ∼1.5M for the `Text+ASR` variants.

4.3 ASR Model

For the ASR module used in the `Text+ASR` framework, we fine-tuned Whisper (Radford et al., 2023),[10] using the training set of the ClArTTS corpus with diacritics. An analysis of this model is provided in Aldarmaki and Ghannam (2023), but due to apparent inconsistency in their use of ClArTTS train/test splits compared to the official dataset splits,[11] we fine-tuned the model from scratch to avoid data leakage.[12]

4.4 Baselines

We compare our proposed `Text+ASR` model to the `Text-Only` baselines that use comparable or similar training dataset. In addition to the Transformer and LSTM models we train, we use popular diacritic restoration models such as the Shakkala model (Barqawi, 2017), which was trained with the original Tashkeela corpus of 75M words. Our implementation of the LSTM `Text-Only` model follows the Shakkelha model described in Fadel et al. (2019b). Shakkelha also includes a variant trained with extra data of 22M words extracted from the original Tashkeela corpus and the Holy Quran. We use it as another baseline that is trained with much more data. For a comprehensive evaluation, we also include results from other popular diacritization APIs even though their training details and datasets are not disclosed, but are worth including due to their popularity: Mishkal (Zerrouki, 2020), ALI-Soft (URL, 2023), Farasa (QCRI, 2020), and Camelira (Obeid et al., 2022).

5 Results & Analysis

Table 1 shows the results for our proposed model compared to the baselines using Diacritic Error Rates (DER). We report performance including and excluding the 'no diacritic' tag, with and without case ending diacritics.

[9]The code for running our experiments is available at `https://github.com/SaraShatnawi/Diacritization.git`

[10]`huggingface.com/openai/whisper-medium`

[11]We obtained the splits from `www.clartts.com`.

[12]The fine-tuned model is available at `huggingface.co/sashat/whisper-medium-ClassicalAr`

Model	Train Set	Including 'no diacritic'		Excluding 'no diacritic'	
		w. case ending	w.o case ending	w. case ending	w.o case ending
Mishkal (Zerrouki, 2020)	-	21.79	15.80	24.70	17.09
ali-soft.com	-	50.74	47.96	60.03	55.62
Farasa (QCRI, 2020)	-	19.17	22.05	22.64	26.53
Camelira (Obeid et al., 2022)	-	29.16	27.36	33.67	32.29
Shakkala (Barqawi, 2017)	Tashkeela (Original)†	6.85	5.48	8.02	6.63
Shakkelha (Fadel et al., 2019b)	Tashkeela	6.02	4.87	6.89	5.70
	Tashkeela + Extra‡	4.87	3.54	5.93	4.32
Text-Only Transformer	Tashkeela	11.73	10.00	13.22	12.39
	CLArTTS	20.42	16.96	24.70	20.27
	Tashkeela + CLArTTS	9.63	7.38	11.61	8.91
Text+ASR Transformer	CLArTTS	5.66	4.52	6.64	5.45
	Tashkeela + ClArTTS	**3.63**	**2.71**	**4.07**	**3.17**
Text-Only LSTM	Tashkeela	6.10	4.74	6.97	5.55
	CLArTTS	7.97	6.60	9.58	7.97
	Tashkeela + CLArTTS	4.93	3.55	5.86	4.30
Text+ASR LSTM	CLArTTS	2.97	2.18	3.03	2.30
	Tashkeela + ClArTTS	**2.70**	**1.83**	**2.85**	**1.99**

Table 1: Diacritic Error Rate (DER) % for Text+ASR model and the Text-Only models. Baselines that did not disclose details about their training data are shown in Grey background. Baselines that are trained on variants of the Tashkeela corpus are shown in Yellow. † refers to the original Tashkeela corpus of 75M words. ‡ extra data of 22M words. Tashkeela refers to the subset of 2.3M words from (Fadel et al., 2019a), which is the one we use for training our models, in addition to the ClArTTS train set. **Bold** scores refer to the models that have the lowest DER within comparable models with the same architecture. Underlined scores refer to the lowest DER overall.

Performance of previous Text-Only baselines: As seen from Table 1 (grey rows), high error rates are attained by the text-based APIs: Mishakal, ALI-Soft, Farasa, and Camelira on our test set. Baselines shown in yellow in Table 1, which are trained on variants of the Tashkeela corpus, achieved the lowest DER overall. As previously mentioned, we trained LSTM Text-Only model following the same model and training data as Shakkelha (Tashkeela) to perform our experiments and obtained comparable performance (compared to our Text-Only LSTM model trained on Tashkeela). Shakkelha also has a variant trained with 22M extra words (Tashkeela + Extra) which achieves better performance, demonstrating the effect of dataset size and quality on performance.

