LAMAD: A Linguistic Attentional Model for Arabic Text Diacritization

Raeed Al-Sabri and **Jianliang Gao***
School of Computer Science and Engineering, Central South University, China
alsabriraeed@tu.edu.ye, gaojianliang@csu.edu.cn

Abstract

In Arabic Language, diacritics are used to specify meanings as well as pronunciations. However, diacritics are often omitted from written texts, which increases the number of possible meanings and pronunciations. This leads to an ambiguous text and makes the computational process on undiacritized text more difficult. In this paper, we propose a Linguistic Attentional Model for Arabic text Diacritization (LAMAD). In LAMAD, a new linguistic feature representation is presented, which utilizes both word and character contextual features. Then, a linguistic attention mechanism is proposed to capture the important linguistic features. In addition, we explore the impact of the linguistic features extracted from the text on Arabic text diacritization (ATD) by introducing them to the linguistic attention mechanism. The extensive experimental results on three datasets with different sizes illustrate that LAMAD outperforms the existing state-of-the-art models.

1 Introduction

Arabic is one of the most widely spoken Semitic languages, the official language for about 27 countries, and spoken by more than 400 million speakers around the world (Ma et al., 2020). Diacritics are marks added above or below letters to give the word correct meaning and pronunciation(Alansary, 2018). However, More than 97% of Arabic texts ($e.g.$, magazines, newspapers, books,etc) are not diacritized (Neme and Paumier, 2020) increasing the text ambiguity which poses a challenge for diacritized texts based computational models (Abbad and Xiong, 2020; Hadjir et al., 2019). For example, translating the undiacritized Arabic sentences using Arabic machine translators face some difficulties. Figure 1 shows two Arabic sentences translation results without/with diacritization using Google

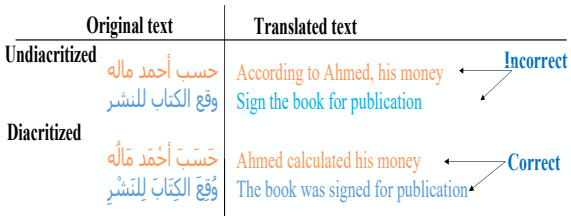

Figure 1: An example of Arabic translation without/with diacritization.

Translate[1]. It is noticeable that the undiacritized sentences are wrongly translated. For example, the undiacritized sentence حسب أحمد ماله is translated into "According to Ahmed, his money." while the correct meaning is "Ahmed calculated his money." The main reason for the failure of machine translation is that there are too many possible meanings for an undiacritized word, and the exact meaning can be revealed by the word context and diacritization, which is difficult for machine translation to figure out even with the presence of some Arabic contextual undiacritized words. On the other hand, the diacritized sentences are translated correctly due to the diacritics specify the word meanings. Arabic diacritizers can help reveal this ambiguity and improve the performance of various diacritized-text based NLP applications: automatic speech recognition (Abed et al., 2019), Arabic machine translation (Ameur et al., 2020), text to speech (Zine and Meziane, 2017), Part-of-Speech (POS) tagging (AbuZeina and Abdalbaset, 2019), and indexing diacritized text enable the search engine to exclude the unwanted matches.

Arabic diacritization problem has been addressed utilizing classical machine learning methods ($e.g.$, Hidden Markov models, the dynamic programming method Byte Pair Encoding (BPE), and Support Vector Machine) (Hifny, 2019; Pasha et al., 2014) or deep learning based approaches (Darwish

*Corresponding author

[1] https://translate.google.com/

Findings of the Association for Computational Linguistics: EMNLP 2021, pages 3757–3764
November 7–11, 2021.

et al., 2021; Fadel et al., 2019b; Al-Thubaity et al., 2020; Fadel et al., 2019a), which yield state-of-the-art performance for Arabic text diacritization using Bidirectional LSTM or convolutional neural networks. To improve the performance of ATD, deep learning models are enhanced by Arabic diacritization rules (Abbad and Xiong, 2020; Abandah et al., 2015).

