Restoration of Arabic Diacritics

N-gram Language Models, Dynamic Programming, RNNs, and Transformers

Yasser Hifny

Faculty of Computers and Artificial Intelligence
Capital University

1. What Is Diacritic Restoration?

Task

Add diacritics to undiacritized Arabic text while preserving the original letters.

ذهب الطالب إلى المدرسة

⇓

ذَهَبَ الطَّالِبُ إِلَى الْمَدْرَسَةِ

- Diacritics can change pronunciation, meaning, and grammatical case.
- Choosing correctly requires morphological and syntactic context, and sometimes semantic context.

2. Examples of Ambiguity

Undiacritized text	First reading	Second reading
علم	عِلْمٌ	عَلَمٌ
كتب	كَتَبَ	كُتُبٌ
ملك	مِلْكٌ	مَلَكَ

Key Idea

We search for the most probable diacritized sequence given the whole sentence, not each word in isolation from its context.

2. Types of Diacritical Marks

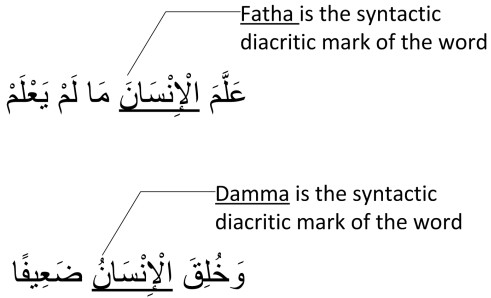


Figure: Same morphological diacritics but different case ending.

Morphological Tashkeel: Internal diacritics applied within a word to resolve semantic ambiguity, and (b) Syntactic Tashkeel: Terminal diacritics placed on the stem's final letter based on Arabic grammatical and syntactic rules.

3. Problem Formulation

Let the undiacritized input be $X = (x_1, \ldots, x_T)$, and let $Y = (y_1, \ldots, y_T)$ be a diacritization hypothesis.

$$\hat{Y} = \arg \max_{Y \in \mathcal{C}(X)} P(Y \mid X)$$

- $\mathcal{C}(X)$: all possible diacritizations consistent with the original letters.
- In the statistical model, each candidate diacritized word is represented as a state.
- We use a language model to score word sequences, then search for the best path using dynamic programming.

4. Decomposing the Joint Probability

Using the chain rule:

$$P(W) = \prod_{i=1}^{T} P(w_i \mid w_1, \ldots, w_{i-1})$$

And with a Markov approximation of order $n-1$:

$$\boxed{P(W) \approx \prod_{i=1}^{T} P(w_i \mid w_{i-n+1}^{i-1})}$$

- w_i: the candidate diacritized word at position i.
- We add sentence-start and sentence-end symbols when training the model and computing scores.
- We use logarithms to turn the product into a sum.

5. Arabic Example: Choosing a Word's Diacritics in Context

Undiacritized text: قرأ الطالب علمًا جديدًا

- Hypothesis A: قَرَأَ الطَّالِبُ عِلْمًا جَدِيدًا
- Hypothesis B: قَرَأَ الطَّالِبُ عَلَمًا جَدِيدًا

Language Model Scores

$$S(W) = \sum_{i=1}^{T} \log P(w_i \mid w_{i-n+1}^{i-1})$$

We choose the hypothesis with the highest score among the candidate paths.

The numerical values in the explanatory examples are illustrative and are not results measured on the dataset.

6. Unigram Model

The model assumes that words are independent:

$$P(W) \approx \prod_{i=1}^{T} P(w_i), \qquad P(w) = \frac{C(w)}{N}$$

- $C(w)$ is the number of times the diacritized word appears in the training data.
- N is the total number of words in the training data.
- The model ignores word order, so it is a weak baseline for disambiguation.

Arabic Example

A unigram model may prefer the common word عِلْمٌ over عَلَمٌ even when the context fits it less well.

