Journal of Interdisciplinary Qur'anic Studies

Journal of Interdisciplinary Qur'anic Studies Vol. 5, Issue 1, March 2026, 5-25

# *Enhancing Arabic Word Stemming Using the Char Stemmer Deep Learning Model*

Azal Alaswaad

*PhD Candidate, Department of Computer Engineering, Iran University of Science and Technology, Tehran, Iran.*

Behrouz Minaei-Bidgoli [1]

*Professor, Department of Computer Engineering, Iran University of Science and Technology, Tehran, Iran.*

*Article History: Received 3 December 2025; Accepted 17 February 2026*

ABSTRACT:

Original Paper

The stemming of Arabic words is a crucial step in several text processing tasks, including text mining, information retrieval, and natural language processing. Arabic stemmers face many challenges, mainly due to the complex nature of Arabic words and their different writing styles. To address these challenges, this paper presents a novel approach integrated into a Python module for improving conventional Arabic stemming. The proposed approach employs neural network architectures to achieve high accuracy in Arabic text stemming. We utilize a Bi-LSTM architecture to extract Arabic stems and construct sequence-to-sequence stemming models. The proposed model is named Char Stemmer. To evaluate this model, we use a benchmark dataset of Qur'anic words and classical Arabic texts. Extensive experiments were conducted to evaluate the model, including comparisons with several well-established Arabic stemmers, namely Khoja, Light-10, P-Stemmer, Tashaphyne, and Al-Khalil. The experimental results show that the proposed model achieves a stem accuracy of 93.88%, substantially outperforming conventional rule-based and light stemming methods. The results demonstrate the effectiveness of deep learning sequence models for Arabic stemming and highlight the potential of character-level models for capturing complex morphological structures. The proposed model is generalizable, scales well, and can serve as a domain-independent solution for a wide range of Arabic NLP applications. Furthermore, the model can be directly integrated into Arabic information systems for tasks such as automated text normalization, intelligent search, and knowledge extraction in applied computing environments.

1. Corresponding Author. Email Address: b_minaei@iust.ac.ir

http://dx.doi.org/10.37264/JIQS.V5I1.6

KEYWORDS: Qur'anic Arabic, Qur'anic Corpus, Computational Qur'anic Studies, Arabic Natural Language Processing, Seq2Seq Models, Stemming, Deep Learning Architectures.

# *1. Introduction*

Natural Language Processing (NLP) is a major branch of artificial intelligence that focuses on enabling computers to understand, process, and generate human language. Given that language is central to human communication, the creation of models able to use natural language has been a long-standing goal in computing (Kamath et al. 2019). Nowadays, NLP approaches are extensively used in various fields such as information retrieval, text categorization, machine translation, and intelligent decision support systems. One of the most basic techniques in NLP is stemming, which is used to find the stem of a word. Stemming reduces the size of an index, makes matching more efficient, and improves the performance of downstream NLP tasks by minimizing morphological variation. Hence, stemming is important in applications like information retrieval, text categorization, question answering, and text mining (Alshalabi et al. 2022).

Arabic is also a challenging language for NLP due to its complex morphology, rich and highly productive morphological system with large numbers of word forms, complex orthography, and a high degree of speech ambiguity. Arabic words are usually inflected into a stem derived from one of ten well-established triconsonantal or (rarely) quadrilateral roots, to which various affixes and, rarely, circumfixes may be added, encoding additional syntactic information, but not necessarily meaning. Also, Arabic orthography permits the existence of several readings for the same surface form and correspondingly gives rise to homographs and context-sensitive meanings. These features make stem extraction much more difficult and reduce the efficiency of standard stemming approaches.

Several Arabic stemming methods have been introduced in the literature, including rule-based heavy stemmers and light statistical ones (Galal et al. 2019; Kanan et al. 2022; Hamza et al. 2021). Though these methods have performed with different degrees of success, most of them are based on manually created linguistic rules and pre-defined lists of affixes that limit their adaptability and robustness in dealing with irregular cases and ambiguous word formation. This does not, however, apply to processing classical Arabic text, which exhibits more complex word morphology and orthographic variation than Modern Standard Arabic.

Recent developments in deep learning, especially in sequence modeling

architectures, have achieved impressive results across many NLP tasks. Character-level seq2seq models, in particular, demonstrate a strong capability in learning fine-grained morphological regularities without being explicitly fed with linguistic rules. These advances motivate the current research, which investigates deep learning for Arabic stemming and considers a data-driven character-level approach.

To formulate the research hypothesis for this paper, we pose the following question: Is it feasible to design a model that accurately extracts the stem of an Arabic word, and what method yields the best results? To address this research question, we hypothesize that deep learning can enhance the accuracy of word stemming. Specifically, we propose using a sequence-to-sequence (seq2seq) model. Seq2seq models, commonly used in tasks like machine translation, handle inputs and outputs of varying and unaligned lengths. The standard approach involves an encoder-decoder architecture with two main components: an encoder, comprising several recurrent neural networks (RNNs), to encode the input word sequence of characters, and a decoder, also comprising several RNNs, to generate the corresponding output sequence.