Performance of proposed models: Our proposed Text+ASR framework results in the best performance overall across all metrics, reducing absolute DER by a large margin compared to their equivalents Text-Only models and compared to all baselines. The Text+ASR model trained on ClArTTS only, which is a much smaller data set compared to Tashkeela, outperformed most other text-based baselines; and outperformed the best performing Shakkelha model with extra training data when using LSTM (and approached the performance when using Transformer). This clearly demonstrates the advantage of using features from speech modality to improve diacritic restoration for speech data. Compared to the Text-Only model trained with ClArTTS alone, the Text+ASR model improved performance by more than 15% absolute DER when using Transformer and 5% absolute DER when using LSTM. Further, fine-tuning the Text-Only model trained on Tashkeela with ClArTTS dataset (Tashkeela + ClArTTS) significantly boosts its performance by around 8% using Transformers and 3.4% using LSTM. Using the Text+ASR) framework, using the combined data improves performance by around 2% in Transformer but provides similar performance in LSTM compared to using CLaRTTS only. Overall, LSTM shows better performance compared to the Transformer model. The best performing Text-Only model is the LSTM architecture trained with Tashkeela and fine-tuned on ClArTTS train set, which achieves 4.93% DER including case ending and 'no diacritic', while the equivalent Text+ASR model achieves 2.7% DER, a 45% relative improvement.

5.1 Analysis

Text+ASR cross attention layer: The cross-attention mechanism plays a major role in integrating information from text and speech inputs to produce the final diacritics. Figure 3 shows the cross-attention weights between the raw text and

Figure 3: Cross-attention weights between the undiacritized input (black) and ASR text (magenta).

ASR hypothesis obtained from one of the 4 attention heads. We can observe that the weights are higher around the most relevant regions for prediction, regardless of ASR errors. Table 2 show an example of a phrase correctly diacritized by the `Text+ASR` model trained on ClArTTS while the Text-Only models struggle to identify the correct form of the ambiguous phrase.

Reference	وَسَرَابِيلَ تَقِيكُمْ بَأْسَكُمْ
Camelira (Obeid et al., 2022)	وَسَرَابِيلُ تَقِيكُمْ بِأَسْكُمِ
Shakkala (Barqawi, 2017)	وَسَرابِيلُ تَقِيكُم بَأْسَكُم
Text-Only[ClArTTS]	وَسِرَّابِيلٌ تَقِيكُمْ بِأَسْكَمْ
Text+ASR[ClArTTS]	وَسَرَابِيلَ تَقِيكُمْ بَأْسَكُمْ

Table 2: Diacritization from different models of a phrase from the ClArTTS test set.

5.2 Ablation Experiments

Impact of concatenation at the last layer: We experimented with or without concatenating the outputs of cross-attention layer with the outputs of the `Text-Only` encoder. As shown in Table 4, the effect of concatenation varies depending on the encoder architecture. For Transformer, relying solely on the cross-attention outputs provides slightly better performance. On the other hand, the bi-LSTM shows significantly better results with the concatenation. We surmise that this is due to the overall better features obtained from the `Text-Only` model using bi-LSTM compared to Transformer models.

Impact of the sliding window for inference: In Section 3.2, we described a sliding window approach to handle longer sequence lengths at inference. We evaluated the performance both before and after applying this approach. Table 5 shows notable improvement in results following the application of the proposed inference method.

5.3 Additional Experiments on MSA

As discussed in Section 4.1, ClArTTS dataset consists of an audiobook in Classical Arabic, and the Tashkeela corpus is derived mostly from Classical Arabic texts. We performed additional experiments to inspect the generalization of the diacritic restoration models trained on these data sets to other variants of Arabic and other speech genres. Diacritized datasets in other Arabic variants are rather scarce, but a small manually diacritic dataset derived from broadcast news has recently been released (Baali et al., 2023). This dataset (dubbed QASR TTS)[13] was manually diacritized for TTS and contains one hour of speech by a male speaker and one hour of speech by a female speaker. Note that this dataset contains MSA speech (as opposed to classical Arabic in CLArTTS), has a different genre (broadcast news vs. audiobook), and may have inconsistent speaking and pronunciation styles due to the more casual nature of the recordings.

The results are shown in Table 3. We show the results separately for the male and female speakers. We observe that all models, including all `Text-Only` baselines, perform poorly on this dataset. While the `Text+ASR` framework results in some reduction in error rates compared to `Text-Only` models with identical architectures, the improvements are not consistent or substantial enough. Sources of errors include lexical, domain, and style shifts in the text itself as well as ASR errors. As reported in Table 6, the pre-trained ASR model results in much higher error rates on QASR sets compared to our Classical Arabic test set.