Recent diacritization models adapting convolutional or recurrent neural networks show an improvement of Arabic text diacritization (ATD) (Mubarak et al., 2019; Alqahtani et al., 2020; Abbad and Xiong, 2020; Fadel et al., 2019b; Zalmout and Habash, 2017). However, most of those models are based on the character-level representation, which helps generalize the model but loses some useful linguistic features such as part of speech, word number, etc. To solve this problem, we propose a novel linguistic attentional model in which we introduce a linguistic feature representation at the character-level utilizing word and character linguistic features, and investigate their impact on ATD. Then, a linguistic attention mechanism for ATD is proposed to capture the most crucial features which influence the word diacritization. Our main contributions are summarized as the following:

- We propose a novel linguistic attentional model for ATD by introducing a new linguistic feature representation utilizing word and contextual character features and presenting a linguistic attention mechanism to focus on the effective features.
- We conduct extensive experiments on three benchmark datasets to explore the impact of the linguistic features on ATD, which show that linguistic features efficiently improve the diacritization performance and our proposed model outperforms the various state-of-the-art models.

2 The Proposed LAMAD

Figure 2 shows the architecture of the proposed linguistic attentional model. In our model, we introduce a new linguistic feature representation to extract the linguistic contextual features. Then, the linguistic attention mechanism for ATD is presented. The linguistic attention mechanism is adopted to distinguish the different importance of those features and capture the most crucial textual ones and have a decisive effect on diacritization, which is designed for the first time in Arabic diacritization and proved its effectiveness.

2.1 Initial Linguistic Feature

We extract contextual linguistic features which affect ATD according to the Arabic diacritization. In Arabic, diacritizing word is affected by linguistic features in text contexts such as part of speech, gender, named entity, word number, etc. Therefore, we utilize them in our model and present a new linguistic character-level representation to help the model improve the accuracy of the diacritization, which is the first time introduced for the diacritization problem (more details see Appendix A.3).

2.2 Linguistic Feature Embedding

In our model, the aim of the linguistic feature embedding is to convert the sequence of linguistic features into a low-dimensional vector sequence. The linguistic feature embedding layer receives the character features and produces a predefined vector representation for the features. Given a vector consisting of T linguistic features $C = \{f_1, f_2, ..., f_T\}$, every feature f_i is presented as a real-valued vector x_i. For each feature in C, the embedding matrix $C^{cf} \in \mathbb{R}^{d^c} |L|$ is looked up, where L is fixed-size character features, and d^c is the character-feature embedding size. The parameter that will be learned is the matrix C^{cf} and d^c is a hyper-parameter chosen by the user. The character linguistic feature f_i is converted into feature embedding x_i using the matrix-vector multiplication as:

$$x_i = C^{cf}\, l^i \tag{1}$$

where l^i is a vector of size $|L|$.

2.3 Linguistic Feature Learning

Considering the close relevance between two turns of characters, we utilize Bi-LSTM as an encoder to capture the features from both sides. The input into this layer is a set of embedded character-level linguistic features as a real-valued vector $embs = \{x_1, x_2, .., x_T\}$. LSTM composes of three main components: the forget gate f_t , which removes unnecessary information, input gate i_t , which adds information to cells, and the output gate, which filters and outputs necessary information. The current input x_i, the state generated by the previous step h_{i-1}, and the state of the current state of the cell c_{i-1}(peephole) are used to decide