The contributions of this work can be summarized as follows:

- Character-level Seq2Seq model for Classical Arabic stemming: This paper presents a character-level seq2seq stemming model based on Bidirectional LSTM (BiLSTM) as an encoder and LSTM as a decoder. The model is specifically applied to classical Arabic texts, which are known for their rich and complex morphological structures.
- Application to Qur'anic and Classical Arabic Morphology: Unlike many previous studies that focus on Modern Standard Arabic, this work evaluates the model on classical Arabic data, including Qur'anic text, providing insights into handling traditional morphological patterns.
- Unified Preprocessing and Evaluation Framework: The proposed model applies a consistent normalization and preprocessing pipeline across all experiments, ensuring a fair comparison between the proposed model and baseline stemmers.
- Comparative Evaluation with Widely Used Arabic Stemmers: The model is evaluated against several well-known Arabic stemming approaches, including Khoja, Light-10, P-Stemmer, Tashaphyne, and Al-Khalil, using the same dataset and evaluation metrics.

# *2. Literature Review*

We organize Related Works into two sections to cover research and ideas related to our paper.

## *2.1. Using seq2seq to extract stemming*

Abuein et al. (2024) proposed a new deep learning (DL)-based Arabic sarcasm detection framework for Twitter, called ArSa-Tweets, comprising five different deep learning models. A new robust preprocessing phase was introduced to improve the performance of the proposed sarcasm detection model. Further, this work presented a new golden corpus called ArSa-data to support sarcasm analysis of Arabic Twitter, which was considered a publicly available sarcastic corpus for researchers in this domain. Baseline experiments were conducted by deploying five NLP models: LSTM, Multi-headed CNN-LSTM-GRU, BERT, AraBert-V01, and AraBert-V02. To validate the proposed Arabic sarcasm detection system, several preprocessing steps were conducted in the proposed ArSa-Tweets system to enhance performance.

A recent study introduced a lemmatization algorithm based on Recurrent Neural Network (RNN) models specifically designed for the Urdu language. The model was tested on two datasets: the Universal Dependencies Corpus of Urdu (UDU) and the Urdu Monolingual Corpus (UMC). Lemmatization was performed using RNN models, with semantic and syntactic embeddings generated through the Word2Vec model and edit trees. Various models were tested under specific hyper parameters, including Bidirectional Long Short-Term Memory (Bi-LSTM), Bidirectional Gated Recurrent Units (Bi-GRU), Attention-Free Encoder-Decoder (AFED), and Bidirectional Gated Recurrent Neural Network (Bi-GRNN). The experimental results demonstrated that the Attention-Free Encoder-Decoder model achieved accuracy, recall, precision, and F-score values of 0.96, 0.95, 0.95, and 0.95, respectively, surpassing the performance of existing models (Hafeez et al. 2023). Das et al. (2023) investigated the effectiveness of various machine learning methods, including deep learning and hybrid models, for text categorization in Bangla and English. The study focused on sentiment analysis of comments from DARAZ, a well-known Bengali e-commerce platform containing reviews in both Bangla and English. Seven machine learning and deep learning models, including Long Short-Term Memory (LSTM), Bidirectional LSTM (Bi-LSTM), Convolutional 1D (Conv1D), and a hybrid Conv1D-LSTM model, were implemented. To improve model accuracy, preprocessing techniques were applied to a modified text corpus.

Shahade et al. (2023) proposed a deep learning approach for multilingual opinion mining, utilizing a hybrid fine-tuned Smith algorithm combined with the Adam optimizer (HFS-AO). This approach supported three languages—English, Marathi, and Hindi—which were selected through a web scraping algorithm. The collected data were annotated using the Zero-shot instance-weighting technique and underwent comprehensive preprocessing, including noise removal, data cleaning and reduction, transformation, stop-word removal, tokenization, stemming, lemmatization, and Part-of-Speech (POS) tagging. Al-Khatib et al. (2023) presented an Arabic light stemming algorithm, called Tashaphyne 0.4. This algorithm extracted the most roots and stems from words in Arabic text. Thus, the algorithm acted as a stemmer, root extractor, and segmentation tool simultaneously. The model consisted of three phases: Preparation, Root Extraction, and Stem Extraction. Kanan et al. (2022) introduced an improved P-stemmer by integrating it with various classifiers. Due to a shortage of Arabic datasets, the authors created an Arabic dataset from diverse online news sources to evaluate performance.

Almanaseer et al. (2021) applied a Deep Belief Network (DBN) for Arabic text diacritical restoration. DBN, a deep learning algorithm consisting of multiple Restricted Boltzmann Machines (RBMs), demonstrated effective data modeling and feature extraction. The approach utilized greedy layer-wise training followed by fine-tuning to achieve competitive classification performance. Ezhilarasi and Maheswari (2021) designed a lemmatizer for analyzing 11th-century paleographic stone inscriptions. Their approach utilized a Bi-LSTM Sequential Model for POS Tag Classification, effectively categorizing and predicting tags for new script words through tag assignment and prediction. Nuţu (2021) proposed a deep learning sequence-to-sequence method to advance automatic Romanian lemmatization. This study compared 24 systems using various combinations of recurrent, attention, and convolutional layers, with input consisting of word-lemma pairs at word and trigram levels. Two scenarios were explored for trigrams: predicting the lemma for each word in the sequence or only for the middle word. The lemmatizers were evaluated on two Romanian datasets: the belletristic subset of the CoRoLa corpus and the Romanian Explicative Dictionary (DEX).

