Arabic text diacritization using deep neural networks and Transformer-based architecture

Zineb chgari

National school of applied science

zineb.chgari@etu.uae.ac.ma

ABSTRACT

This study investigates the use of deep learning architecture with a focus on character-leven neural networks for autonomous Arabic diacritical analysis. Four architectures have been implemented: a transformer encoder-decoder, a Bigru model, a baseline stacked Bilstm, and a CBHG model. DER and WER were used as evaluation metrics when we trained and evaluated the models on the Tashkeela corpus. According to the results, the CBHG model maintained faster inference times while slightly outperforming the transformer encoder-decoder in terms of diacritical accuracy.However, our results indicate that the Transformer model may perform even better with bigger datasets, more parameter adjustment, and a larger model capacity

INTRODUCTION

Arabic is currently the fourth most utilized language on the internet due to the fast expanding corpus of electronic texts in the language [1]. Nearly hundreds of millions of Arabs speak Arabic as their native language, and more than a dozen nations have made it their official language [2]. Strong technologies are becoming more and more necessary to process the growing number of distinct Arabic corpora for other crucial uses [3].

Nonetheless, the development of Arabic Natural Language Processing (NLP) is made more difficult by the distinctive linguistic characteristics of the Arabic language. Arabic has numerous distinctive characteristics, including a distinctive cursive and joined script, a huge number of significant, non-alphabetic diacritical characters, and a right-to-left writing orientation. Computerized models that aim to operate with Arabic texts face challenges because of these features [3].

However, the structural and morphological characteristics of Arabic such as its right-to-left script, cursive and context-sensitive
letter forms, and reliance on diacritical marks—pose significant challenges for Natural Language Processing (NLP) applications [3].

Accordingly, many automatic methods to restore diacritics on Arabic electronic texts have been proposed with varying accuracy rates. Some of these methods followed traditional rule-based approaches, while others have applied statistical methods. Nonetheless, the most recent and promising systems, nevertheless, are those employing Artificial Intelligence (AI) applications, including Deep Learning (DL) and Neural Networks (NN) [4]. It has been reported the best-performing AI systems depended on DL, such as speech recognition devices in smartphones or Google's automatic translator [5]. This study presents an overview of the current methods proposed to automatically restore diacritics in digital Arabic texts with a specific focus on DL. It compares their success rates in order to guide future research into the most successful methods of the automatic Arabic diacritization process.

This study investigates and evaluates recent deep learning and Transformer-based approaches for restoring diacritics in Arabic texts. By comparing their effectiveness, the research aims to highlight the most promising directions for advancing automatic Arabic diacritization.

1. Related work

Automatic diacritization of Arabic text is one of the most challenging tasks in Arabic natural language processing. The researchers have employed several approaches to address this problem, including rule- based, statistical, hybrid, and deep learning ones.

The rule-based technique requires a deep understanding of the language to build a set of rules to recover the diacritics. Many systems use this technique alongside other techniques, such as

statistical and deep learning techniques. This approach has been used by [6] to build a text-to speech system and [7] to build several NLP systems such as machine translation and named entity recognition. Many systems are purely based on statistical approaches, such as [8]–[9] . In an early work to address the diacritization problem, Gal [8] built a Hidden Markov Model (HMM) for Hebrew and Arabic languages. To train the Arabic language model, he used the Quran corpus and reported a 14% Word Error Rate (WER) with Case-Ending (CE). Nelken and Shieber [10] used weighted finite-state transducers. The system used a combination of three probabilistic language models: a word-based, a letter-based, and an orthographical language model. Their best model used a combination of 3 gram words, a clitics concatenation, and 4-gram letter models. The model got 23.61% WER, 12.79% Diacritic Error Rate (DER) with CE; and 7.33% WER, 6.35% DER without CE.

