

# Arabic Text Diacritization Project Report

Name	ID	Work Load
Omar Mohammed	9220550	Model Architecture
Amr Saad	9220560	Preprocessing
Amr Mahmoud	9220565	Feature Extraction
Kareem Ashraf	9220591	Feature Extraction

# Contents

<b>1</b>	<b>Preprocessing Pipeline</b>	<b>3</b>
<b>2</b>	<b>Model Architecture</b>	<b>3</b>
2.1	Architecture Components . . . . .	3
2.1.1	1. Input Layers (3 parallel inputs) . . . . .	3
2.1.2	2. Embedding Layers . . . . .	3
2.1.3	3. Feature Concatenation . . . . .	3
2.1.4	4. Bidirectional LSTM Layer . . . . .	3
2.1.5	5. Output Layer . . . . .	3
2.2	Model Summary Visualization . . . . .	4
<b>3</b>	<b>Features and Preprocessing</b>	<b>4</b>
3.1	Multi-Level Feature Extraction . . . . .	4
3.2	Vocabulary Management . . . . .	4
<b>4</b>	<b>Training Process</b>	<b>4</b>
4.1	Configuration . . . . .	4
<b>5</b>	<b>Evaluation Metrics</b>	<b>4</b>
5.1	Primary Metric: DER (Diacritic Error Rate) . . . . .	4
5.2	Position-Based Analysis . . . . .	5
<b>6</b>	<b>Results and Future Improvements</b>	<b>5</b>
6.1	Model Strengths . . . . .	5

# 1 Preprocessing Pipeline

1. **Text Cleaning:** Removes non-Arabic characters while preserving Arabic letters, diacritics, and whitespace.
2. **Sentence Tokenization:** Splits text into sentences using PyArabic library.
3. **Windowing:** Segments long sentences into fixed-size windows (1000 characters) for efficient processing.
4. **Character Extraction:** Separates base characters from their diacritical marks using Unicode normalization.
5. **Feature Engineering:** Extracts multiple feature types (character-level, word-level, positional).

# 2 Model Architecture

The system employs a sophisticated **Bidirectional LSTM (BiLSTM)** architecture with multi-feature input.

## 2.1 Architecture Components

### 2.1.1 1. Input Layers (3 parallel inputs)

- **Character Input:** Sequence of character IDs.
- **Word Input:** Sequence of word IDs (aligned with characters).
- **Position Input:** Position markers (0=not end, 1=end of word, 2=space).

### 2.1.2 2. Embedding Layers

- **Character Embedding:** 128-dimensional embeddings for characters.
- **Word Embedding:** 128-dimensional embeddings for words.
- **Position Embedding:** 16-dimensional embeddings for position markers.

### 2.1.3 3. Feature Concatenation

- Combines all embeddings:  $128 \text{ (char)} + 128 \text{ (word)} + 16 \text{ (position)} = 272 \text{ dimensions}$ .

### 2.1.4 4. Bidirectional LSTM Layer

- **Units:** 256 LSTM units (128 forward + 128 backward).
- **Total output dimension:** 512 ( $256 \times 2$  for bidirectional).

In addition to the BiLSTM configuration, we also experiment with a **Bidirectional RNN (BiRNN)** layer using the *same architecture and hyperparameters* to compare performance and evaluate the impact of recurrence type on diacritic restoration.

### 2.1.5 5. Output Layer

- **Dense Layer:** Maps to the number of diacritic classes.
- **Activation:** Softmax (multi-class classification).

## 2.2 Model Summary Visualization

Input: [Character IDs, Word IDs, Position IDs]  
|  
Embeddings: [128-dim, 128-dim, 16-dim]  
|  
Concatenation: 272-dim  
|  
BiLSTM/BiRNN: 512-dim (256x2)  
|  
BiLSTM/BiRNN: 512-dim (256x2)  
|  
Dense + ReLu  
|  
Dense + ReLu  
|  
Output + Softmax: Diacritic predictions per character

## 3 Features and Preprocessing

### 3.1 Multi-Level Feature Extraction

- **Character-Level:** Maps characters to unique IDs; handles <PAD> and UNK.
- **Word-Level:** Maps words to IDs based on frequency; provides contextual information.
- **Positional:** Encodes if a character is at the start, middle, or end of a word.

### 3.2 Vocabulary Management

- Character vocabulary: `utils/char2id.pickle`
- Word vocabulary: `utils/word2id.pickle`
- Diacritic mappings: `utils/diacritic2id.pickle`

## 4 Training Process

### 4.1 Configuration

- **Window Size:** 1000 characters
- **Batch Size:** 64
- **Epochs:** 10
- **Optimizer:** Adam

## 5 Evaluation Metrics

### 5.1 Primary Metric: DER (Diacritic Error Rate)

DER is the percentage of incorrectly predicted diacritics:

$$\text{DER} = \frac{\text{Number of Incorrect Diacritics}}{\text{Total Number of Diacritics}} \times 100\% \quad (1)$$

## 5.2 Position-Based Analysis

The evaluation includes position-aware DER calculation:

1. **DER for Non-Last Characters:** Error rate for characters within words.
2. **DER for Last Characters:** Error rate for word-final characters.
3. **Overall DER:** Combined error rate.

## 6 Results and Future Improvements

```
=====
DER Analysis by Character Position in Words
=====
DER for non-last characters: 2.90%
DER for last characters:    5.68%
Overall DER:                3.57%

Accuracy for non-last characters: 97.10%
Accuracy for last characters:    94.32%
Actual Accuracy: 96.43%
=====
```

Figure 1: Validation performance of the Bidirectional LSTM model, showing DER.

```
=====
DER Analysis by Character Position in Words
=====
DER for non-last characters: 3.17%
DER for last characters:    6.43%
Overall DER:                3.95%

Accuracy for non-last characters: 96.83%
Accuracy for last characters:    93.57%
Actual Accuracy: 96.05%
=====
```

Figure 2: Validation performance of the Bidirectional RNN model, showing DER.

### 6.1 Model Strengths

- Multi-Feature Approach improves accuracy significantly.
  - Bidirectional context captures forward and backward dependencies.
-