7. Bigram Model

$$P(W) \approx P(w_1) \prod_{i=2}^{T} P(w_i \mid w_{i-1})$$

$$\boxed{P(w_i \mid w_{i-1}) = \frac{C(w_{i-1}, w_i)}{C(w_{i-1})}}$$

- It captures the compatibility of two adjacent words.
- Contextual example: الطَّالِبُ ذَهَبَ versus الطَّالِبَ ذَهَبَ.
- Unsmoothed probability estimates assign zero to unobserved bigrams.

8. Trigram Model

$$P(W) \approx P(w_1)P(w_2 \mid w_1)\prod_{i=3}^{T} P(w_i \mid w_{i-2}, w_{i-1})$$

$$P(w_i \mid w_{i-2}, w_{i-1}) = \frac{C(w_{i-2}, w_{i-1}, w_i)}{C(w_{i-2}, w_{i-1})}$$

- It uses the two previous words to estimate the current word.
- Longer context helps with grammatical case, but worsens data sparsity.
- It needs smoothing or interpolation with lower-order models.

9. Quadgram Model

$$P(W) \approx P(w_1)P(w_2 \mid w_1)P(w_3 \mid w_1, w_2) \prod_{i=4}^{T} P(w_i \mid w_{i-3}, w_{i-2}, w_{i-1})$$

$$P(w_i \mid w_{i-3}^{i-1}) = \frac{C(w_{i-3}, w_{i-2}, w_{i-1}, w_i)}{C(w_{i-3}, w_{i-2}, w_{i-1})}$$

- Stronger context if the four-gram appears in the training data.
- Greater sparsity: many four-word contexts appear only once or never.
- It should be evaluated with smoothing tuned on a validation set.

10. The Zero-Probability Problem and Smoothing

If a sequence does not appear in the training data, the direct count-based probability estimate may be zero.

$$P_{\text{Laplace}}(w_i \mid h) = \frac{C(h, w_i) + \alpha}{C(h) + \alpha|V|}$$

- h is the preceding context, $|V|$ is the vocabulary size, and $\alpha = 1$ in Laplace smoothing.
- For higher-order n-gram models, absolute discounting or Kneser–Ney may work better.
- A policy for out-of-vocabulary (OOV) words must be defined before evaluation.

11. Comparing Language Model Orders

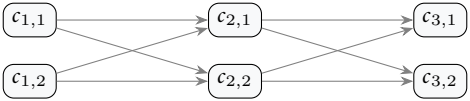
Model	Context	Advantage	Limitation
Unigram	None	Simple	Ignores order
Bigram	One word	Captures adjacency	Limited context
Trigram	Two words	Wider context	More sparsity
Quadgram	Three words	Longer context	Needs more data

Experimental Recommendation

Fix the data split, normalization, vocabulary, and smoothing policy, then compare all models on the same test set.

12. From Candidate Words to a Search Lattice

For each undiacritized word x_i, we generate a candidate set C_i of diacritized words whose letters match.



We need to find the best path without enumerating all possible combinations.

13. The Dynamic Programming Principle

We define $D_i(c)$ as the best score of a path ending at candidate c at position i.

$$D_i(c) = \max_{c' \in C_{i-1}} [D_{i-1}(c') + \log P(c \mid c')]$$

For a bigram model, with an additional score for the first word:

$$D_1(c) = \log P(c)$$

- We store the best predecessor in a backpointer array.
- After the last word, we pick the highest score and trace the pointers backward.
- The state can be generalized to store the last $n - 1$ words in higher-order n-gram models.

14. Worked Example: An Arabic Sentence

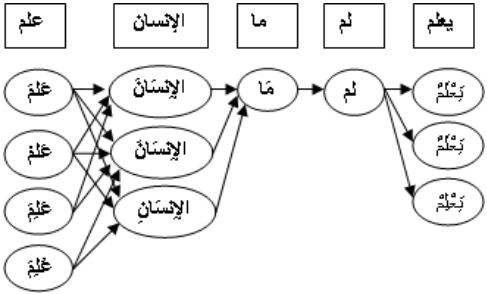


Figure: علم الإنسان ما لم يعلم

Path Update

$$D_i(c) = \max_{c'} \{D_{i-1}(c') + \log P(c \mid c')\}$$

Transition scores accumulate from left to right, and then we reconstruct the best path.