Deep Neural Networks (DNN) techniques have recently been used to solve this problem with significant improve ments over the previous techniques. Many deep learning models are simple sequential models consisting of Recurrent Neural Networks (RNNs) and fully-connected (FC) layers. Al Sallab et al. [11] designed a system based on DNN and Confused Sub-Classes Resolution (CSR). The system attained 12.7% WER and 3.8% DER with CE. Abandah et al. [12] proposed a deep learning model based on Long short-term memory (LSTM) layers. The system significantly improved the diacritization over the previous works, achieving 5.82% WER, 2.09% DER with CE; and 3.54% WER, 1.28% DER without CE. Another system based on deep learning is designed by [13]. They examine several deep learning models with different architectures and different numbers of layers. Their best model, a three-layer bidirectional LSTM model, achieved 8.14% WER and 5.08% 3 VOLUME4, 2016 This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI Mokhtar and Qamar: Effective Deep Learning Models for Automatic Diacritization of Arabic Text DER with CE. Fadel et al. [14] used two deep learning approaches: Feed-Forward Neural Network (FFNN) and RNN, with several techniques such as 100-hot encoding, embeddings, CRF, and Block-Normalized Gradient (BNG). Their best settings got 7.69% WER, 2.60% DER with CE; and 4.57% WER, 2.11% DER without CE. Mubarak et al. [15] presented a character-level sequence-to-sequence deep learning model. The model used a Neural Machine Translation (NMT) setup on overlapping windows of words. The system achieved state-of-the-art results of 4.49% WER and 1.21% DER with CE. The encoder-decoder model that we propose in this paper, resembles this model to a great extent, but our model adapted a TTS model, which requires significant modifications to fit the diacritization problem. Moreover, the encoder-decoder model uses the location sensitive attention [16], whereas Mubarak et al. used the content-based attention [17]. Al-, Thubaity et al. [18] built a model using bidirectional LSTM networks with CRF. The model achieved 4.92% WER and 1.34% DER in the Holy Quran corpus.

2.METHODOLOGY

In this section, we present our implementation of four character-level deep learning architectures for Arabic text diacritization. The first model is a baseline sequence model based on stacked bidirectional LSTM layers with an embedding layer, referred to as RNN**.** It serves as a reference to evaluate the performance of basic recurrent structures on Tashkeela corpora. The second model, named StackedBiGRU, replaces LSTM units with gated recurrent units (GRUs) to reduce computational complexity while preserving temporal dependencies. The third model integrates a CBHG (Convolutional Bank, Highway network, and Bi-GRU) module inspired by the Tacotron architecture, originally developed for speech synthesis. This model combines convolutional, recurrent, and highway layers to capture both local and long-range patterns in the sequence. Finally, we implement a full Transformer Encoder-Decoder model adapted from sequence-to-sequence architectures used in neural machine translation, where the encoder processes the undiacritized input and the decoder predicts the corresponding diacritics sequence. In the following subsections, we describe the corpus, preprocessing steps, training strategy, and the evaluation metrics used to assess the effectiveness of each model on the Arabic diacritization task.

A.DIACRITIZATION CORPORA

The training data is the most important component of any deep learning model. Regretfully, there isn't a standard corpus for diacritical recovery in the Arabic language. There are two accessible corpora: ATL (paid) and Tashkeela. We made advantage of the Tashkeela corpus, which comprises 97 traditional Arabic works with 75 million fully vocalize

B .CORPUS PREPROCESSING

The data preprocessing can be divided into various stages:

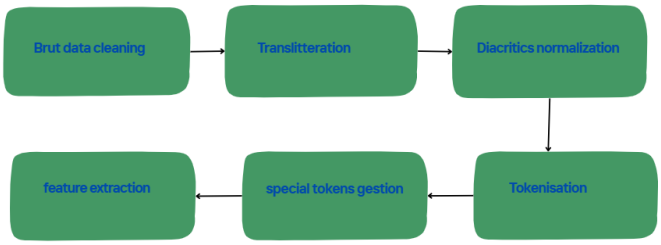


Fig. 1:Data preprocessing pipeline

Data Cleaning **:** All characters except the Arabic letters,diacritics, and punctuation are removed from the corpus,which is essential as it keeps only the characters thatcontribute to the diacritization to be learned by the models.

Transliteration: The cleaned Arabic text is transliterated into Buckwalter encoding**,** a system that maps Arabic letters to ASCII characters. For instance, ك becomes **k,** and ت becomes **t**. This transliteration facilitates character-level tokenization and standardizes input format for neural models.