Bounhas et al. (2020) focused on creating a morpho-semantic knowledge graph from Arabic vocalized corpora. The authors employed a tool suite to analyze and disambiguate classical Arabic texts, incorporating short diacritics to reduce ambiguities. The morphological analysis combined MADAMIRA and the Ghwanmeh stemmer to extract multiple lexicons. Alnaied et al. (2020) introduced the Arabic Morphology Information

Retrieval (AMIR) system for extracting Arabic stems. AMIR applied a set of rules related to Arabic letter relationships to identify stems for text indexing in Arabic retrieval systems. The system was evaluated using the EveTAR (2016) dataset, covering various Arabic event detection types. Galal et al. (2019) introduced a model called GStem to gather similar words that share the same root based on the Arabic extra letters as a preprocessing layer for the CNN. To evaluate the model, the authors collected an Arabic news dataset and used it in the experiments. They tested the WE from scratch using the collected training samples, and did not rely on any pre-trained data for word2vec or for GStem. The experiments showed that using GStem as a preprocessing step increased the accuracy of the CNN model because the number of distinct words was reduced.

Zalmout and Habash (2017) employed Bidirectional LSTM (Bi-LSTM) models for morphological tagging and language modeling, utilizing the outcomes of these models to rank the analyses of a morphological analyzer. The study incorporated various sub-word and morphological features at different linguistic levels in the tagger, including both word-based and character-based embeddings, and compared this system to a state-of-the-art benchmark. The dataset used for this study was the Penn Arabic Treebank (PATB parts 1, 2, and 3). Abdelali et al. (2016) introduced a new stemmer called Fast and Accurate Arabic Word Segmentation (FARASA). FARASA employs Support Vector Machine (SVM)-based ranking to determine the optimal segmentation of words into prefixes, stems, and suffixes. Hammo (2007) introduced a morphological analysis of Qur'anic texts, showing the Arabic morphology for some surahs from the Qur'an. This study also focused on comparing and contrasting English and Arabic morphology to determine the points where they differ by conducting morphological analysis for both languages.

## *2.2. Existing Datasets*

Noor Dataset comprises over 260,000 unique words sourced from a variety of materials, including the Holy Qur'an and classical Arabic texts.

# *3. Methodology*

In the domain of computational linguistics, particularly within the context of Arabic natural language processing, the development of robust and efficient algorithms for text normalization and stemming is crucial. To address these challenges, the proposed method introduces a novel approach implemented as a Python module using PyTorch, a popular deep learning

library. The method leverages neural network architectures, specifically Long Short-Term Memory (LSTM) networks, to achieve high accuracy in stemming Arabic texts. The proposed model aims to improve preprocessing quality for downstream natural language processing tasks such as machine translation, sentiment analysis, and text classification.

## *3.1. Data Preparation and Preprocessing*

### *3.1.1. Dataset Description*

The dataset utilized in this study consists of Arabic word–stem pairs. It is derived from the Noor dataset, which includes classical Arabic texts and Qur'anic vocabulary, totaling approximately 260,000 words. This dataset provides a rich and diverse representation of Arabic linguistic patterns, particularly those found in classical Arabic, making it suitable for training and evaluating stemming models. Each instance in the dataset is represented as a pair of words stored in a text file, where each line contains a tab-separated pair: the original (unstemmed) word and its corresponding stem. This structure allows the model to learn a direct mapping between surface word forms and their normalized stem representations in a supervised learning setting.

Before proceeding, it is important to clarify the meaning of the term "stem" as used in this paper. In Arabic NLP, there is often confusion between root extraction and stemming. Root extractors aim to identify the root (typically consisting of three letters), for example, K-T-B from *maktabah*. Our task is different. We follow the light stemming approach, where the stem is a surface form that can stand as a word after removing affixes. For example, the stem of *al-maktabāt* is *maktabah*, not K-T-B.

For the experiments conducted in this work, a subset of the dataset was used, specifically the file QuranWords2.txt, which contains word–stem pairs extracted from Qur'anic texts. This subset was intentionally selected due to its morphological richness and complexity, providing a challenging benchmark for evaluating the effectiveness of the proposed model. The dataset was divided into 90% for training and 10% for testing. This split ensures that the model is evaluated on unseen data to assess its generalization capability. The dataset includes a wide range of morphological variations, including prefixes, suffixes, and inflectional patterns commonly found in classical Arabic.

It is important to note that the dataset focuses on classical Arabic; therefore, the model's performance may be domain-dependent, and further evaluation on other Arabic text domains is required to assess its

generalizability.

### *3.1.2. Normalization and Tokenization*

Before feeding the data into the model, it undergoes normalization. The Normalizer Builder class from the Piraye library is employed to perform several normalization tasks:

- Alphabet Normalization: Converts all Arabic characters into a consistent form, addressing the issue of character variability across different Arabic scripts.
- Digit Normalization: Standardizes Arabic digits to ensure numerical data is uniformly represented.
- Punctuation Handling: Removes or standardizes punctuation marks, which are generally not required for the stemming process.
- Diacritic Removal: Deletes diacritical marks that are extensively used in Arabic for disambiguation but are often omitted in everyday typing and text messaging and are therefore not crucial for stemming.
- Whitespace Management: Eliminates extra spaces to ensure consistent separation between words.

These normalization steps help reduce the variability in the dataset, which could otherwise lead to an unnecessarily large model size and overfitting due to the high dimensionality of the input data. The dataset was divided into two main sets: 90% for training and 10% for testing, ensuring a reliable evaluation of the model's performance.

