Eraserhead at OSACT7 Shared Task: ASR Consistency Filtering and Speaker-Adaptive Post-Processing for Arabic Speech Diacritization

Muhammad Abu Horaira, Nahian Chowdhury
Chittagong University of Engineering and Technology
Chittagong, Bangladesh
{u2004029,u2004026}@student.cuet.ac.bd

Abstract

Arabic speech diacritization is the task of restoring short vowel marks to undiacritized text derived from speech input. It remains difficult because ASR output can be noisy, dialectal variation is substantial, and speakers often differ in how they realize word-final diacritics. In this paper, we describe our submission to Task 2 of the KSAA-2026 Shared Task on Arabic Speech Dictation with Automatic Diacritization, where our system ranked 4th on the official leaderboard. Our approach builds on a pretrained ASR-aware diacritization model and adds three components: ASR Consistency Filtering, confidence-based ensembling of three checkpoints, and speaker-adaptive post-processing specifically for word-final diacritics. Rather than discarding problematic data, our filtering strategy replaces unreliable ASR transcripts with the undiacritized gold text rather than removing training examples, which makes training more stable. On the official test set, our system achieved a Diacritic Error Rate (DER) of 8.23, a Word Error Rate (WER) of 30.37, and a Sentence Error Rate (SER) of 80.79 under the With Case Endings (WCE), Including No Diacritic (Incl. 0) evaluation setting. It also outperformed the organizers' fine-tuned Text+ASR baseline in three of the four main evaluation settings.

Keywords: Arabic diacritization, automatic speech recognition, speech-aware diacritization, consistency filtering

1. Introduction

Arabic diacritization, the task of restoring of short vowels and symbols to undiacritized text, remains a significant challenge in Arabic NLP. Their absence in most writing creates lexical and syntactic ambiguity for both humans and systems (Elgamal et al., 2024). Hierarchical neural approaches operating at both word and character levels have shown strong results on benchmark text diacritization datasets (AlKhamissi et al., 2020). However, text-based diacritization models are less effective on speech transcripts, where spoken-language variation leads to substantially higher error rates (Shatnawi et al., 2024). This challenge becomes even greater in ASR output, which may contain recognition errors, spurious tokens, and dialectal variation that diverges from the assumptions of text-only diacritizers.

The KSAA-2026 Shared Task on Arabic Speech Dictation with Automatic Diacritization (Al Wazrah et al., 2026) (Task 2) targets this problem directly. Given a speech utterance and its undiacritized transcript, systems must produce a fully diacritized output at the character level, predicting all diacritic marks, including fatha, damma, kasra, sukūn, shaddah, and tanwīn. The dataset covers Modern Standard Arabic and dialectal speech from multiple speaker backgrounds, introducing additional variation in diacritic realization.

We present a system built on the Diac Transformer (Shatnawi et al., 2024) and make the following contributions:

- We introduce *ASR Consistency Filtering*, which replaces unreliable ASR transcripts with the undiacritized gold text based on character-level overlap, rather than discarding training examples.
- We train three checkpoints, two with different filtering thresholds (τ=0.5 and τ=0.9) and one with τ=0.5 and a different random seed, and combine them using character-level confidence-voting ensemble decoding.
- We apply speaker-adaptive post-processing that normalizes word-final diacritics based on per-speaker stylistic patterns estimated from development-set annotations.

Our system achieved 8.23 DER and 30.37 WER on the primary test setting, ranking 4th on the official leaderboard and outperforming the organizers' fine-tuned Text+ASR baseline in three of four evaluation conditions.

2. Related Work

Arabic diacritization has been widely studied using neural sequence models. Fadel et al. (2019) reported strong results on the Tashkeela benchmark, while Mubarak et al. (2019) showed that sequence-to-sequence models with voting can outperform earlier feature-based approaches. Later, Qin et al. (2021) showed that even noisy auxiliary information, such as automatically generated diacritizations, can still be useful when combined with regularized decoding and adversarial training. Similarly,

Proceedings of The 7th Workshop on Open-Source Arabic Corpora and Processing Tools (OSACT7) with 5 Shared Tasks, pages 247–251
May 11, 2026.