Diacritic Normalisation **:**This step ensures consistent ordering of combined diacritics. For example, sequences like **a~** (fatha followed by shadda) are unified to **~a** (shadda comes first). Normalizing diacritic order prevents alignment errors during tokenization and improves the model's tagging .

Tokenization: The transliterated text is split into (character, diacritic**)** pairs. If a character has no diacritic, a special tag like **<NT>** (No Tashkeel) is assigned. This token-level annotation is essential for training sequence labeling models that predict the correct diacritic for each character.

Special Token gestion: Special tokens are added to define sequence boundaries and structure:

- **<BOS>** (Beginning of Sequence)
- <**EOS**> (End of Sequence)
- **<PAD>** (used to pad shorter sequences)
- **<MASK>** (optional, used in models like transformers)

These markers help the model understand sequence Limits and support batch training with variable lengt inputs.

Feature extraction : With the aid of retropropagation, the embeddings are automatically learned during model training. When a character or word is entered into the model, it is first transformed into a dense vector using an embedding couch that is initialised with arithmetic values. These vectors are then used to make predictions by recurrent couches (such as LSTM or GRU). The error between the model's predictions and the actual étiquettes is calculated during training and then spreads backwards over the network. This propagation updates all of the model's parameters, including the weight of the embedding couch.

C. BASELINE MODEL

The goal of this model is to assess the performance of a lightweight recurrent neural network architecture on Arabic text diacritization. The model starts with a 128- dimensional embedding layer that transforms each input token into a dense vector representation. This embedding layer is followed by a stack of one or more bidirectional LSTM layers, each with 128 hidden units per direction. These layers are designed to capture both past and future context in the sequence. A final fully connected layer with a softmax activation projects the output of the last LSTM layer to a probability distribution over the diacritics vocabulary (15 classes including the no-diacritic option). The number of trainable parameters depends on the number of LSTM layers used; for a single-layer configuration, the model contains approximately X million trainable parameters (to be computed based on actual vocabulary size and sequence length). This architecture serves as a baseline for comparison with more complex models such as CBHG or encoder-decoder frameworks.

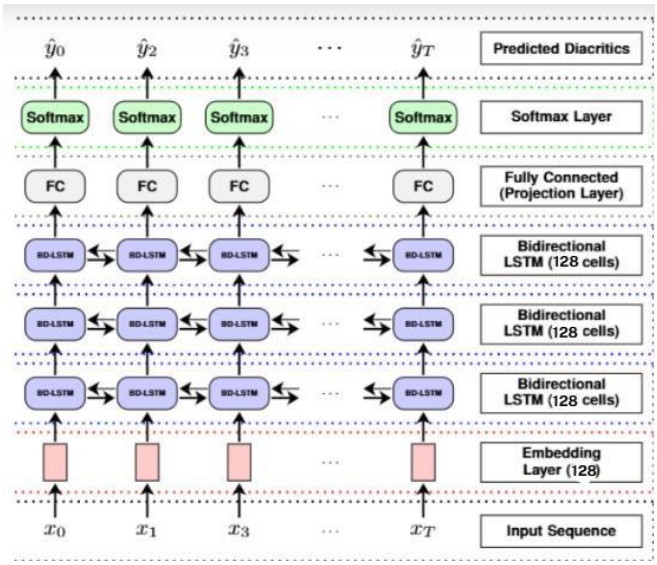


Fig. 2: The structure of the baseline model.

D.BIGRU MODEL

This architecture is designed to perform Arabic text diacritization using a compact and efficient recurrent model. It begins with an embedding layer of size 128 that transforms each input token into a dense representation. Then, the sequence passes through two stacked

bidirectional GRU layers, each with 128 hidden units in both directions, enabling the model to understand context from both sides of a word. The final output is generated by a fully connected layer that predicts the appropriate diacritic class for each token. The model outputs raw logits, making it compatible with training using loss functions like sparse categorical cross-entropy with logits. Its overall parameter count depends on the vocabulary size and sequence length, but remains relatively small, offering a good trade-off between performance and computational efficiency.