As an illustrative example, Table 7 shows how ASR performance affects the result of the `Text+ASR` model in complex ways, sometimes leading to better and sometimes worse performance compared to `Text-Only`, which eventually leads to similar error rates. Furthermore, all diacritic restoration models used as baselines perform poorly on this set. While we do not know

[13] `arabicspeech.org/qasr_tts`

Model	Train Set	Including 'no diacritic'		Excluding 'no diacritic'	
		w. case ending	w.o case ending	w. case ending	w.o case ending
Test Set: Male Speaker from QASR TTS					
Mishkal (Zerrouki, 2020)	-	25.24	14.47	25.45	12.87
ali-soft.com	-	41.68	36.63	30.90	32.78
Farasa (QCRI, 2020)	-	28.81	28.81	21.39	5.60
Camelira (Obeid et al., 2022)	-	30.70	22.64	22.65	12.15
Shakkala (Barqawi, 2017)	Tashkeela (Original)†	19.79	11.55	24.12	13.12
Shakkelha (Fadel et al., 2019b)	Tashkeela	20.28	12.14	24.83	13.99
	Tashkeela + Extra‡	19.59	11.49	23.93	13.13
Text-Only Transformer	Tashkeela	23.81	16.06	27.19	16.14
	CLArTTS	34.31	34.31	40.91	31.28
	Tashkeela + CLArTTS	29.28	23.84	33.68	25.59
Text+ASR Transformer	CLArTTS	24.25	16.56	28.22	17.20
	Tashkeela + ClArTTS	**21.69**	**14.44**	**24.55**	**14.16**
Text-Only LSTM	Tashkeela	20.20	**12.02**	22.59	**10.95**
	CLArTTS	24.93	17.54	28.93	18.15
	Tashkeela + CLArTTS	20.67	12.63	23.49	12.04
Text+ASR LSTM	CLArTTS	21.25	14.13	22.95	12.76
	Tashkeela + ClArTTS	**19.82**	12.34	**21.98**	11.24
Test Set: Female Speaker from QASR TTS					
Mishkal (Zerrouki, 2020)	-	36.28	27.30	42.62	32.51
ali-soft.com	-	42.80	38.39	43.98	45.93
Farasa (QCRI, 2020)	-	31.10	22.29	30.14	23.69
Camelira (Obeid et al., 2022)	-	32.40	23.15	34.72	24.86
Shakkala (Barqawi, 2017)	Tashkeela (Original)†	34.65	28.63	43.21	35.14
Shakkelha (Fadel et al., 2019b)	Tashkeela	34.99	29.04	43.56	35.56
	Tashkeela + Extra‡	34.01	27.86	42.39	34.15
Text-Only Transformer	Tashkeela	38.10	32.69	35.27	21.77
	CLArTTS	46.48	42.56	47.96	37.20
	Tashkeela + CLArTTS	39.19	35.35	37.27	26.19
Text+ASR Transformer	CLArTTS	38.59	33.28	36.62	23.17
	Tashkeela + ClArTTS	**35.69**	**30.72**	**31.84**	**18.85**
Text-Only LSTM	Tashkeela	35.09	**29.09**	30.60	16.01
	CLArTTS	38.26	33.25	36.04	22.82
	Tashkeela + CLArTTS	35.33	29.53	31.45	17.05
Text+ASR LSTM	CLArTTS	35.59	30.69	30.25	17.37
	Tashkeela + ClArTTS	**34.06**	29.11	**29.12**	**15.82**

Table 3: Diacritic Error Rate (DER) % of QASR (Male and Female) dataset for Text+ASR (text + ASR) and the text-only models. Baselines that did not disclose details about their training data are shown in Grey background. Baselines that are trained on variants of the Tashkeela corpus are shown in Yellow. † refers to the original Tashkeela corpus of 75M words. ‡ extra data of 22M words. Tashkeela referes to the subset of 2.3M words from (Fadel et al., 2019a), which is the one we use for training our models, in addition to the ClArTTS train set. **Bold** scores refer to the models that have the lowest DER within comparable models with the same architecture. Underlined scores refer to the lowest DER overall.

Model	Concatenation	Including 'no diacritic'		Excluding 'no diacritic'	
		w. case ending	w.o case ending	w. case ending	w.o case ending
Transformer	✓	3.83	2.78	4.19	**3.10**
Transformer	×	**3.63**	**2.71**	**4.07**	3.17
LSTM	✓	**2.70**	**1.83**	**2.85**	**1.99**
LSTM	×	7.00	5.96	8.06	7.05

Table 4: Diacritic Error Rate (DER) scores on test set with and without concatenation at the last layer. Results are shown for the Text+ASR model trained on Tashkeela + ClArTTS, but the same trend is observed for all variants.