Thompson and Alshehri (2022) showed that multi-task learning with translation as an auxiliary task can improve Arabic diacritization by injecting additional semantic and linguistic information. These results suggest that external or imperfect auxiliary signals can still be beneficial when handled carefully, which is closely related to the motivation behind our ASR consistency filtering.

Recent work has also explored the use of speech-related information for Arabic diacritization. Shatnawi et al. (2024) used diacritized ASR transcripts as a second input through cross-attention and reported substantial reductions in diacritic error rates on speech data. Aldarmaki and Ghannam (2023) also showed that ASR-aware diacritization can outperform applying text-only diacritization as a post-processing step. Building on this line of work, our system adds consistency-based filtering for noisy ASR inputs and speaker-adaptive post-processing for word-final diacritics.

3. Dataset

We used the official KSAA-2026 shared-task dataset. The corpus contains approximately five hours of Arabic speech covering Modern Standard Arabic and dialectal speech from nine nationalities. The official splits consist of 2,327 training utterances ($\approx$4.5 h), 260 development utterances ($\approx$0.5 h), and 260 test utterances ($\approx$0.5 h). All recordings are 16 kHz mono WAV files of at most nine seconds. Task 2 is defined as speech dictation with automatic diacritization, where systems take speech audio together with undiacritized text and produce a fully diacritized transcript. In our experiments, the diacritized references provided in the official data were used as supervision, and undiacritized text was obtained by removing diacritic marks when needed. The cross-dialect and cross-speaker diversity motivated our speaker-adaptive post-processing. In accordance with the shared-task rules, we used only the officially released training data and no external resources.

4. System Description

4.1. Overview

Our system followed a multi-stage pipeline for Arabic speech dictation with automatic diacritization. First, each input utterance was transcribed offline using a Whisper-based ASR model (Shatnawi, 2024), and the resulting transcripts were cached. Second, we applied *ASR Consistency Filtering* to determine whether each ASR transcript should be used directly or replaced with a fallback representation during training. Third, we fine-tuned a pretrained character-level diacritization model under three runs: two with different filtering thresholds (τ=0.5 and τ=0.9) and one additional run at τ=0.5 with a different random seed to increase ensemble diversity. Finally, at inference time, we combined the three checkpoints with confidence-based ensemble decoding and applied speaker-adaptive post-processing with constrained decoding to produce the final diacritized output.

4.2. Base Model and ASR

We built on the Diac Transformer, a character-level Transformer encoder with 1,631,507 trainable parameters, $d_{\text{model}} = 256$, and two Transformer blocks. The model used two input branches: a text branch encoding undiacritized Arabic input and an ASR branch encoding a diacritized Whisper transcript. The two branches were fused through cross-attention inside the encoder. A classification head predicted one diacritic label per character, where label 0 denoted the absence of a diacritic. We initialized the model from the pretrained checkpoint `rufaelfekadu/diac-transformer-text-asr-tashkeela-clartts`[1] (Fekadu, 2024).

ASR transcripts were generated using `sashat/whisper-medium-ClassicalAr`[2] (Shatnawi, 2024), used as a pretrained component without further fine-tuning. Train, development, and test ASR outputs were cached and reused during training and inference. Spoken punctuation was removed from ASR transcripts before passing them to the ASR branch.

4.3. Preprocessing and ASR Consistency Filtering

During TSV construction, rows were skipped if the diacritized gold text contained no Arabic letters, contained no harakat, or became empty after sanitization, where sanitization replaces tabs and line breaks with spaces, collapses repeated spaces, and strips surrounding whitespace.