E.CBHG MODEL

The encoder-decoder architecture is designed for problems in which the length of input and output sequences are different. In the diacritization problem, however, the length of input and output sequences are the same. This property allows us to design a model based on only the encoder part of the encoder-decoder. The model has 14 Million trainable parameters, much more than the encoder-decoder model, but will be much faster since all the diacritics are predicted at once, unlike the encoder-decoder, where for each time step, the decoder predicts only one diacritic.We called this model the CBHG model since it used the CBHG module as its core architecture. We added a fully-connected projection layer and a softmax layer on top of the CBHG module to output the diacritics (Figure3).

The CBHG model works as follows: a 128- dimensions embedding layer first processes the input sequence. The embedding output is passed to two layers of non-linear transformation called pre-net: the first consists of 128 units with a RELU activation function and a dropout probability of 0.3, and the second consists of 128 units with the same activation and dropout probability. The pre-net output is then fed to the CBHG module, which outputs the input sequence's final representation. We added a fully- connected layer to project down the CBHG module's output to the number of possible diacritics. Lastly, we used a softmax layer to output the probability distribution for each diacritic.

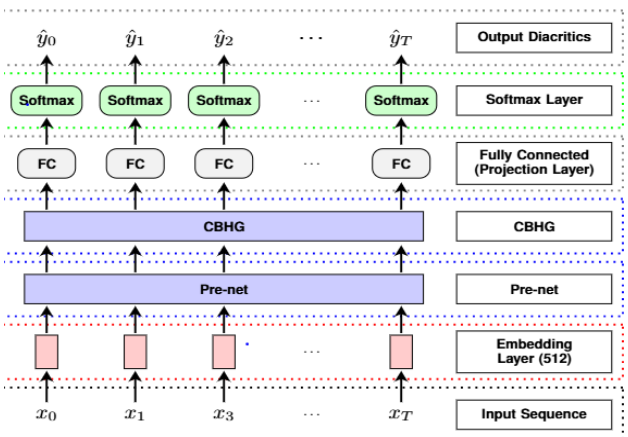


FIG 3: The CBHG model architecture. It is implemented using only the encoder of the encoder-decoder model with more robust parameters. We added a fully-connected layer and a softmax layer on top of the encoder to output the probability distribution for each character.

E.Transformer encoder-decoder

The Encoder-Decoder (ED) model is based on a three-seat architecture with internal dimensions of 512 and 16 heads of attention. It is initialized randomly without the need for pre-training. This model is based directly on diacritical data, such as Tashkeela, and formulates the task as a sequence-to-sequence translation, where each diacritical mark is pre-written by taking into account both the incompletely diacritical text and previously generated diacritical marks .

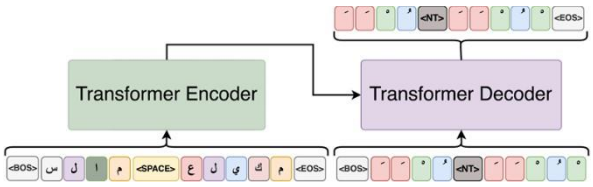


Figure4: Encoder-decoder Mode

F.DIACRITIZATION SYSTEMS EVALUATION

Diacritization Error Rate (DER) and Word Error Rate (WER) are the two most widely used metrics for assessing diacritization systems. It is possible to compute these metrics with or without Case-Ending (CE). Since the calculation without CE primarily relies on grammatical principles, the final character of each word is not included in the computation of errors. Error rates with CE indicate overall performance, whereas error rates without CE indicate performance on the core word. The error rates for both measurements without CE are nearly always lower (better) than the error rates with CE. Rate of Diacritization Error (DER): Given all characters and their correct corresponding diacritics, this metric calculates the percentage of characters that were not correctly diacritized. We calculate this metric by extracting diacritics from both the original and the predicted files and apply Eq.4