### *3.1.3. Padding Mechanism*

After normalization and tokenization, each word is split into individual characters, and an end-of-sequence (EOS) token is appended. The pad sequence function takes a sequence of characters, a maximum sequence length, and a padding character.

## *3.2. Model Architecture*

The architecture of the model plays a critical role in determining its effectiveness and efficiency in handling the challenges of Arabic text stemming. The proposed model, named Char Stemmer, utilizes a neural network architecture that leverages the strengths of Long Short-Term Memory (LSTM) networks and embedding layers.

### *3.2.1. Embedding Layer*

The first component of the Char Stemmer model is the embedding layer. This layer transforms sparse representations of input characters (character indices) into dense, continuous vector representations with an embedding dimension of 512 in this implementation.

### *3.2.2. LSTM Decoder*

The output from the encoder is fed into an LSTM decoder, which aims to generate the stemmed sequence of characters. Unlike the encoder, the decoder is not bidirectional but uses the concatenated hidden states from the encoder to initialize its internal state. At each time step, the decoder predicts the next character in the stemmed sequence based on its current state and the encoder's output.

### *3.2.3. Output Layer and Loss Function*

The final component of the Char Stemmer model is the output layer. This layer maps the high-dimensional output of the decoder to the size of the vocabulary, accurately predicting the probability distribution over all possible characters at each position in the output sequence. The model then applies a SoftMax function to this output to predict the most probable next character in the stemmed word. For training the model, a Cross-Entropy loss function is employed. This loss function is appropriate for classification tasks such as character prediction, where it measures the difference between the predicted probability distribution and the ground-truth distribution.

The Char Stemmer model applies complex rules that assume the script and morphology of Arabic. Through the combination of embedding layers with bi-directional and uni-directional LSTMs, the model effectively captures both local character interactions and longer-range dependencies essential for accurate stemming. The selection of the loss function is also consistent with the goal of the model of effectively learning to predict sequences of characters. These architectural components make the proposed approach capable of addressing Arabic text stemming challenges and provide a strong framework for different NLP tasks related to Arabic texts.

The character-level processing begins with a dense embedding layer that transforms sparse character representations into 512-dimensional continuous vector spaces. This embedding mechanism captures semantic relationships between Arabic characters, providing a rich foundation for morphological analysis. Figure 1 displays the proposed model for extracting

the stems.

### *3.2.4. Optimization and Loss Calculation*

The model uses the Adam optimizer (torch.optim.Adam), a popular choice for training deep neural networks due to its adaptive learning rate capabilities. This optimizer helps the model converge faster and more effectively than standard gradient descent methods. The learning rate and other parameters of Adam are set to their default values, which provide a good starting point.

The loss function utilized is the Cross-Entropy Loss, which is suitable for classification tasks such as predicting the next character in the stemming sequence. During training, the loss is calculated between the model's predictions and the ground-truth labels for each batch. This loss value indicates how well the model is performing; a lower loss implies better model predictions.

## *3.3. Training Procedure*

### *3.3.1. Data Loading and Batching*

The prepared and preprocessed dataset is loaded using the Arabic Stemmer Dataset class. The dataset is split into training and test subsets, with 90% of the data used for training and the remaining 10% reserved for evaluation. For efficient training, the data is batched using PyTorch's DataLoader. A batch size of 32 is used.

### *3.3.2. Model Configuration*

Before training begins, the Char Stemmer model is instantiated with predefined hyperparameters, including the input size (the size of the character vocabulary), the hidden size (1024 for the LSTM layers), and the output size (equal to the input size, representing the character predictions).

### *3.3.3. Training Loop*

The model was trained for 20 epochs, which provided a balance between convergence and computational cost. An embedding dimension of 512 and a hidden layer size of 1024 were used to capture complex morphological patterns. A batch size of 32 was selected to ensure stable training. The dataset was split into 90% for training and 10% for testing to evaluate

generalization performance. A detailed summary of all hyperparameter settings is provided in Table 1.

*Table 1. Hyperparameter settings of the proposed model*

| Hyperparameter | Value |
|---|---|
| Embedding Dimension | 512 units |
| Hidden Layer Size | 1024 LSTM units |
| Batch Size | 32 sequences |
| Training Epochs | 20 (with early stopping) |
| Optimizer | Adam (default parameters) |
| Loss Function | Cross-entropy with label smoothing |
| Learning Strategy | Mixed-precision training |

The model was trained on an NVIDIA GPU for approximately 12 hours. For early stopping, a patience of 5 epochs was used based on the validation loss. Dropout of 0.3 was applied after each LSTM layer to prevent overfitting. Figure 1 and Figure 2 illustrate the proposed model for Arabic stemming.

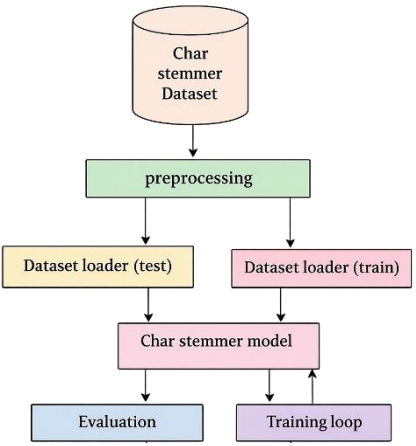


*Figure 1. The proposed model (Char Stemmer)*

The training procedure for the Char Stemmer model is designed to ensure robust learning without overfitting. By carefully managing data loading, model configuration, optimization, and iterative learning through epochs, the procedure lays a strong foundation for the model to learn the complex

mappings required for effective Arabic text stemming. This structured approach not only optimizes the training phase but also provides a foundation for comprehensive evaluation and deployment in practical NLP applications.