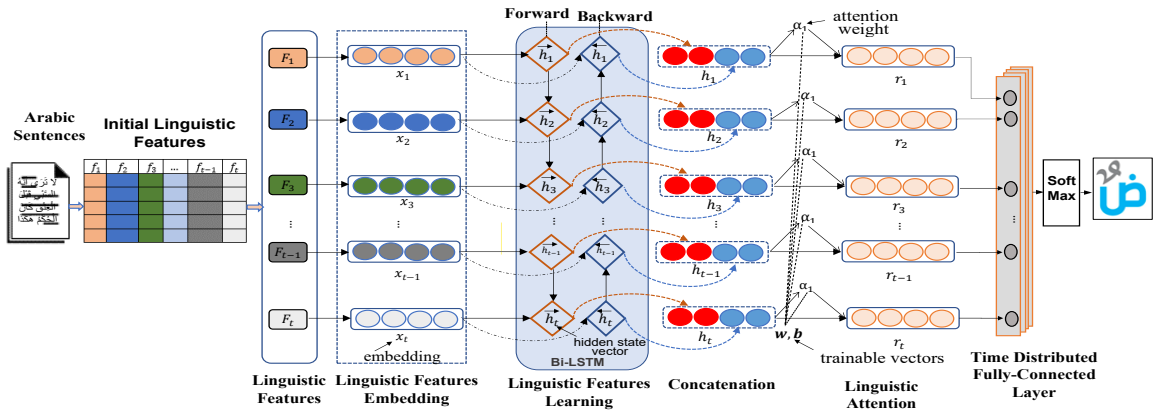

Figure 2: A Linguistic Attentional Model for ATD (LAMAD)

whether to take the inputs, forget the stored memory, and output the state that can be expressed by the following equations:

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i) \quad (2)$$

$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f) \quad (3)$$

$$g_t = tanh(W_{xc}x_t + W_{hc}h_{t-1} + W_{cc}c_{t-1} + b_c) \quad (4)$$

$$c_t = i_t\, g_t + f_t\, c_{t-1} \quad (5)$$

$$o_t = (W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_{t-1} + b_o) \quad (6)$$

$$h_t = o_t\, tanh(c_t) \quad (7)$$

where σ is the sigmoid function, and f, i, c, and o are forget, input memory cell activation and output vectors, respectively. The b bais and W vectors are learned while training. Each Bi-LSTM cell is composed of two LSTM cells. One LSTM cell processes input data from right to left and the other from left to right.

2.4 Linguistic Attention mechanism

The linguistic attention mechanism aims to capture the most effective linguistic features on ATD. Let H be a matrix consisting of output vectors $[h_1, h_2, ..., h_T]$ that Bi-LSTM layer produced, where T is the length of the character linguistic feature vector. The attention weight α_i is formed as follows:

$$m_i = \tanh(h_i), \quad (8)$$

$$\hat{\alpha}_i = w_i m_i + b_i, \quad (9)$$

$$\alpha_i = \frac{\exp(\hat{\alpha}_i)}{\sum_j exp(\hat{\alpha}_i)}, \quad (10)$$

where w and b are trained parameter vectors of the attention layer. The dimensional vector and its elements are the weights corresponding to each feature in the input character features. Therefore, the output representation r_i is given by:

$$r_i = \alpha_i h_i \quad (11)$$

3 Experiments and Results

3.1 Datasets

Three diacritized datasets, which cover various genres and different sizes, are used: Quran, The Holy Islamic book and most correct diacritized dataset (Hamed and Zesch, 2017), Tashkeela (Fadel et al., 2019a), assembled from articles, books, speeches, etc., and Sahih Al-Bukhary (Al-Thubaity et al., 2020), a collection of Islamic hadith. The datasets include small and large datasets with long and short sentences. Table 1 shows the datasets statistics.

3.2 Evaluation Metrics

Diacritization Error Rate (DER) and Word Error Rate (WER) are the two main evaluation metrics used to test the performance of Arabic diacritizer (Hamed and Zesch, 2017). DER is the proportion of characters that are labelled with incorrect diacritic. WER is the percentage of words in which at least one letter has been incorrectly diacritized.

