Cairo University
Faculty of Engineering

Department of Computer
Engineering

Fall 2025

NLP

Project Report

Submitted to

Eng. Omar Galal

Submitted by

Name	ID
Omar Sayed Ibrahim	9220538
Ahmed Mostafa Attia	9220110
Ahmed El Sayed Mahmoud	9220020
Abdelrahman Mohammed	9220438

1. Methodology

1.1 Data Preprocessing

The preprocessing pipeline ensures clean, standardized input:

- *TextCleaner Class*:
    - Removes numbers, brackets, unwanted punctuation, English characters, emojis, and symbols using regular expressions.
    - Normalizes spaces and removes diacritics for input features while preserving them for labels.
- *TextPreprocessor Class*:
    - Cleans lines and saves versions with/without diacritics.
    - Tokenizes into sentences of maximum length (600 characters) using textwrap.
- *DatasetBuilder Class*:
    - Encodes characters to indices (vocabulary size: 44 characters).
    - Encodes diacritics to labels (16 classes, including combinations like (1617, 1614) for shadda + fatha).
    - Pads sequences and creates PyTorch DataLoaders with batching.

This results in tensor datasets where inputs are character sequences and targets are diacritic labels, excluding non-Arabic or punctuation characters from labeling.

1.2 Model Architecture

The core model is *CharBiLSTM*, a character-level BiLSTM:

- *Input*: Sequence of character indices (batch_size × max_length).
- *Embedding Layer*: Trainable embeddings (dimension: 300) convert indices to dense vectors. (Variant: One-hot encoding replaces this with a one-hot vector per character.)
- *BiLSTM Layer*: Bidirectional LSTM (hidden_dim: 256, num_layers: 5, dropout: 0.2) captures contextual dependencies from both directions.
- *Batch Normalization*: Applied to LSTM outputs for stability.
- *Fully Connected Layer*: Maps to output classes (16 diacritics).
- *Output*: Logits for each character position (batch_size × max_length × 16).

The model handles variable-length sequences via masking (ignoring padding index 15 in loss computation).

The 2nd Model is RNN,

Architecture Components:

- **Input Layer**: TF-IDF features (dimension: 100) serve as direct input—no embedding layer required
- **Single-layer unidirectional RNN with hidden size: 128**

- **Output Layer**:
    - A fully connected linear layer maps the RNN hidden state to label predictions:
    - Linear: 128 → 16
    - Output classes: 16 diacritic labels (matching your LABELS mapping)

1.3 Training Process

1.3.1 Training Process For BILSTM Model

- *Optimizer*: Adam (learning rate: 0.001).
- *Scheduler*: StepLR (step_size: 5, gamma: 0.1) for learning rate decay.
- *Loss Function*: Cross-Entropy Loss (masked to ignore padding).
- *Metrics*: Accuracy and Macro F1-score (accounting for class imbalance in diacritics).
- *Training Loop* (Trainer Class):
    - Epochs: 20.
    - Saves best model based on validation accuracy.
    - Checkpoints for resuming training

1.3.2 Training Process For RNN + TF-IDF

- *TF-IDF vectors transformed to PyTorch tensors.*
    - *Batch size: 32*
- *DataLoader used for shuffled mini-batches.*
- *Training performance tracked via:*
    - *batch loss*
    - *classification accuracy*

1.4 Prediction and Evaluation

- *Predictor Class*:
    - For single sentences: Cleans, tokenizes, predicts, and reconstructs text with diacritics.
    - For datasets: Generates predictions and saves to CSV (e.g., submission.csv).
- *Evaluation*: Computes accuracy on validation/test sets, masking non-predictable characters (e.g., punctuation).

1.5 Model Variants

- *BiLSTM with Trainable Embeddings*: Learns dense representations during training, capturing semantic similarities between characters.
- *BiLSTM with One-Hot Encoding*: Uses sparse one-hot vectors (dimension: vocabulary size) as input, bypassing trainable embeddings for a baseline comparison.

2. Experiments and Results

2.1 Training and Validation Metrics for BILSTM Models

The models were trained on the full dataset with the following final epoch results:

- *BiLSTM (Trainable Embeddings)*:
    - **Loss**: 0.01227
    - **Train Accuracy:** 98.86% (Last Char: 98.223%)
    - **Train DER:** 1.14% (1.777%)
    - **Train F1:** 0.964
    - **Validation Accuracy:** 98.29% (Last Char: 96.553%)
    - **Validation DER:** 1.71% (3.447%)
    - **Validation F1:** 0.934
- *BiLSTM (One-Hot Encoding)*:
    - **Loss:** 0.03117
    - **Train Accuracy**: 98.84% (Last Char: 90.571%)
    - **Train DER**: 1.16% (19.429%)
    - **Train F1:** 0.9569
    - **Validation Accuracy:** 98.18% (Last Char: 87.694%)
    - **Validation DER:** 1.82% (12.306%)
    - **Validation F1:** 0.9263

2.2 Training and Validation Metrics for RNN + TF-IDF Model

Best Model Metrics (Epoch 15):

- **Training Loss**: 0.3119
- **Training Accuracy**: 89.82%(DER: 10.18%)
- **Validation Accuracy**: 83.20%(DER: 16.80%)

We use the BILSTM with The trainable embeddings variant outperforms the one-hot encoding and the BI RNN Model in all metrics, likely due to better representation learning and reduced overfitting on sparse inputs, so this model was used at Kaggle's test set submission

3. Workload Distribution

Omar Sayed Ibrahim	DatasetBuilder, LSTM with Trainable embedding, base Trainer
Ahmed Mostafa Attia	RNN With TF-IDF, GUI, preserving Code in Trainer
Ahmed El Sayed Mahmoud	TextCleaner, TextPreprocessor
Abdelrahman Mohammed	Predictor, LSTM With One Hot Encoding