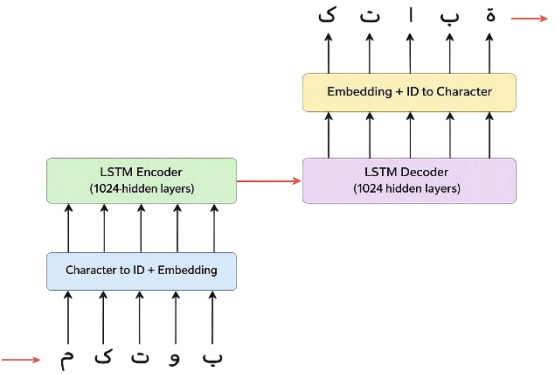


*Figure 2. The architecture of the Char Stemmer model*

# *4. Experiments and Results*

Assessing the performance of the proposed model is critical to understanding its effectiveness and identifying areas for improvement. This section discusses the metrics used to assess the model's performance and provides an analysis of the results obtained after training on a dataset composed of classical Arabic texts.

## *4.1. Evaluation Metrics*

The performance of the proposed model is evaluated using several key metrics:

- Accuracy: Measures the percentage of correct predictions over all predictions made.
- Precision, Recall, and F1-score: These metrics provide deeper insight into the model's performance, especially in handling class imbalances that are common in character-based tasks.

## *4.2. Results Analysis*

The final training epoch yielded a training loss of 0.2748 and an accuracy of 93.88%. The model was evaluated on unseen data, resulting in a test loss of 0.2889 and an accuracy of 93.88%. These results indicate that the model

effectively learned the stemming task with a high degree of accuracy and minimal overfitting. Table 2 summarizes these results.

*Table 2. The result of char stemmer model*

| Stemmer | Accuracy | Precision | Recall | F-measure |
|---|---|---|---|---|
| **Char Stemmer** | 0.93 | 0.82 | 0.70 | 0.75 |

However, the precision (82.01%), recall (70.46%), and F1-score (75.28%) on the test data suggest some challenges in the model's ability to generalize across all aspects of the data. While the precision is relatively high, indicating that the model's predictions are usually correct when it predicts a character, the lower recall and F1-score point to difficulties in consistently identifying all correct characters, especially in more ambiguous or rare stemming scenarios found in classical texts.

In our experiments, the model was trained for 20 epochs, which provided a balance between training time and convergence without overfitting. We used an embedding dimension of 512, allowing the model to capture rich character-level representations while keeping the computational cost manageable. The hidden layer size was set to 1024 units to enhance the model's capacity for learning complex morphological patterns in Arabic words. The batch size was set to 32, offering a compromise between stable gradient updates and training speed. The dataset was divided into 90% for training and 10% for testing, ensuring that the model was evaluated on unseen data to assess its generalization performance. Figure 3 illustrates the confusion matrix for the Char Stemmer model.

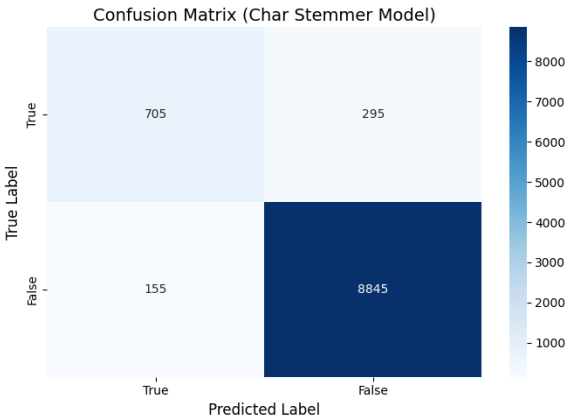


*Figure 3. Confusion matrix for the Char Stemmer model*

The performance evaluation of the Char Stemmer model demonstrates its effectiveness in processing and stemming classical Arabic texts with high accuracy. However, the detailed metrics also highlight areas for potential improvement, particularly in enhancing recall without sacrificing precision. Future work could explore more sophisticated model architectures, richer contextual embeddings, or advanced training techniques such as transfer learning from classical to modern texts to further refine the model's performance.

The Char Stemmer model, as detailed in the previous sections, presents a novel approach to stemming Arabic text, particularly focusing on the complexities of classical Arabic morphology. This discussion explores the reasons behind the model's effectiveness, its limitations, and the theoretical implications of the chosen methodology.

## *4.3. Error Analysis*

The Char Stemmer model attained an overall accuracy of 93.88%, with precision and recall of 82.01% and 70.46%, respectively. Despite its strong performance, errors primarily stem from the morphological richness of Qur'anic Arabic. Complex affixation patterns and irregular inflections challenge the model's ability to identify correct stems. Ambiguities arise when words share surface forms but differ in meaning or grammatical category, such as the word *malik*, which can be either a noun (king) or a verb (he has). Rare or infrequent words present additional difficulties due to their limited representation in the training data. Moreover, the absence of diacritics hampers the disambiguation of words with identical orthography but distinct pronunciations. Finally, the character-level approach limits contextual understanding, which is vital for accurate lemmatization in ambiguous cases.