		Including 'no diacritic'		Excluding 'no diacritic'	
Model	**SW Inference**	**w. case ending**	**w.o case ending**	**w. case ending**	**w.o case ending**
Transformer	✓	**3.63**	**2.71**	**4.07**	**3.17**
Transformer	×	4.12	3.11	4.61	3.54
LSTM	✓	**2.70**	**1.83**	**2.85**	**1.99**
LSTM	×	2.88	2.07	2.94	2.17

Table 5: Diacritic Error Rate (DER) scores on test set with and without applying sliding window inference. Results are shown for the Text+ASR model trained on Tashkeela + ClArTTS, but the same trend is observed for all variants.

the full sources used to trained the diacritization APIs, we know that the majority of content in Tashkeela consists of Classical Arabic text, and MSA text is estimated to be only 1.15% of the corpus (Zerrouki and Balla, 2017). This shift in the Arabic variant, which results in both lexical and stylistic variations, is likely to be the leading cause for the poor generalization of these models in this dataset.

Dataset	Without Diacritics		With Diacritics	
	CER	WER	CER	WER
ClArTTS Test	2.20	8.02	2.90	14.43
QASR Female	11.6	36.9	27.5	87.3
QASR Male	11.1	36.4	21.06	72.4

Table 6: ASR Character Error Rate (CER) % and Word Error Rate (WER) % using the Whisper model fine-tuned on ClArTTS training set.

Reference	سَلَّمَتْ إِذَنْ الْأَحْزابُ الْمِصْرِيَّةُ الْمُبارِكَةُ لِخَريطَةِ الطَّريقْ
ASR	سَلَّمَتْ إِذَنَّ الْأَحْزَابُ الْمُصْرِيَّةُ الْمُبَارَكَةُ لِخَرِيطَةِ الطَرِيقِ
Text-Only	سُلَمْت إِذَنْ الْأَحْزَابُ الْمُصْرِيَّةُ الْمُبَارَكَةُ لِخُرِيطَةِ الطَّرِيقَ
Text+ASR	سَلَّمَتْ إِذَنَّ الْأَحْزَابُ الْمُصْرِيَّةُ الْمُبَارَكَةُ لِخَرِيطَةِ الطَرِيقِ

Table 7: Diacritization of a phrase from QASR TTS test set. The diacritic restoration models are our best models trained on both Tashkeela and ClArTTS.

6 Discussion & Conclusion

We described a framework for diacritic restoration that incorporates input from speech utterances as well as text to improve the performance of diacritic restoration models applied to speech data. The model consists of a pre-trained ASR model that produces provisional diacritized hypotheses, which are incorporated into a sequence labeling model via a cross-attention mechanism. The proposed framework can be applied to any sequence labeling model, and we experimented with transformer and LSTM architectures. We also proposed a sliding window inference method that improves the length generalization of the model so it can be applied more robustly to longer sequences. The proposed framework consistently improved diacritization performance compared to an equivalent text-only model, leading to significant reductions in diacritic error rates on our test set compared to all existing diacritic restoration models. We observed particular performance gains on the Classical Arabic test set which is consistent with the data used for fine-tuning the diacritized ASR and the diacritic restoration models. The model outperformed all text-based baselines that are trained with much larger text data sets, achieving a 45% relative reduction in diacritic error rates compared to the best performing baseline. However, given the high variability in speech datasets, additional manually-diacritized datasets are required to enable broader generalization across Arabic variants (such as MSA and dialectal Arabic) and genres (e.g. newswire or casual speech). Our results show that highly accurate diacritization can be obtained using a relatively small diacritized speech data set used for training the ASR model and the diacritic restoration models, which can facilitate the production of larger diacritized corpora for speech applications.

Limitations

The main limitation of this work is the lack of diacritized speech data sets that could be used for training and testing the framework. While we observed strong performance on the Classical Arabic test set, we could not develop similar models for other variants of Arabic due to limited resources. In addition, we could not evaluate other classical Arabic test sets since we are not aware of any available sets beyond the Quranic data set, which intersects with the Tashkeela corpus. In our analysis, the results appear to be sensitive to ASR performance, which in turn is a factor of the type of training data used to fine-tune the ASR model.

References

Gheith Abandah and Asma Abdel-Karim. 2020. Accurate and fast recurrent neural network solution for the automatic diacritization of arabic text. *Jordanian Journal of Computers and Information Technology*, 6(2).