15. Why Do We Use the Logarithm?

Instead of multiplying very small probabilities:

$$P(W) = \prod_i P(w_i \mid h_i)$$

we compute the sum of logarithms:

$$\log P(W) = \sum_i \log P(w_i \mid h_i)$$

- It protects against numerical underflow.
- It does not change the ranking of paths, because the logarithm is an increasing function.
- All paths must use the same score units, including sentence boundaries.

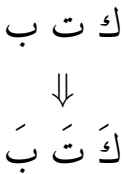
16. Search Complexity

If the average number of candidates per position is K and the sentence length is T:

- Brute-force enumeration: $O(K^T)$; dynamic programming: $O(TK^2)$ for search time, with $O(KT)$ lattice nodes.

17. From Word Models to Character Models

Diacritization can be treated as character-level sequence classification.



- Input: the Arabic characters after removing diacritics from the input.
- Target: the correct diacritic associated with each character, or a composite tag specifying the diacritic marks.
- The neural model captures context across the sentence without building an explicit candidate list for each word.

18. Classifying Each Character into 15 Tags

Class Name	Diacritic
Fatha	بَ
Kasra	بِ
Dhamma	بُ
Tanween Fath	بً
Tanween Kasr	بٍ
Tanween Dhamm	بٌ
Shadda	بّ
Shadda + Fatha	بَّ
Shadda + Kasra	بِّ
Shadda + Dhamma	بُّ
Shadda + Tanween Fath	بًّ
Shadda + Tanween Kasr	بٍّ
Shadda + Tanween Dhamm	بٌّ
Sukoon	بْ
No Tashkeel	<NT>

Table 1: Arabic Diacritics

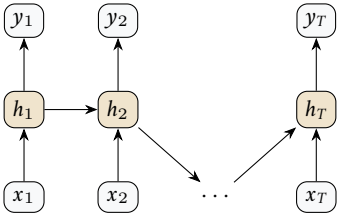
Figure: Tag set for nerual multiclass classification.

https://arxiv.org/pdf/2407.03236

19. RNN Architecture for Diacritization

$$h_t = \tanh(W_x x_t + W_h h_{t-1} + b_h)$$

$$o_t = W_o h_t + b_o, \qquad P(y_t \mid X) = \text{softmax}(o_t)$$



20. Training an RNN Model

We use the per-character cross-entropy loss:

$$\mathcal{L} = -\sum_{t=1}^{T} \log P(y_t^* \mid X)$$

- The loss is computed only over real character positions, not over padding.
- BiLSTM/GRU can be used to obtain left and right context.
- Evaluate the result after reassembling the diacritics onto the Arabic text.

21. Transformer Encoder for Diacritization

$$Q = HW^Q, \quad K = HW^K, \quad V = HW^V$$

$$\text{Attn}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right) V$$

- Self-attention relates each character to all characters within the context window.
- Each position's representation is passed to a softmax classifier over 15 diacritic classes.
- A bidirectional Transformer encoder can be used; the task here is sequence tagging, not autoregressive generation.

21. Transformer Encoder Figure

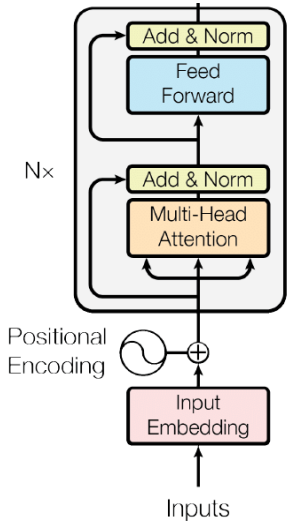


Figure: Transformer Encoder

21. Transformer Encoder for Diacritics Restoration

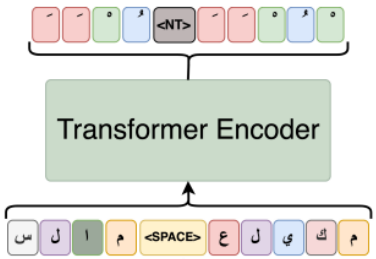


Figure: Transformer Encoder for diacritics restoration.