## *4.4. Discussion*

- Robust Preprocessing: The model's success begins with comprehensive preprocessing steps that normalize and prepare the classical Arabic texts for machine learning.
- Character-Level Embedding: By transforming each character into a dense vector representation, the model captures subtle semantic and syntactic nuances of the Arabic language.
- Bidirectional LSTM Encoder: The utilization of a bidirectional LSTM allows the model to incorporate context from both the beginning and the

end of a word when making predictions about its stem.

- Sequential Decoding: The LSTM decoder predicts one character at a time, using the previous character's context to influence the next.
- Focused Loss Function: Employing Cross-Entropy Loss ensures that the training process focuses on improving the probabilities of correctly predicted characters.

## *4.5. Limitations and Areas for Improvement*

While the Char Stemmer achieves high accuracy, its recall and F1-score indicate room for improvement, especially in handling rare and ambiguous cases inherent in classical texts. These limitations suggest a few potential areas for enhancement:

- Expanded Training Data: Including a more diverse set of training examples, especially those that represent edge cases, could help improve the model's ability to generalize across less frequent morphological patterns.
- Hyperparameter Optimization: Further tuning of parameters such as the embedding size, the depth of LSTM layers, or the learning rate could yield improvements in model accuracy and recall.

## *4.6. Comparative Performance Evaluation*

We selected a set of well-known stemmers for comparison. We used these stemmers because they achieved high accuracy in previous studies. The stemmers used in the comparison are as follows:

- **Khoja Stemmer:** The Khoja stemmer removes the longest suffixes and prefixes. It then matches the remaining part with verb and noun patterns to derive the stem. It uses several linguistic data files, including lists of characters, punctuation marks, diacritics, and stop words. Khoja has two dictionaries, one for stems and another for roots. The Khoja stemmer is available as a Java implementation (Khoja & Garside 1999).
- **Light-10:** Light-10 performs the following steps for stemming:
  - It divides the text into words.
  - Normalization converts all (آ, إ, أ) to (ا), (ى) to (ي), and (ة) to (ه).
  - It removes the letter (و) when used as the conjunction "and".
  - It removes prefixes and suffixes from words. The prefixes used by

Light-10 include (ال, وال, بال, كال, فال, و, لل), while the suffixes include (ي, ة, ه, يـة, يـه, ين, ون, ات, ان, هـا). The Light-10 implementation is available in Java (Larkey et al. 2007).

- **Tashaphyne:** The Tashaphyne stemmer normalizes words and removes diacritics and elongation from input words. It then segments and stems the input using an affix lookup list for different levels of stemming. The Tashaphyne implementation is available in Python (Al-Khatib et al. 2023).
- **NLTK (Natural Language Toolkit):** NLTK is a collection of open-source programs for NLP. It includes over 50 corpora and lexical resources for tasks such as stemming, tokenization, classification, and parsing. NLTK was developed by Steven Bird and Edward Loper (Elbes et al. 2019).
- **P-Stemmer:** P-Stemmer, created by Kanan and Fox, is an updated version of Larkey's light stemmer. It removes only the prefixes rather than removing both prefixes and suffixes, resulting in effective and reliable stemming performance. The P-Stemmer implementation is available in Python (Kanan et al. 2019).

## *4.7. Comparative Performance Evaluation*

Following the evaluation of the proposed model, we compared it with five Arabic stemmers using the same Noor dataset. Table 3 presents a comprehensive comparison across multiple evaluation metrics.

*Table 3. Comparative performance of Arabic stemmers*

| Stemmer | Accuracy | Precision | Recall | F1-Score |
|---|---|---|---|---|
| **Char Stemmer** | 0.93 | 0.82 | 0.70 | 0.75 |
| **Al-Khalil** | 0.49 | 0.52 | 0.58 | 0.55 |
| **Tashaphyne** | 0.45 | 0.48 | 0.63 | 0.52 |
| **Light-10** | 0.37 | 0.41 | 0.57 | 0.45 |
| **Khoja** | 0.24 | 0.48 | 0.67 | 0.56 |
| **P-Stemmer** | 0.68 | 0.71 | 0.83 | 0.77 |

### *4.7.1. The Results of the Comparison*

- Superior Accuracy: The proposed model achieves significantly higher accuracy (93.88%) compared to other Arabic stemmers.
- Precision-Recall Trade-off: While the proposed model achieves the highest precision (82.01%), it shows a lower recall (70.46%) compared to the P-Stemmer (83%). This behavior reflects a precision-oriented strategy, which is preferable in downstream NLP tasks where over-stemming can introduce semantic errors.
- F1-Score: The proposed model achieves an F1-score of 75.28%, ranking second after P-Stemmer (77%), demonstrating balanced performance across precision and recall. Figure 4 illustrates the comparative performance of the proposed model and the other Arabic stemmers.