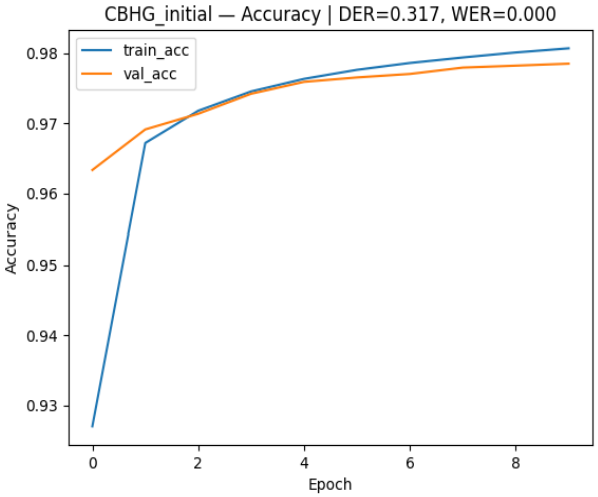
$$\mathrm{DER} = \frac{D_W}{D_W + D_C} \times 100 \qquad (4)$$

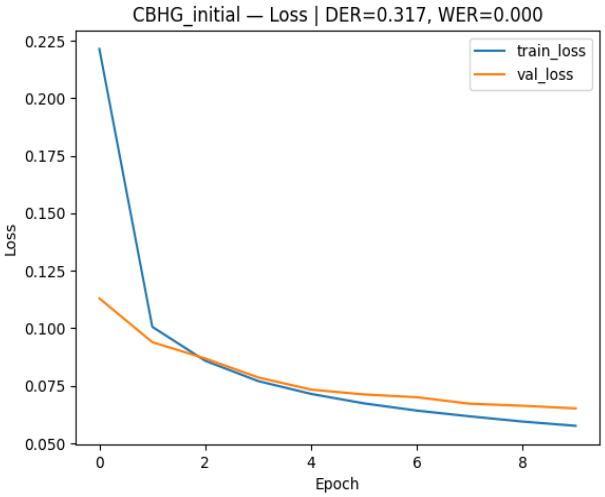
where DW is the number of wrong diacritics, and DC is number of correct diacritics. Eq. 4 calculates the errors of all characters, including spaces and punctuations. The characters that can be diacritized with more than one diacritic are treated as one diacritic. In the case of error rate without CE, the last

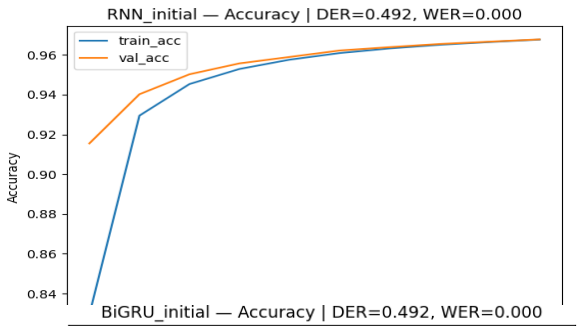
character of each word is excluded from the calculation. Word Error Rate (WER): This metric calculates
the percentage of words that contain at least one diacritization error. In this calculation, we compare all words from both the original and the predicted files to calculate the percent-age of unequal words
where W_W is the number of unequal words, and WC is the number of equal words. In the case of calculation without CE, the two compared words are equal even if both words'last diacritics are different.We trained the three models using the three corpora that we extracted from the Tashkeela corpus. We trained both the baseline and the CBHG model using tesla 4 GPU with 64 batch size. For the transformer encoder-decoder we used the same GPU with 32 batche size.

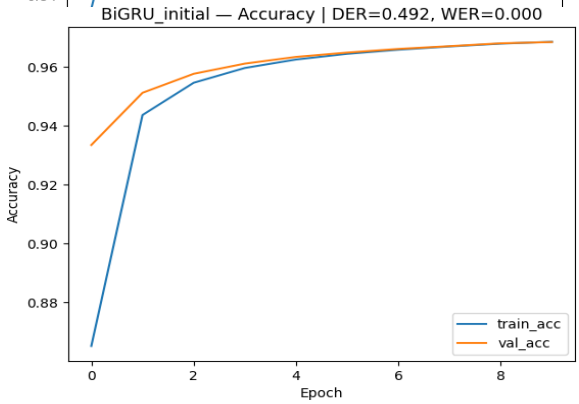
Model	DER	WER
BaselineModel	**0.4918**	**0.0002**
BIGRU	**0.4922**	**0.0002**
CBHG	**0.2915**	**0.0002**