3.3 Baselines

To evaluate the performance of our model (LAMAD), it is compared with five state-of-the-art models that use the character-level representations: RNN-BNG model (RNN-BNG) (Fadel et al., 2019b), BiLSTM-CRF (Al-Thubaity et al., 2020), and Multi-components system (Abbad and Xiong, 2020), A-TCN (Alqahtani et al., 2019), and BiLSTM (Náplava et al., 2018) which proved to be more effective than word-level representation models and can be generalized for new text context. We

Dataset	No. Lines	No. words	No. Char	Avg C/L	Avg C/W	Avg W/L
The Holy Quran	6236	78245	404846	64.92	5.17	12.55
Sahih Al-Bukhary	8877	516065	2505108	282.2	4.90	58.14
Tashkeela	55000	2654285	12183337	221.52	4.59	48.26

Table 1: Datasets Statistics.

Linguistic Features	DER	WER
CHAR-PRIOR-SEG -POS -STEM	**1.85**	**6.76**
CHAR-PRIOR-CE-NUM-FEM-STEM-NE	2.33	8.89
PRIOR-CE-NUM-FEM-STEM-NE	16.56	24.68
CHAR-PRIOR-CE-FEM-STEM	2.66	10.24
CHAR-PRIOR-SEG-CE-NUM-FEM-NE	2.29	8.49
CHAR-PRIOR-SEG-CE-NUM-FEM	2.40	9.05
CHAR-PRIOR-SEG-NUM-FEM-NE	2.29	8.44
CHAR	9.22	27.38

Table 2: Diacritization performance on Quran Corpus with linguistic feature representations.(CHAR= pre and post characters, PRIOR= diacritics prior, SEG= character position in word segment, CE= whether character is case ending letter, NUM= word number, FEM= word gender, STEM= word stem, NE= word named entity, POS= word part of speech).

Dataset/Model	Quran	Sahih Al-Bukhary	Tashkeela
(Hifny, 2019)	6.04	5.6	4.2
Hybrid system	6.3	5.90	5.10
Multi_components	12.37	6.35	9.89
BiLSTM-CRF	5.04	3.25	2.9
A-TCN	5.40	3.60	3.40
BiLSTM	4.92	3.31	3.0
RNN-BNG	5.36	3.34	2.60
LAMAD (Ours)	**1.85**	**1.71**	**2.13**

Table 3: DER comparison of our proposed model with the state-of-the-art models for each dataset.

also compare our model to a hybrid system that use morphological and syntactic diacritics rules and statistical treatments (a hybrid system) (Chennoufi and Mazroui, 2017) and Byte Pair Encoding (BPE) method with sub-word units dictionary to (Hifny, 2019), which are computationally cost and fail to be generalized for text in different context.

3.4 Preprocessing

Due to datasets are not in unified structures, preprocessing step is performed: the datasets are divided into lines, each line is a sentence, where "?", "!" and "." are used as separators. Then, we removed the extra diacritics such as Sukun from "ال" and duplicated diacritics and diacritics that appear on non-Arabic letters. We also unify the position of the compound diacritics such as the Shaddah should come first. The diacritics are unified to be after the letter if it appeared before the letter. Moreover, the lines that are non-diacritized or have less than 80% of the diacritized characters are removed.

3.5 Results and Comparisons

In this paper, we first investigate the performance of linguistic features on ATD using the proposed linguistic attentional model. We explore the impact of different linguistic representations to choose the most effective representation on Quran dataset, the most correct diacritized corpus and has short sentences which can test the robustness of the proposed model. Extensive experiments have been done to study the impact of linguistic features on ATD, but we only report the representations that achieved the lowest DER and WER in table 2. From the results, it is seen that utilizing the linguistic features significantly enhance the performance of ATD and the best results were scored using the CHAR-PRIOR-SEG-POS-STEM representation, which achieved about 1.85% DER on Quran Dataset, although it has few features. The prior feature is a binary vector indicating whether the character can accept the Arabic diacritic decided from the training set.

The comparison results in Table 3 and 4 show that our model achieves the best results compared to the baseline models. For example, it achieves about 3.19%, 3.51%, and 10.52% better DER than BiLSTM-CRF model, RNN-BNG model, and Multi_components System, respectively, on Quran dataset, and about 6.58%, 9.17%, and 17.93% better WER than BiLSTM-CRF model, RNN-BNG model, and Multi_components System, respectively, on Quran corpus. It is also noticed from the results on the Tashkeela dataset that our model outperforms the baseline models. For example, our model achieves about 0.47% and 7.76% better DER and about 0.38% and 18.36% better WER than RNN-BNG and Multi_components model.