To reduce the effect of noisy ASR inputs, we introduced *ASR Consistency Filtering*. For each example i, we computed a similarity ratio between the space-removed undiacritized gold text and the space-removed diacritics-stripped ASR output. Let a_i denote the space-removed undiacritized gold text and b_i the space-removed diacritics-stripped ASR output. The consistency score was computed as

$$\text{ratio}_i = \text{SequenceMatcher}(a_i, b_i). \quad (1)$$

[1] https://huggingface.co/rufaelfekadu/diac-transformer-text-asr-tashkeela-clartts

[2] https://huggingface.co/sashat/whisper-medium-ClassicalAr

Given a threshold τ, the ASR-side input used during training, denoted by $\hat{x}_i$, was determined as follows:

$$\hat{x}_i = \begin{cases} r_i & \text{if } \mathrm{ratio}_i \geq \tau, \\ \bar{x}_i & \text{if } \mathrm{ratio}_i < \tau \text{ or } r_i \text{ is empty}, \end{cases} \quad (2)$$

where r_i is the punctuation-cleaned ASR transcript and $\bar{x}_i$ is the undiacritized gold text. Thus, unreliable ASR was replaced with the undiacritized gold text as a fallback input rather than removed. This uses the undiacritized gold text rather than diacritized gold, so it does not inject target diacritics into the ASR branch during training.

We trained two threshold settings, $\tau = 0.5$ (`tau05`) and $\tau = 0.9$ (`tau09`). In addition, we trained a third run at $\tau = 0.5$ with a different random seed (`seed123`) to increase ensemble diversity. All three checkpoints were combined in the final ensemble.

4.4. Fine-Tuning

We fine-tuned the model using Adam with a learning rate of 5×10^{-5} and weight decay of 10^{-4}. Training used cross-entropy loss with label smoothing of 0.1, batch size 8, gradient accumulation over 16 steps, gradient clipping with a maximum norm of 1.0, and a cosine annealing learning-rate schedule with $\eta_{\min} = 10^{-6}$. We trained for up to 500 epochs and applied early stopping with a patience of 70 based on validation loss. The hyperparameters were selected based on development-set validation loss and WER.

We trained three runs: `tau05` ($\tau = 0.5$, default seed), `tau09` ($\tau = 0.9$, default seed), and `seed123` ($\tau = 0.5$, seed 123). The additional seed-123 run was included to increase ensemble diversity. We did not evaluate multiple alternative random seeds. Notably, `tau05` and `seed123` used the same filtered TSVs and differed only in random initialization. For all runs, we saved only the checkpoint with the best validation loss and used it for inference.

4.5. Ensemble and Post-Processing

At inference time, we combined the three checkpoints (`tau05`, `tau09`, and `seed123`) using character-level confidence voting. For each character position i, we computed softmax probabilities for each model and selected the prediction from the model with the highest maximum probability at that position:

$$k^\star = \arg\max_k \max(\mathrm{probs}_k[i]), \quad \hat{y}_i = \arg\max\big(\mathrm{probs}_{k^\star}[i]\big). \quad (3)$$

This ensemble operated position-wise and did not average probabilities across models.

θ	DER	WER	#SUKUN
0.55	10.04	32.63	15
0.60	9.77	31.26	13
0.65	**9.76**	**31.18**	**9**
0.70	9.75	31.19	12
0.75	9.99	31.72	7
0.80	10.16	32.02	4

Table 1: Grid search for the speaker-adaptive threshold θ on the development set (WCE, Incl. 0). Lower is better for DER and WER.

We then applied speaker-adaptive post-processing. Using development-set predictions, we counted word-final sukun versus vowel realizations for each speaker and classified speakers as `SUKUN`, `VOWEL`, or `MIXED` using a threshold θ. Intuitively, `SUKUN` speakers tend to suppress word-final case endings, `VOWEL` speakers tend to preserve them, and `MIXED` speakers show no strong preference. During post-processing, word-final tanwin and short case vowels were replaced with sukun only for `SUKUN` speakers; predictions for `VOWEL` and `MIXED` speakers were left unchanged. Speakers unseen in the development set were treated as `MIXED`.

We selected θ by grid search over $\{0.55, 0.60, 0.65, 0.70, 0.75, 0.80\}$ on the development set using WER as the selection criterion, since WER requires the full word to be diacritized correctly. As shown in Table 1, the best development-set WER was obtained at $\theta = 0.65$, which was therefore used in the final submitted system.