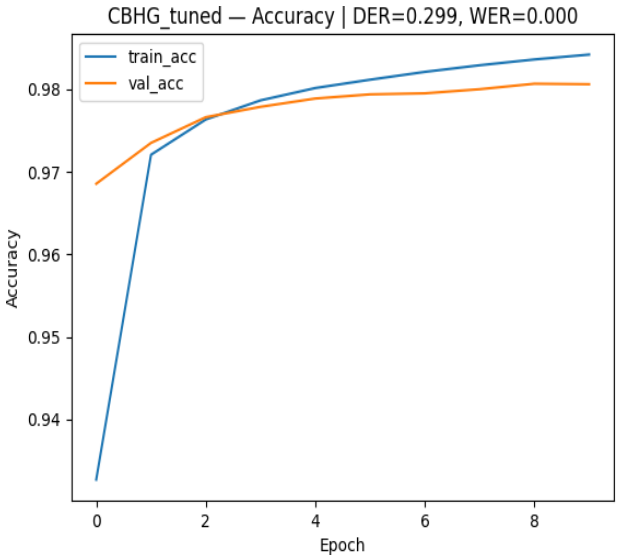
Since the CBHG model outperformed the other two models, we conducted a random search to identify the optimal combination of hyperparameters
The cbhg model's results improved after hyperparameter tuning, going from almost **0.29** to **0.26,** indicating a decrease in the diacritical prediction error.
Additionally, the Word Error Rate (WER), which was already very low at **0.00017**, remained nearly unchanged—suggesting that tuning mainly enhanced the diacritic-level accuracy without significantly impacting the word-level structure, which was already well preserved .











4. Interpretation

Based on the result, the cbhg somewhat outperformed the transormer encoder –decoder model in term of diacritization precision. The transformer still shows good results since of its capabilities of capturing long-tern dependecies through it's attention mechanism unlike the cbhg model which relieves on bigru and convolutionnal layers and that's why it's limited to a more constrained context window.

CONCLUSION

In this study, we explored several deep learning architectures, such as Bilstm, stacked Bigru, CBHG model, and transformer encoder-decoder, for the purpose of Arabic diacritization. Using the evaluation metrics der and wer, the tests were conducted on the Tashkeela corpus. Results from the experiment indicate that cbhg performed marginally better than the transformer encoder-decoder; however, the transformer may perform better if training data is scaled up, model capacity is increased, and more thorough hyperparameter searches are conducted.

Acknowledgment

My sincere gratitude goes out to our lecturer Pr.Mohammed cherradi for providing me with this opportunity to work on this project. Throughout this journey, your support, discipline, and trust have been invaluable, and the experience has significantly enhanced my comprehension of both Arabic NLP and applied deep learning. I appreciate your believing in my abilities and supporting me.

REFERENCES

[1] I. Guellil, H. Saâdane, F. Azouaou, B. Gueni, et D. Nouvel, « Arabic natural language processing: An overview », J. King Saud Univ. - Comput. Inf. Sci., vol. 33, no 5, p. 497-507, juin 2021, doi: 10.1016/j.jksuci.2019.02.006

[2] A. Fadel, I. Tuffaha, B. Al-Jawarneh, et M. Al-Ayyoub, « Arabic Text Diacritization Using Deep Neural Networks », in 2019 2nd International Conference on Computer Applications & Information Security (ICCAIS), Riyadh, Saudi Arabia: IEEE, mai 2019, p. 1-7. doi: 10.1109/CAIS.2019.8769512.

[3] A. Roberts, L. Al-Sulaiti, et E. Atwell, « aConCorde : Towards an open-source, extendable concordancer for Arabic », Corpora, vol. 1, no 1, p. 39-60, mai 2006, doi: 10.3366/cor.2006.1.1.39.

[4] « Full and Partial Diacritic Restoration: Development and Impact on Downstream Applications - ProQuest ». Disponible sur: https://www.proquest.com/openview/47e9740bdd31901021b8a0867c0bf0e2/1?cbl=51922&diss=y&pq-origsite=gscholar.

[5] « Explained: Neural networks », MIT News | Massachusetts Institute of Technology. Disponible sur: https://news.mit.edu/2017/explained-neural-networks-deep-learning-0414.

