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

1 Preprocessing Pipeline **3**

2 Model Architecture **3**
- 2.1 Architecture Components . . . 3
  - 2.1.1 1. Input Layers . . . 3
  - 2.1.2 2. Embedding Layers . . . 3
  - 2.1.3 3. Feature Concatenation . . . 3
  - 2.1.4 4. Bidirectional LSTM Layers . . . 3
  - 2.1.5 5. Dense Transformation Layers . . . 4
  - 2.1.6 6. Output Layer . . . 4
- 2.2 Model Summary Representation . . . 4

3 Features and Preprocessing **4**
- 3.1 Multi-Level Feature Extraction . . . 4
- 3.2 Vocabulary Management . . . 4

4 Training Process **5**
- 4.1 Configuration . . . 5

5 Evaluation Metrics **5**
- 5.1 Primary Metric: DER (Diacritic Error Rate) . . . 5
- 5.2 Position-Based Analysis . . . 5

6 Results and Future Improvements **5**
- 6.1 Model Strengths . . . 6

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

The architecture receives three synchronized input sequences:

- **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

Each input stream is transformed into a dense vector representation:

- **Character Embedding**: 128-dimensional trainable embeddings.
- **Word Embedding**: 128-dimensional trainable embeddings.
- **Position Embedding**: 16-dimensional embeddings for the three position categories (inside word, end of word, space).

2.1.3 3. Feature Concatenation

The embedding outputs are concatenated to produce a unified feature vector for each time step:

$$128_{\text{char}} + 128_{\text{word}} + 16_{\text{position}} = 272 \text{ dimensions.}$$

2.1.4 4. Bidirectional LSTM Layers

The concatenated sequence is processed by two stacked BiLSTM layers:

- **BiLSTM 1**: 256 units (128 forward + 128 backward), returning the full sequence.
- **BiLSTM 2**: Another 256-unit bidirectional layer applied on top of the first.

Each BiLSTM produces:

$$256 \times 2 = 512\text{-dimensional output per time step.}$$

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. Dense Transformation Layers

Two fully connected layers refine the high-level temporal features:

- **Dense Layer 1**: 256 units, ReLU activation.
- **Dense Layer 2**: 256 units, ReLU activation.

2.1.6 6. Output Layer

- **Final Dense Layer**: Projects features to the number of diacritic classes.
- **Activation**: Softmax applied per time step for character-level classification.

2.2 Model Summary Representation

```
Inputs: Character IDs, Word IDs, Position IDs
        |
Embeddings: 128-char, 128-word, 16-position
        |
Concatenation -> 272-dimensional vector
        |
BiLSTM/BiRNN Layer (256x2) -> 512-dim
        |
BiLSTM/BiRNN Layer (256x2) -> 512-dim
        |
Dense (256, ReLU)
        |
Dense (256, ReLU)
        |
Softmax Output -> Diacritic class 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:

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

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%
Acutual 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%
Acutual 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.