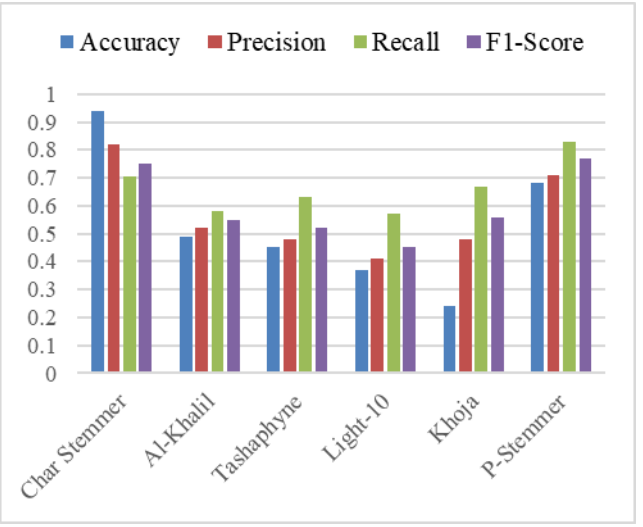


*Figure 4. Comparison of the evaluated stemmers*

### *4.7.2. Ablation Study*

We also conducted a brief ablation study to evaluate the contribution of each component of the proposed model. We trained two simpler versions: one with a regular LSTM encoder instead of a Bi-LSTM (without backward context), and another without dropout. The results are shown in Table 4.

The Bi-LSTM version achieved 93.88% accuracy. When the encoder was replaced with a regular LSTM, accuracy dropped to approximately 91.2%. This result suggests that capturing contextual information from both

prefix and suffix directions improves stemming performance. Without dropout, the model still performed well on the training data but showed signs of overfitting, as reflected by a larger gap between training and test losses. Therefore, the final model incorporates dropout.

*Table 4. Results of the ablation study*

| Model Configuration | Accuracy |
|---|---|
| Full model (Bi-LSTM + Dropout) | 93.88% |
| Without Bi-LSTM (LSTM only) | 91.20% |
| Without Dropout | 92.50% |

## *4.8. Qualitative Analysis of Stemming Results*

To further evaluate the performance of the proposed Char Stemmer model, a qualitative analysis was conducted by examining sample outputs and comparing them with selected baseline stemmers. Table 5 presents several examples of Arabic words along with their corresponding ground-truth stems, the outputs of the proposed model, and the outputs of representative baseline stemmers.

*Table 5. Qualitative comparison of stemming results*

| Word | True Stem | Char Stemmer | Light 10 | P Stemmer | Al-Khalil | Tashaphyne | Khoja |
|---|---|---|---|---|---|---|---|
| **المكتبات** | مكتبة | مكتبة | مكتب | مكتبة | كتاب | كتب | كتب |
| **يستخرجون** | يستخرج | يستخرج | يستخرج | خرج | يستخرج | خرج | خرج |
| **والكاتبين** | كاتب | كاتب | كتب | كتب | كاتب | كتب | كتب |
| **المستغفرين** | مستغفر | مستغفر | غفر | مستغفر | مستغفر | غفر | غفر |
| **بالمدارس** | مدارس | مدارس | مدرسة | مدارس | مدارس | مدرسة | درس |

# *5. Conclusion*

In conclusion, the proposed model represents a significant step forward in the computational processing of Arabic texts, combining advanced machine learning techniques with a deep understanding of linguistic principles. While there are areas for improvement, the current implementation provides a strong foundation for further exploration and development in Arabic NLP, particularly for classical and morphologically

complex texts.

The Char Stemmer model introduced in this paper represents a significant advancement in the field of Arabic natural language processing, specifically addressing the challenges associated with stemming classical Arabic texts. The model leverages a sophisticated deep learning architecture, encompassing robust preprocessing methods, character-level embedding, bidirectional LSTM networks, and a sequential decoding process. These components collectively contribute to the model's ability to accurately and effectively predict the stems of Arabic words, demonstrating high overall accuracy in both training and testing phases.

This paper has successfully demonstrated that deep learning techniques, particularly those based on LSTM networks, are well suited for handling the complexities of Arabic morphology and script variations. The bidirectional approach of the LSTM encoder captures contextual nuances from both directions of the input sequences, enhancing the model's sensitivity to the intricate patterns that define Arabic stemming. Furthermore, the embedding layer plays a crucial role by transforming discrete character representations into rich, continuous vector spaces, enabling the model to learn subtle linguistic distinctions.

### *Acknowledgements*

This paper is derived from the Ph.D. dissertation of Azal Alaswaad, conducted under the supervision of Professor Behrouz Minaei-Bidgoli in the Department of Computer Engineering, Iran University of Science and Technology. The first author would like to express her sincere gratitude to Professor Behrouz Minaei-Bidgoli for his invaluable guidance, continuous encouragement, and insightful comments throughout the doctoral research. The authors also gratefully acknowledge the support of the Vice Chancellor for Research and Technology Affairs of Iran University of Science and Technology for supporting this research.

### *Declarations*

Funding: No funding was received for conducting this study.
Conflict of Interest: The authors declare no competing interests.

## *References*

Abdelali, A., Darwish, K., Durrani, N., & Mubarak, H. (2016). Farasa: A fast and furious segmenter for Arabic. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations* (11–16). San Diego, California. Association for Computational Linguistics. https://doi.org/10.18653/v1/N16-3003

Abuein, Q., Ra'ed, M., Migdady, A., Jawarneh, M. S., & Al-Khateeb, A. (2024).