Dataset/Model	Quran	Sahih Al-Bukhary	Tashkeela
(Hifny, 2019)	14.6	11.3	9.98
Hybrid system	15.10	12.3	10.5
Multi_components	24.69	14.15	25.67
BiLSTM-CRF	13.34	8.07	7.86
A-TCN	13.87	10.40	9.25
BiLSTM	12.62	9.21	8.53
RNN-BNG	15.93	9.58	7.69
LAMAD (Ours)	**6.76**	**5.61**	**7.31**

Table 4: WER comparison of our proposed model with the state-of-the-art models for each dataset.

Dataset	With case ending	Without case ending	With case ending	Without case ending
	Including no diacritic		Excluding no diacritic	
Quran	1.85	1.06	2.70	1.44
Sahih AlBukhary	1.71	1.19	2.52	1.67
Tashkeela	2.13	1.61	2.71	1.97

Table 5: DER performance with/without case endings with including and excluding no diacritic characters.

In our work, we also report the diacritization errors with/without case endings. In addition, the redefinition of DER and WER by (Fadel et al., 2019b) in which the irrelevant characters such as punctuations and numbers are excluded while counting the percentage of mislabeled characters, is also used. Tables 5 and 6 show the diacritization errors DER and WER with/without case endings and also with excluding and including no diacritic characters.

For more in-depth analysis, we randomly choose 100 words that have diacritization errors for each testing dataset to analyze the types of common errors. Table 7 displays the number of words with one, two, and three or more diacritization errors. The results show that the model almost behaves similarly for each dataset in terms of the number of diacritization errors per words such as most words (88 on average) have one diacritic error. These observations demonstrate the consistency of the performance of the model regardless of data type.

Dataset	With case ending	Without case ending	With case ending	Without case ending
	Including no diacritic		Excluding no diacritic	
Quran	6.76	2.93	6.56	2.79
Sahih AlBukhary	5.61	2.89	5.20	2.63
Tashkeela	7.31	3.69	6.9	3.49

Table 6: WER performance with/without case endings with including and excluding no diacritic characters.

Error samples	Number of diacritic errors		
	1	2	≥ 3
Quran	88	10	2
Sahih AlBukhary	87	9	4
Tashkeela	89	10	1

Table 7: Numbers of words with one, two, three or more diacritization errors in the error samples.

Error samples	Beginning	Middle	End
Quran	10	22	82
Sahih AlBukhary	16	28	73
Tashkeela	14	21	77

Table 8: Positions of diacritization errors in error samples.

Table 8 shows the diacritization errors that appeared at the beginning, middle, and end of the words. We observed that the model is sensitive to the syntactic roles since most of the errors appeared at the end of the words.

Table 9 shows the diacritization error distributions over three major Arabic POS categories: verbs, nouns, and particles. We observed that most of errors appeared on nouns. The main reason is that nouns in Arabic occur more frequently.

4 Conclusion

In this paper, we propose a linguistic attentional model to tackle the Arabic diacritization problem. A new features representation method is presented, and the impact of morphological and syntactic information, extracted as features, is investigated. To evaluate the proposed system, three diacritized Arabic corpora are used; two of them are small datasets and one large dataset with long sentences. The proposed LAMAD achieved the best results compared to the state-of-the-art models.

Acknowledgements

The work is supported by the National Natural Science Foundation of China (No. 61873288).

Error samples	Nouns	Verbs	Particles
Quran	59	36	5
Sahih AlBukhary	87	10	3
Tashkeela	82	16	2

Table 9: Diacritization errors distributions over the main Arabic POS categories in the error samples.