https://arxiv.org/pdf/2407.03236

22. RNN vs. Transformer Encoder

Property	RNN / BiLSTM	Transformer Encoder
Sequence processing	Sequential in time	Parallel within a layer
Context	Hidden state	Self-attention
Training	Backpropagation through time	Attention cost grows with context length
Long-range dependency	May vanish	Relatively direct access between positions
Data requirements	Can work with less data	Benefits from large data/pretraining

23. Model Output and Text Reconstruction

1. Remove the diacritics from the input while preserving character order.
2. The model predicts a diacritic tag for each applicable position.
3. The marks are attached to the characters according to the representation convention.
4. Punctuation and spaces are preserved, and the base letters must not be modified.

Example

Input: ذهب الطالب ⟶ Output: ذَهَبَ الطَّالِبُ

24. Evaluation Metrics: DER and WER

Diacritic Error Rate (DER) – character level

$$\text{DER} = \frac{\#\text{letters with an incorrect diacritic}}{\#\text{evaluated letters}} \times 100$$

Word Error Rate (WER) – word level

$$\text{WER} = \frac{\#\text{words with at least one incorrect diacritic}}{\#\text{evaluated words}} \times 100$$

- Diacritic Accuracy $= 100 - \text{DER}$ when both use the same positions.
- WER $\geq$ DER, because one wrong letter makes the whole word wrong.
- Base letters are never changed, so errors are almost all substitutions.

24b. With and Without Case Ending

The **case ending** is the diacritic on the last letter of a word. It depends on syntax and is the hardest to predict, so results are reported in two settings.

Setting	Evaluated positions	Evaluated words
With case ending (CE)	All letters	All words
Without case ending (no CE)	All letters except the last letter of each word	Last letter ignored when checking a word

Example

Reference: الطَّالِبُ Prediction: الطَّالِبَ

Only the case ending differs: this is an error **with** CE, and correct **without** CE.

Report both settings, and do not compare numbers across different protocols.

25. How Do We Compute Accuracy for N-gram Models?

1. Train unigram, bigram, trigram, and quadgram models on the diacritized text of the training split only.
2. Remove diacritics from the test inputs and generate diacritization candidates for each word.
3. Search for the best path for each order using context-appropriate DP.
4. Compare the predicted text with the diacritized reference text at the same positions.

Results Table to Fill In After Running

Model	Diacritic Accuracy (%)	DER (%)
Unigram	--	--
Bigram	--	--
Trigram	--	--
Quadgram	--	--

Reliable experimental values cannot be filled in without downloading the data and running the models on a fixed split; do not fabricate numbers.

26. The Tashkeela Dataset

Data Source

`https://huggingface.co/datasets/Misraj/Sadeed_Tashkeela`

- Inspect the data schema and fields before processing.
- Remove empty records and normalize the text while preserving the reference diacritics.
- Split by document or source to reduce leakage of repeated texts.
- Document the training, validation, and test sizes, and the number of characters and words.

27. Assignment: Statistical System

1. Build a parser that removes diacritics from the input without deleting letters or spaces.
2. Create a lexicon of candidate diacritized words for each undiacritized word.
3. Train unigram, bigram, trigram, and quadgram models with smoothing.
4. Implement Viterbi or DP with a state that keeps the context required by the model order.
5. Evaluate WER and DER on a fixed test set. You can use this script `https://github.com/AliOsm/arabic-text-diacritization/blob/master/helpers/diacritization_stat.py` to report the results.

28. Required Deliverables

- Complete code for data preparation, training, and evaluation.
- Build unigram/bigram/trigram/quadgram models + DP decoder.
- Training and validation curves for the Transformer encoder models.
- Report model size, training and inference time, and the accuracy of each model.

29. Conclusion

- Diacritic restoration is choosing the best sequence of diacritized words or tags.
- N-gram models estimate word-sequence probabilities; dynamic programming accumulates scores locally to find the best path.
- Neural and Transformer encoder models predict diacritics at the character level using sentence context.

Thank You

Questions and Discussion