Finally, we applied constrained decoding after speaker-adaptive post-processing: if the diacritics-stripped predicted word did not match the diacritics-stripped input word, the predicted word was replaced with the original undiacritized input token. This served as a safeguard against word-level mismatches.

5. Results and Analysis

We report Diacritic Error Rate (DER), Word Error Rate (WER), and Sentence Error Rate (SER). Following the shared-task evaluation, we treat WER as the primary metric because it requires the full word to be diacritized correctly.

5.1. Development Set Ablation

Table 2 shows the contribution of each system component on the development set, evaluated both with and without case endings. Fine-tuning on consistency-filtered data provided the largest improvement, reducing WER from 43.07 to 32.75 with `tau05`. The stricter filtering run `tau09` (τ=0.9) performed slightly worse than `tau05`, suggesting that

Setting	System	DER	WER	SER
WCE	Baseline	15.94	43.07	85.38
	tau05	10.28	32.75	85.00
	tau09	10.79	33.38	87.31
	seed123	10.34	32.45	87.21
	Ensemble	10.24	32.15	84.23
	+Post-proc	**9.72**	**31.23**	**83.46**
WOCE	Baseline	10.21	24.86	78.08
	tau05	6.37	18.01	70.00
	tau09	6.82	18.51	71.54
	seed123	6.32	17.51	70.60
	Ensemble	6.29	17.58	69.62
	+Post-proc	6.29	17.58	69.62

Table 2: Development-set ablation results under the with-case-ending (WCE) and without-case-ending (WOCE) settings. Ensemble combines tau05, tau09, and seed123. Post-processing uses $\theta = 0.65$. Lower is better.

aggressive fallback reduced the usefulness of ASR-side information. Ensembling the three checkpoints yielded a further but modest gain, improving WER from 32.75 to 32.15. Finally, speaker-adaptive post-processing with θ=0.65 and constrained decoding reduced WER by an additional 0.92 points, producing the best development result of **9.72 DER, 31.23 WER, and 83.46 SER**. Under the WOCE metric, post-processing yielded no additional gain, as the speaker-adaptive corrections were confined to case endings, which are excluded from WOCE evaluation.

5.2. Official Test Results

Table 3 compares our official submission against the organizers' fine-tuned Text+ASR baseline. Our system outperformed the baseline in three of the four official evaluation settings. Under Incl. 0, WCE, our system improved WER from 31.84 to 30.37 while also reducing DER and SER. Under Incl. 0, WOCE, we improved all three metrics, including DER from 7.89 to 6.76. Under Excl. 0, WCE, the gains were smaller but still consistent across all three metrics. The only setting where the baseline remained stronger was Excl. 0, WOCE, which is consistent with our design because the post-processing mainly targets word-final case-ending behavior.

5.3. Analysis

Effect of the consistency threshold. In our experiments, the milder setting τ=0.5 performed better than the stricter setting τ=0.9: tau05 achieved 10.28 DER and 32.75 WER on the development set, compared with 10.79 DER and 33.38 WER for tau09. This suggests that overly aggressive fallback weakens the benefit of the ASR branch by discarding useful speech-side information together with noisy transcriptions.

Setting	System	DER	WER	SER
Incl. 0, WCE	Baseline	9.91	31.84	82.93
	Ours	**8.23**	**30.37**	**80.79**
Incl. 0, WOCE	Baseline	7.89	20.99	67.07
	Ours	**6.76**	**20.84**	**64.02**
Excl. 0, WCE	Baseline	8.52	24.73	78.66
	Ours	**8.46**	**24.33**	**74.39**
Excl. 0, WOCE	Baseline	**4.82**	**10.89**	**50.61**
	Ours	5.87	13.43	55.18

Table 3: Official test-set results for our submission compared with the organizer's fine-tuned Text+ASR baseline. WCE = with case endings, WOCE = without case endings, and 0 denotes no-diacritic characters. Lower is better.

Effect of ensembling. The three-checkpoint confidence-voting ensemble improved development performance from 32.75 to 32.15 WER and from 10.28 to 10.24 DER. This suggests limited but useful complementarity among the checkpoints.