References

Gheith A. Abandah, Alex Graves, Balkees Al-Shagoor, Alaa Arabiyat, Fuad Jamour, and Majid Al-Taee. 2015. Automatic diacritization of Arabic text using recurrent neural networks. *International Journal on Document Analysis and Recognition*, 18(2):183–197.

Hamza Abbad and Shengwu Xiong. 2020. Multi-components System for Automatic Arabic Diacritization. *LNCS (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 12035 LNCS(April):341–355.

Sa'ed Abed, Mohammad Alshayeji, and Sari Sultan. 2019. Diacritics Effect on Arabic Speech Recognition. *Arabian Journal for Science and Engineering*, 44(11):9043–9056.

Dia AbuZeina and Taqieddin Mostafa Abdalbaset. 2019. Exploring the Performance of Tagging for the Classical and the Modern Standard Arabic. *Advances in Fuzzy Systems*, 2019.

Abdulmohsen Al-Thubaity, Atheer Alkhalifa, Abdulrahman Almuhareb, and Waleed Alsanie. 2020. Arabic diacritization using bidirectional long short-term memory neural networks with conditional random fields. *IEEE Access*, 8:154984–154996.

Abdulmohsen Al-Thubaity, Atheer Alkhalifa, Abdulrahman Almuhareb, and Waleed Alsanie. 2020. Arabic Diacritization Using Bidirectional Long Short-Term Memory Neural Networks with Conditional Random Fields. *IEEE Access*, 8:154984–154996.

Sameh Alansary. 2018. Alserag: An automatic diacritization system for Arabic. *Studies in Computational Intelligence*, 740:523–543.

Sawsan Alqahtani, Ajay Mishra, and Mona Diab. 2019. Efficient convolutional neural networks for diacritic restoration. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1442–1448, Hong Kong, China. Association for Computational Linguistics.

Sawsan Alqahtani, Ajay Mishra, and Mona Diab. 2020. A multitask learning approach for diacritic restoration. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8238–8247, Online. Association for Computational Linguistics.

Mohamed Seghir Hadj Ameur, Farid Meziane, and Ahmed Guessoum. 2020. Arabic Machine Translation: A survey of the latest trends and challenges. *Computer Science Review*, 38:100305.

Tim Buckwalter. 2002. Buckwalter arabic morphological analyzer version 1.0. *Linguistic Data Consortium, University of Pennsylvania*.

Amine Chennoufi and Azzeddine Mazroui. 2017. Morphological, syntactic and diacritics rules for automatic diacritization of arabic sentences. *J. King Saud Univ. Comput. Inf. Sci.*, 29(2):156–163.

Kareem Darwish, Ahmed Abdelali, Hamdy Mubarak, and Mohamed Eldesouki. 2021. Arabic diacritic recovery using a feature-rich bilstm model. *ACM Trans. Asian Low-Resour. Lang. Inf. Process.*, 20(2).

Kareem Darwish and Wei Gao. 2014. Simple effective microblog named entity recognition: Arabic as an example. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 2513–2517, Reykjavik, Iceland. European Language Resources Association (ELRA).

Kareem Darwish and Hamdy Mubarak. 2016. Farasa: A new fast and accurate Arabic word segmenter. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 1070–1074, Portorož, Slovenia. European Language Resources Association (ELRA).

Ali Fadel, Ibraheem Tuffaha, Mahmoud Al-Ayyoub, et al. 2019a. Arabic text diacritization using deep neural networks. In *2019 2nd international conference on computer applications & information security (ICCAIS)*, pages 1–7. IEEE.

Ali Fadel, Ibraheem Tuffaha, Bara' Al-Jawarneh, and Mahmoud Al-Ayyoub. 2019b. 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*, pages 215–225, Hong Kong, China. Association for Computational Linguistics.

Ismail Hadjir, Mohamed Abbache, and Fatma Zohra Belkredim. 2019. An approach for arabic diacritization. In *International Conference on Applications of Natural Language to Information Systems*, pages 337–344. Springer.

Osama Hamed and Torsten Zesch. 2017. A Survey and Comparative Study of Arabic Diacritization Tools. *J. Lang. Technol. Comput. Linguistics*, 32(1):27–47.

