Mansoura Journal of Computers and Information Sciences

A Hybrid Approach for Automatic Morphological Diacritization of Arabic Text

Hatem M Noaman
Computer Science Department,
Mansoura University, Egypt
hatemnoaman@yahoo.com

Shahenda S. Sarhan
Computer Science Department,
Mansoura University, Egypt
shahenda_sarhan@yahoo.com

M. A. A. Rashwan
Electronics and Communications Department,
Cairo University, Egypt
mrashwan@RDI-eg.com

ABSTRACT

Arabic Modern texts are commonly written without diacritization, which is a critical task for other Arabic processing tasks as word sense disambiguation, automatic speech recognition, and text to speech, where word meaning or pronunciation is decided based on the diacritic signs assigned to each letter.

This paper presents a novel approach for automatic Arabic text diacritization using deep encode-decode recurrent neural networks that is followed by several text correction techniques, to improve the overall system output accuracy. Experimental results of the proposed system on Wikinews test set show superior performance and are competitive with those of the-state-of-the-art diacritization methods. Namely, our method achieves morphological diacritization Word Error Rate (WER) 3.85% and Diacritic Error Rate (DER) 1.12%.

Keywords

Arabic Natural Language Processing ; Automatic Morphological Diacritization; deep encode-decode recurrent neural networks.

1. INTRODUCTION

The Arabic language and other languages based on Arabic letters have diacritics signs which are small characters that are placed over other basic characters. They play a crucial role in indicating how each letter in the word will be pronounced. The Arabic language has 8 different signs that can be used to diacritize any given character as summarized in Table 1. It shows the different diacritic signs associated with Arabic letter Taa ('ت'), besides this list, combination of "Shadda" with "Fatha", "Kasra" or "Damma" can be used to emphasis more stress pronunciation over a given character.

Most modern standard Arabic texts are written without using the diacritic signs. This adds ambiguity especially in syntax and semantic analysis levels, where single word can have more than one interpretation. For example: the word Elm ("علم") can have more than 13 forms with different meanings according to its position in the text [1]. Arabic diacritic signs are derived by native speakers easily with their prior knowledge of language rules and their ability to infer word meaning from its context. On the other side, texts without diacritic signs add more challenge for Arabic language processing for many tasks like Text-To-Speech (TTS), machine translation, language analysis... etc. Diacritic signs restoration is a process of automatically converting text without or with partially diacritic signs to a fully diacritized text.

Table 1: Different diacritic forms of the Arabic letter Taa ('ت')

Diacritic	Diacritized Letter(Taa)	Transliteration	Unicode
Fatha	تَ	A	U+064E
Kasra	تِ	U	U+0650
Dumma	تُ	I	U+064F
TanweenFathatan	تً	F	U+064B
TanweenKasratan	تٍ	K	U+064D
TanweenDammatan	تٌ	N	U+064C
Shadda	تّ	~	U+0651
Sukon	تْ	O	U+0652

In this work, we propose a novel diacritization model, based on sequence to sequence neural networks [2]. This type of neural networks architecture adds decoder and encoder layers on top of language model, as shown in Figure 1. This architecture gives the ability to map two different sequences: input and target sequences. In literature, this model is widely used in machine translation between two different languages. In this paper, we adopt this model to solve the automatic Arabic text diacritization. In the encoder step, the model converts an input sequence, which is the input word without diacritization into a fixed representation. In the decoder step, the language model is trained on both the output sequence (words with diacritization) as well as the fixed representation from the encoder.