[6]. A. El-Imam, « Phonetization of Arabic: rules and algorithms », Comput. Speech Lang., vol. 18, no 4, p. 339-373, oct. 2004, doi:10.1016/S0885-2308 (03)00035-4.

[7] K. Shaalan, “Rule-based approach in arabic natural languag processing, ” International Journal on Information and Communication Technologies (IJICT), vol. 3, pp. 11–19, 06 2010.

[8] Y. Gal, “An HMM approach to vowel restoration in Arabic and Hebrew,” in Proceedings of the ACL-02 Workshop on Computational Approaches to Semitic Languages (Philadelphia, Pennsylvania, USA), Association for Computational Linguistics, July 2002.

[9] S. Ananthakrishnan, S. Bangalore, and S. S. Narayanan, “Automatic 12 diacritization of arabic transcripts for automatic speech recognition,” in Proceedings of the International Conference on Natural Language Processing (ICON), (Kanpur, India), pp. 47–54, December 2005.

[10] R. Nelken and S. M. Shieber, “Arabic diacritization using weighted finite state transducers,” in Proceedings of the ACL Workshop on Computational Approaches to Semitic Languages, Semitic ’05, (Stroudsburg, PA, USA), pp. 79–86, Association for Computational Linguistics, 2005.

[11] A.AlSallab, M. Rashwan, H. M.Raafat, and A. Rafea, “Automatic Arabic diacritics restoration based on deep nets,” in Proceedings of the EMNLP 2014 Workshop on Arabic Natural Language Processing (ANLP), (Doha, Qatar), pp. 65–72, Association for Computational Linguistics, Oct. 2014.

[12] G. A. Abandah, A. Graves, B. Al-Shagoor, A. Arabiyat, F. Jamour, and M. Al-Taee,

"Automatic diacritization of arabic text using recurrent neural networks," International Journal on Document Analysis and Recognition (IJDAR), vol. 18, pp. 183–197, 06 2015.

[13] Y. Belinkov and J. Glass, "Arabic diacritization with recurrent neural networks," in Proceedings of the 2015 Conference on Empirical Meth ods in Natural Language Processing, (Lisbon, Portugal), pp. 2281–2285, Association for Computational Linguistics, Sept. 2015.

[14] A. Fadel, I. Tuffaha, B. Al-Jawarneh, and M. Al-Ayyoub, "Neural Arabic text diacritization: State of the art results and a novel approach for machine translation," in Proceedings of the 6th Workshop on Asian Translation, (Hong Kong, China), pp. 215–225, Association for Computational Lin guistics, Nov. 2019.

[15] H. Mubarak, A. Abdelali, H. Sajjad, Y. Samih, and K. Darwish, "Highly effective Arabic diacritization using sequence to sequence modeling," in Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Tech nologies, Volume 1 (Long and Short Papers), (Minneapolis, Minnesota), pp. 2390–2395, Association for Computational Linguistics, June 2019.

[16] J. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio, "Attention-based models for speech recognition," in Proceedings of the 28th International Conference on Neural Information Processing Systems- Volume 1, NIPS'15, (Cambridge, MA, USA), p. 577–585, MIT Press, 2015.

[17] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," CoRR, vol. abs/1409.0473, 2014.

[18] A. Al-Thubaity, A. Alkhalifa, A. Almuhareb, and W. Alsanie, "Arabic diacritization using bidirectional long short-term memory neural networks with conditional random fields," IEEE Access, vol. 8, pp. 154984–154996, 2020.

[19] F. Alasmary, O. Zaafarani, et A. Ghannam, « CATT: Character-based Arabic Tashkeel Transformer », 14 juillet 2024, arXiv: arXiv:2407.03236. doi: 10.48550/arXiv.2407.03236.

[20] A. Fadel, I. Tuffaha, B. Al-Jawarneh, et M. Al- Ayyoub, « Neural Arabic Diacritization: State of the Art Results and a Novel Approach for Machine Translation », in Proceedings of the 6th Workshop on Asian Translation, 2019, p. 215-225. doi: 10.18653/v1/D19-5229.

zineb-chgari/Arabic-diacritization