Yasser Hifny. 2019. Open vocabulary arabic diacritics restoration. *IEEE Signal Processing Letters*, 26(10):1421–1425.

Tinghuai Ma, Raeed Al-Sabri, Lejun Zhang, Bockarie Marah, and Najla Al-Nabhan. 2020. The Impact of Weighting Schemes and Stemming Process on Topic Modeling of Arabic Long and Short Texts. *ACM Trans. Asian Low-Resour. Lang. Inf. Process.*, 19(6).

Hamdy Mubarak, Ahmed Abdelali, Hassan Sajjad, Younes Samih, and Kareem Darwish. 2019. 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 Technologies, Volume 1 (Long and Short Pa-*

pers), pages 2390–2395, Minneapolis, Minnesota. Association for Computational Linguistics.

Jakub Náplava, Milan Straka, Pavel Straňák, and Jan Hajič. 2018. Diacritics restoration using neural networks. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Alexis Amid Neme and Sébastien Paumier. 2020. Restoring Arabic vowels through omission-tolerant dictionary lookup. *Language Resources and Evaluation*, 54(2):487–551.

Iroro Orife. 2018. Attentive Sequence-to-Sequence Learning for Diacritic Restoration of YorùBá Language Text. In *Interspeech 2018*, pages 2848–2852, ISCA. ISCA.

Arfath Pasha, Mohamed Al-Badrashiny, Mona Diab, Ahmed El Kholy, Ramy Eskander, Nizar Habash, Manoj Pooleery, Owen Rambow, and Ryan Roth. 2014. MADAMIRA: A fast, comprehensive tool for morphological analysis and disambiguation of Arabic. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 1094–1101, Reykjavik, Iceland. European Language Resources Association (ELRA).

Mohsen A.A. Rashwan, Ahmad A. Al Sallab, Hazem M. Raafat, and Ahmed Rafea. 2015. Deep learning framework with confused sub-set resolution architecture for automatic arabic diacritization. *IEEE Transactions on Audio, Speech and Language Processing*, 23(3):505–516.

Nasser Zalmout and Nizar Habash. 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*, pages 704–713, Copenhagen, Denmark. Association for Computational Linguistics.

Yuan Zhang, Chengtao Li, Regina Barzilay, and Kareem Darwish. 2015. Randomized greedy inference for joint segmentation, POS tagging and dependency parsing. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 42–52, Denver, Colorado. Association for Computational Linguistics.

Oumaima Zine and Abdelouafi Meziane. 2017. Novel approach for quality enhancement of Arabic text to speech synthesis. *3rd International Conference on Advanced Technologies for Signal and Image Processing, ATSIP 2017*, pages 1–6.

A Appendix

A.1 Implementation Details

We run LAMAD using Keras library in Python3.7 (with Tensoflow backend). We train our model using Adam optimizer with the default parameters of Keras library such as the learning rate is 0.001. The batch size is 512, the epochs was set to 60 with early stopping if there is no improvement within 10 epochs, and the loss function is categorical cross-entropy. We used the original length sentences in the datasets without increasing the input length such as in (Al-Thubaity et al., 2020). The datasets are randomly partitioned into train (80%), valid. (10%), and test sets (10%).

A.2 Arabic Diacritics

Arabic word is composed of letters (Arabic Alphabets), always written, and diacritics (Table 10), ignored in most of Arabic written texts due to time-consuming and only reliable on Arabic linguistic experts. Diacritics are marks that appear above or below the word letters giving it a pronunciation, meaning, syntactic role and form distinction in various languages such as Arabic (Rashwan et al., 2015) and Yorùbá (Orife, 2018).