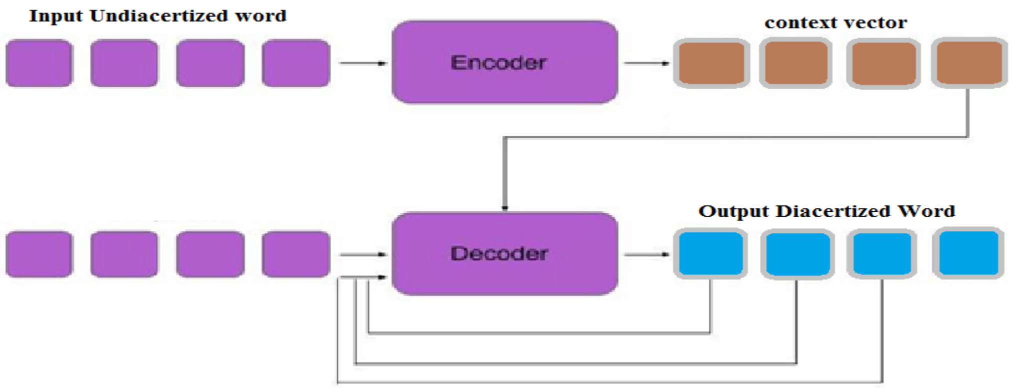

Figure 1: The Basic sequence to sequence neural networks model [2]

The rest of this paper is organized as follows. Section 2 summarizes the related work while Section 3 presents the proposed model. Sections 4 and 5 are, respectively, dedicated to the technical implementation details of the proposed approach and the experimental results. Finally, conclusions and future work are given in Section 6.

2. Related work

Different methods have been investigated to handle Arabic text diacritization problem. The previous work can be divided into 3 common approaches: rule-based, statistical, and hybrid approaches [3]. Rule-based approach tries to solve the problem of Arabic diacritization by applying a set of rules derived from Arabic language morphology and grammar systems. This approach was introduced by El-Sadany and Hashish (1988) [4]. Their work uses morphophonemic and morphographemic rules extracted from vocabulary, morphological analyzer, and grammar module. Later, El-Imam (2004) [5] and Shaalan (2010) [6] used the same approach to restore Arabic diacritization signs. However, Shaalan (2010) [6] only predicts the missing diacritics. Another approach was investigated to handle automatic Arabic diacritization problem based on statistical models. Hidden Markov model (HMM) combined with Viterbi search algorithm were proposed by Gal (2002) [8]. Gal's model archives 86% accuracy and most errors are Out-Of-Vocabulary (OOV) words. Another work related to Arabic speech recognition is done by Kirchhoff et al. (2002) [9]. They tried to improve the diacritization accuracy using maximum likelihood unigram prediction. Their results are 28% word error rate, and 9% if last character diacritization is ignored. Another statistical model based on trigram language model and dynamic programming was proposed by Hifny (2012) [10], where each sequence is assigned a probability by the language model and the most probable sequence is selected by dynamic programming algorithm. Smoothing techniques are used to handle Out-Of-Vocabulary (OOV) words. The reported results of this work were 3.4% and 8.9% for diacritic error rate and word error rate, respectively. Azim et al. (2012) [11] combined hidden Markov model (HMM) and acoustic data model with conditional random fields text-based model. They achieved 1.6 and 5.2%, respectively which is considered a very high accuracy. Nevertheless, we cannot use it as acoustic data is not always available for text. Abandah, et al. (2015) [12] used bidirectional long short-term memory (b-LSTM) deep neural networks to restore missed Arabic diacritic signs. They achieved 2.72% and 9.07% diacritic and word error rates respectively, on LDC ATB3 dataset. Adding "Tashkeela" corpus data, their model's results was improved to 2.09% and 5.82% for diacritic and word error rates, respectively.