ArSa-Tweets: A novel Arabic sarcasm detection system based on deep learning model. *Heliyon, 10*(17), e36892. https://doi.org/10.1016/j.heliyon.2024.e36892

Al-Khatib, R. M., Zerrouki, T., Abu Shquier, M. M., & Balla, A. (2023). Tashaphyne0.4: a new Arabic light stemmer based on rhyzome modeling approach. *Information Retrieval, 26*(1-2). https://doi.org/10.1007/s10791-023-09429-y

Almanaseer, W., Alshraideh, M., & Alkadi, O. (2021). A Deep Belief Network Classification Approach for Automatic Diacritization of Arabic Text. *Applied Sciences*, *11*(11), 5228. https://doi.org/10.3390/app11115228

Alnaied, A., Elbendak, M. & Bulbul, A. (2020). An intelligent use of stemmer and morphology analysis for Arabic information retrieval. *Egyptian Informatics Journal, 21*(4), 209–217. https://doi.org/10.1016/j.eij.2020.02.004

Alshalabi, H., Tiun, S., Omar, N., Anaam, E. A., & Saif, Y. (2022). BPR algorithm: New broken plural rules for an Arabic stemmer. *Egyptian Informatics Journal*, 23(3), 363–371. https://doi.org/10.1016/j.eij.2022.02.006

Bounhas, I., Soudani, N. & Slimani, Y. (2020). Building a morpho-semantic knowledge graph for Arabic information retrieval. *Information Processing & Management, 57*(6), 102124. https://doi.org/10.1016/j.ipm.2019.102124

Das, R. K., Islam, M., Hasan, M. M., Razia, S., Hassan, M., & Khushbu S. A. (2023) Sentiment analysis in multilingual context: Comparative analysis of machine learning and hybrid deep learning models. *Heliyon*, 9(9), e20281. https://doi.org/10.1016/j.heliyon.2023.e20281

Elbes, M., Aldajah, A. & Sadaqa, O. (2019). P-stemmer or NLTK stemmer for arabic text classification? In: *2019 Sixth International Conference on Social Networks Analysis, Management and Security (SNAMS)* (516–520). IEEE. Granada, Spain. https://doi.org/10.1109/SNAMS.2019.8931818

Ezhilarasi, S., & Maheswari, P. U. (2021). Depicting a Neural Model for Lemmatization and POS Tagging of words from Palaeographic stone inscriptions. In: *2021 5th International Conference on Intelligent Computing and Control Systems (ICICCS)* (1879–1884). IEEE. Madurai, India. https://doi.org/10.1109/ICICCS51141.2021.9432315

Galal, M., Madbouly, M. M., & El-Zoghby, A. (2019) Classifying Arabic text using deep learning. *Journal of Theoretical and Applied Information Technology*, 97(23), 3412–3422.

Hafeez, R., Anwar, M. W., Jamal, M. H., Fatima, T., Espinosa, J. C. M., López, L. A. D., Thompson, E. B., & Ashraf, I. (2023). Contextual Urdu Lemmatization Using Recurrent Neural Network Models. *Mathematics*, *11*(2), 435. https://doi.org/10.3390/math11020435

Hammo, B., Sleit, A., & El-Haj, M. (2007). Effectiveness of query expansion in searching the Holy Quran. In: *The Second International Conference on Arabic Language Processing*, (1–10). Rabat, Morocco.

https://eprints.lancs.ac.uk/id/eprint/71276

Hamza, M. A. M., Ahmed, T. M. and Hilal, A. M. M. (2021) Text mining: A survey of Arabic root extraction algorithms. *International Journal of Advanced and Applied Sciences*, *8*(1), 11–19. https://doi.org/10.21833/ijaas.2021.01.002

Kamath, U., Liu, J., & Whitaker, J. (2019) *Deep learning for NLP and speech recognition*. Springer.

Kanan, T., Sadaqa, O., Almhirat, A., & Kanan, E. (2019). Arabic light stemming: A comparative study between p-stemmer, khoja stemmer, and light10 stemmer. In: *2019 Sixth International Conference on Social Networks Analysis, Management and Security (SNAMS)* (511–515). IEEE. Granada, Spain. https://doi.org/10.1109/SNAMS.2019.8931842

Kanan, T., Hawashin, B., Alzubi S., Almaita, E., Alkhatib A., Abu Maria K., & Elbes, M. (2022). Improving Arabic text classification using p-stemmer. *Recent Advances in Computer Science and Communications*, *15*(3), 404–411. http://dx.doi.org/10.2174/2666255813999200904114023

Khoja, S. & Garside, R. (1999). *Stemming Arabic text*. Computing Department, Lancaster University.

Larkey, L. S., Ballesteros, L., & Connell, M.E. (2007). Light Stemming for Arabic Information Retrieval. In: Soudi, A., Bosch, A.V., Neumann, G. (eds) *Arabic Computational Morphology* (221–243). Springer, Dordrecht. https://doi.org/10.1007/978-1-4020-6046-5_12

Nuţu, M. (2021) Deep Learning Approach for Automatic Romanian Lemmatization. *Procedia Computer Science*, 192, 49–58. https://doi.org/10.1016/j.procs.2021.08.006

Shahade, A. K., Walse, K. H., Thakare, V. M., & Atique, M. (2023). Multi-lingual opinion mining for social media discourses: An approach using deep learning based hybrid fine-tuned Smith algorithm with Adam optimizer. *International Journal of Information Management Data Insights*, *3*(2), 100182. https://doi.org/10.1016/j.jjimei.2023.100182

Zalmout, N., & Habash, N. (2017). Don't throw those morphological analyzers away just yet: Neural morphological disambiguation for Arabic. In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (704–713). Copenhagen, Denmark. Association for Computational Linguistics (ACL). https://doi.org/10.18653/v1/D17-1073