Effect of speaker-adaptive post-processing. Speaker-adaptive post-processing produced the final development gain, reducing WER from 32.15 to 31.23 and DER from 10.24 to 9.72. The largest benefit was observed in settings where case endings remained part of the evaluation, consistent with the design of the post-processing step. In practice, these gains were most visible in word-final positions, where over-predicted case vowels were corrected to sukun for speakers with sukun-dominant realizations.

Remaining error sources. The remaining errors appear to be concentrated in word-final diacritics. Our system outperformed the organizers' fine-tuned Text+ASR baseline in three of four test settings, but underperformed it in the Excl. 0, WOCE condition. This suggests that our strongest gains came from correcting case-ending behavior, while other diacritic errors remain more challenging.

6. Conclusion

We presented our submission to Task 2 of the KSAA-2026 Shared Task on Arabic Speech Dictation with Automatic Diacritization. Building on the Diac Transformer, our system combined ASR Consistency Filtering, confidence-based ensembling, speaker-adaptive post-processing, and constrained decoding. On the official test set, it achieved 8.23 DER and 30.37 WER under Incl. 0, WCE. Limitations include dependence on development-set speaker statistics, sensitivity to the filtering threshold, reduced transfer across ASR models or dialects, and the absence of explicit syntactic modeling.

7. Bibliographical References

Asma Al Wazrah, Waad Alshammari, Rawan Almatham, Raghad Al-Rasheed, Afrah Altamimi, Rufael Marew, Sawsan Alqahtani, Hanan Aldarmaki, Abdullah Alharbi, Abdulrahman Alshehri, Mohamed Assar, Amal Almazrua, and Abdulrahman AlOsaimy. 2026. Ksaa-2026 shared task on arabic speech dictation with automatic diacritization. In *Proceedings of the 7th Workshop on Open-Source Arabic Corpora and Processing Tools (OSACT7)*.

Hanan Aldarmaki and Ahmad Ghannam. 2023. Diacritic recognition performance in arabic asr. pages 361–365.

Badr AlKhamissi, Muhammad ElNokrashy, and Mohamed Gabr. 2020. Deep diacritization: Efficient hierarchical recurrence for improved Arabic diacritization. In *Proceedings of the Fifth Arabic Natural Language Processing Workshop*, pages 38–48, Barcelona, Spain (Online). Association for Computational Linguistics.

Salman Elgamal, Ossama Obeid, Mhd Kabbani, Go Inoue, and Nizar Habash. 2024. Arabic diacritics in the wild: Exploiting opportunities for improved diacritization. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14815–14829, Bangkok, Thailand. Association for Computational Linguistics.

Ali Fadel, Ibraheem Tuffaha, Bara' Al-Jawarneh, and Mahmoud Al-Ayyoub. 2019. 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.

Hamdy Mubarak, Ahmed Abdelali, Hassan Sajjad, Younes Samih, and Kareem Darwish. 2019. 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, Minneapolis, Minnesota. Association for Computational Linguistics.

Han Qin, Guimin Chen, Yuanhe Tian, and Yan Song. 2021. Improving Arabic diacritization with regularized decoding and adversarial training. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 534–542, Online. Association for Computational Linguistics.

Sara Shatnawi, Sawsan Alqahtani, and Hanan Aldarmaki. 2024. Automatic restoration of diacritics for speech data sets. In *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, Mexico City, Mexico. Association for Computational Linguistics.

Brian Thompson and Ali Alshehri. 2022. Improving Arabic diacritization by learning to diacritize and translate. In *Proceedings of the 19th International Conference on Spoken Language Translation (IWSLT 2022)*, pages 11–21, Dublin, Ireland (in-person and online). Association for Computational Linguistics.

8. Language Resource References

Fekadu, Rufael. 2024. *Diac Transformer: Text and ASR-based Arabic Diacritization Model*. Hugging Face model repository.

Shatnawi, Sara. 2024. *Whisper Medium Classical Arabic*. Hugging Face model repository.