Table 10 shows the diacritics types and their positions in the word. Diacritics appear on the internal word stem called core-word diacritics indicating the lexical selection, or on the last letter called case-endings indicating their syntactic role. For example, the word "kutub[2]" كُتُب can have different case endings such Damma كُتُبُ if it is nominal (ex. Subject), Fatha كُتُبَ if it is accusative (ex. Object), Kasra كُتُبِ if the word is genitive (ex. PP predicate). **To formulate the problem of ATD:** giving a sentence consists of a sequence of characters S, for each character S_i in S, find the correct diacritic from Table 10.

A.3 Linguistic Features

Table 11 shows the utilized linguistic features in LAMAD that are extracted from text.

[2] Buckwalter encoding is used (Buckwalter, 2002)

	Types	Name	Shape	Transl.	IPA	Pos. in Word		Types	Name	Shape	Tran sl.	IPA	Pos. in Word
1		Fatha	◌َ	a	/a/	Any	9		Shadda +Fatha	◌َّ	~a	/a/:	Any
2	Short Vowels	Damma	◌ُ	u	/u/	Any	10		Shadda +Damma	◌ُّ	~u	/u/:	Any
3		Kasra	◌ِ	i	i	Any	11		Shadda +Kasra	◌ِّ	~i	i:	Any
4		Sukun	◌ْ	o	ø	Any	12		Shadda +Sukun	◌ّْ	~o	ø:	Any
5		Tanween Fath	◌ً	F	/an/	End	13	Combinations	Shadda +Tanween Fath	◌ًّ	~F	/an/:	End
6	Nunation	Tanween Damm	◌ٌ	N	/un/	End	14		Shadda +Tanween Damm	◌ٌّ	~N	/un/:	End
7		Tanween Kasr	◌ٍ	K	/in/	End	15		Shadda +Tanween Kasr	◌ٍّ	~K	/in/:	End
8	Gemination	Shadda	◌ّ	~	:	Any							

Table 10: Arabic Diacritics along with their types, names, shapes, Buckwalter Transliterations, international phonetic alphabet representation (IPA), and positions in the word.

Feature	Motivation
Characters	Due to the diacritization of Arabic words is affected by the sentence context, each sentence character is represented as a 40-dimensional and 60-dimensional vector for short and long sentence datasets, respectively. The first half elements in the vector are the undiacritized characters before the current character in the sentence while the last half-elements are the undiacritized characters after it, including the current character. A padding token is used when there is no character to feed.
Prior	This feature is represented by a binary 15-dimensional vector for each character indicating whether the character can accept any of the Arabic diacritic marks which is decided from the diacritics observed per word segment in the training set.
Part of Speech (POS)	The diacritization of Arabic word varies according to the POS of word (Chennoufi and Mazroui, 2017). Determining the POS of word in which the character appears can help the model to predict the appropriate character diacritic. The POS tagger presented in (Zhang et al., 2015) is used.
Gender and Number	The agreement of gender (Male, Female, or unknown) and number (singular, plural, double,or unknown) of the word may allow or disallow specific case ending diacritization. The number/gender tagger introduced in (Zhang et al., 2015) is used to extract gender and number information.
Named Entity	This feature is a binary value that determines whether the word in which the character appears is named entity. Arabic named entities mostly have Sukun case endings. As a result, this feature may help to predict the diacritic of the case ending of named entity word. The simple approach for Named entity recognition from Arabic text presented in (Darwish and Gao, 2014) is used to extract the named entity from the corpora.
Segment Position	The position of the character in a word segment may affect its diacritization. For example, the character that comes at the beginning or the middle never has Tanween diacritics. We mark the characters that comes at the beginning, middle or end of the segments as "B", "M" and "E". If the character appears in a single character segment, it marks as "S". Farasa segmenter (Darwish and Mubarak, 2016) that have achieved high accuracy segmentation is used.
Affixes and Stem	Determining whether a character appears in the affixes or the word stem influences the character diacritization (Chennoufi and Mazroui, 2017). A binary 3-dimensional vector represents each character to decide whether the character came in the prefix, stem, or suffix part of the word.
Case Ending	A binary value feature which determines whether the character expect case ending or core word diacritic.

Table 11: Linguistic features used in LAMAD with motivation