Sa'ed Abed, Mohammad Alshayeji, and Sari Sultan. 2019. Diacritics effect on arabic speech recognition. *Arabian Journal for Science and Engineering*, 44:9043–9056.

Tuka Al Hanai and James R Glass. 2014. Lexical modeling for arabic asr: a systematic approach. In *INTERSPEECH*, pages 2605–2609.

Abdulmohsen Al-Thubaity, Atheer Alkhalifa, Abdulrahman Almuhareb, and Waleed Alsanie. 2020. Arabic diacritization using bidirectional long short-term memory neural networks with conditional random fields. *IEEE Access*, 8:154984–154996.

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

Badr AlKhamissi, Muhammad N ElNokrashy, and Mohamed Gabr. 2020. Deep diacritization: Efficient hierarchical recurrence for improved arabic diacritization. *arXiv preprint arXiv:2011.00538*.

Sawsan Alqahtani, Ajay Mishra, and Mona Diab. 2019. Efficient convolutional neural networks for diacritic restoration. 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)*, pages 1442–1448, Hong Kong, China. Association for Computational Linguistics.

Massa Baali, Tomoki Hayashi, Hamdy Mubarak, Soumi Maiti, Shinji Watanabe, Wassim El-Hajj, and Ahmed Ali. 2023. Unsupervised data selection for tts: Using arabic broadcast news as a case study. *arXiv preprint arXiv:2301.09099*.

Zerrouki Barqawi. 2017. Shakkala, arabic text vocalization.

Iz Beltagy, Matthew E Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*.

Ali Fadel, Ibraheem Tuffaha, Mahmoud Al-Ayyoub, et al. 2019a. Arabic text diacritization using deep neural networks. In *2019 2nd international conference on computer applications & information security (ICCAIS)*, pages 1–7. IEEE.

Ali Fadel, Ibraheem Tuffaha, Bara' Al-Jawarneh, and Mahmoud Al-Ayyoub. 2019b. Neural Arabic text diacritization: State of the art results and a novel approach for machine translation. In *Proceedings of the 6th Workshop on Asian Translation*, pages 215–225, Hong Kong, China. Association for Computational Linguistics.

Amany Fashwan and Sameh Alansary. 2016. A rule based method for adding case ending diacritics for modern standard arabic texts. In *16th International Conference on Language Engineering. The Egyptian Society of Language Engineering (ESOLE)*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Amirhossein Kazemnejad, Inkit Padhi, Karthikeyan Natesan Ramamurthy, Payel Das, and Siva Reddy. 2024. The impact of positional encoding on length generalization in transformers. *Advances in Neural Information Processing Systems*, 36.

Ajinkya Kulkarni, Atharva Kulkarni, Sara Abedalmonem Mohammad Shatnawi, and Hanan Aldarmaki. 2023. Clartts: An open-source classical arabic text-to-speech corpus. *arXiv preprint arXiv:2303.00069*.

Hamdy Mubarak, Amir Hussein, Shammur Absar Chowdhury, and Ahmed Ali. 2021. Qasr: Qcri aljazeera speech resource a large scale annotated arabic speech corpus. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 2274–2285.

Ossama Obeid, Go Inoue, and Nizar Habash. 2022. Camelira: An arabic multi-dialect morphological disambiguator. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 319–326.

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.

QCRI. 2020. Farasa api diacritization module. Accessed on October 12, 2022.

Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. 2023. Robust speech recognition via large-scale weak supervision. In *International Conference on Machine Learning*, pages 28492–28518. PMLR.

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE transactions on Signal Processing*, 45(11):2673–2681.

URL. 2023. Ali-soft. Accessed on October 12, 2023.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *CoRR*, abs/1706.03762.

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.

Taha Zerrouki. 2020. Towards an open platform for arabic language processing.

Taha Zerrouki and Amar Balla. 2017. Tashkeela: Novel corpus of arabic vocalized texts, data for auto-diacritization systems. *Data in brief*, 11:147–151.

A Computational Requirements

We trained our models using one NVIDIA A100-SXM4 GPU with 40GB memory. The GPU processing time for our experiments varies, ranging from 30 minutes to 3 hours, depending on the specific model and dataset employed. Specifically, models utilizing the Tashkeela dataset require approximately three hours for training when implemented with a Transformer architecture and 1 hour and 30 minutes when utilizing LSTMs. On the other hand, experiments with the CLArTTS dataset exhibit a faster processing time, typically ranging from 40 to 60 minutes for Transformer-based models and half of this time for those employing LSTM.