Hybrid approaches try to combine several knowledge resources and merge rule-based and statistical-based techniques to handle Arabic diacritization problem. Vergyri and Kirchhoff (2004) [13] proposed the earliest trial to use hybrid sources of knowledge. They combined acoustic, morphological, and contextual data. They tagged each word with one of the different tags provided by BAMA (Buckwalter, 2004 [14]). They achieved a word error rate and diacritic error rate 27.3% and 11.5%, respectively, without counting shadda diacritic. Zitouni et al. (2006) [15] combined maximum entropy with lexical, segment-based and part-of-speech features. Their results are 18% word error rate and 5.5% diacritic error rate for full word, and 7.9% and 2.5%, respectively, for words without case endings. Depending on morphological analysis and disambiguation of Arabic (MADA) system Habash and Rambow (2007)[16] combined word features with support vector machine (SVM) classifier. They achieved 14.9% word error rate and 4.8% diacritic error rate for full word and 5.5% and 2.2%, respectively, for words without case endings. Rashwan and Al-Badrashiny (2011) [17] introduced a two-layer stochastic system. In the first layer, they proposed A* lattice search and n-gram probability to predict the most likely diacritics. If a word is not found, the second layer will factorize the full word into its prefix, root, pattern ,and suffix features, then use A* lattice search and n-gram probability to predict the most likely diacritics sequence. Their work achieved 12.5% word error rate and 3.8% diacritic error rate with case endings, and 3.1% word error rate and 1.2% diacritic error rate without case endings. Said et al. (2013) [18] achieved diacritic and word error rates of 3.6 and 11.4%, respectively, based on morphological analysis using Hidden Markov Models (HMMs). They restore the syntactic diacritic by resolving the syntactic ambiguity. Rashwan et al. (2015)[19] proposed a deep neural network (DNN) based framework to restore the Arabic diacritics. Morphological, syntactic, and context features are used as input to the proposed DNN framework. Then, a post-processing step is applied to DNN output, using contention sub-set resolution network to improve syntactic diacritization performance. They reported 11.6% complete word error rate and 3% morphological word error rate. Darwish et al (2017) [23] applied bigram word-level language model search and back-off stem. If no proper sequence can be found, morphological patterns is used. For the named entities, they proposed transliteration and sequence labeling based diacritization. To restore last letter diacritization sign, they train Support Vector Machine (SVM) model coupled with morphological patterns and linguistic rules. Another hybrid approach was proposed by Fashwan and Alansary (2017) [24] called SHAKKIL system. In this system, the diacritization is restored with two levels, morphological level which has four layers: uni-morphological, rule-based morphological disambiguation, statistical-based disambiguation, and Out Of Vocabulary (OOV) layers. They use syntactic level to add the word last diacritization sign based on shallow parsing technique. Al-Badrashiny et al (2017) [25] proposed three-layered approach to get full word diacritization. In the first layer they use surface form language model (LM) , followed by the morphological level in the second layer, and then finally they back-off to the character level language model (LM). Khorsheed (2018) proposed diacritization using Hidden Markov models (HMM), the proposed system performance is measured using 3 different test sets. The first one isDS1which was built at King Abdulaziz City for Science and Technology [28]. It includes 231 files of proof read Arabic text. The second is DS2 that includes the fully diacritized text from the Holy Quran, and the third is DS3 which consists of 54,463 Arabic names, best diacritic error rate results reported by this work are: 26.09% for DS1, 29.07% for DS2 and 19.43% for DS3. However, these system results are very poor compared to the-state-of-the-art results.

In this paper we handle Arabic diacritics restoration as a machine translation problem. This approach has been adopted by researchers based on different machine translation techniques. Elshafei et al. (2006) [20] uses hidden Markov models (HMM) based machine translation approach. They reported accuracy improvement by 2.5% by using preprocessing stage and trigram for selected number of words. Another machine translation based approach proposed by Schlippe et al. (2008) [21] combines the rule-based diacritization system with conditional random fields.

3. Proposed Model

3.1 Arabic Unicode diacritic signs preprocessing and data preparation:

To make our data more suitable to sequence to sequence model, a different code is assigned to each Arabic diacritic signs, as shown in Table 2. Codes vary from 1 to 9 for different diacritic signs, with 1 byte Unicode representation, from 12 to 15 for 2 bytes Unicode representation and zero value indicates a letter without diacritic sign. At the stage of data preprocessing and preparation, input text is buffered character by character. In Arabic Unicode system, each character is represented by two bytes and diacritic signs are represented as separated Unicode. As shown in Table 2, most signs are represented by two bytes except signs that include shadda sign combined with another sign will represented by 4 bytes.

Figure 2 illustrates data preparation flowchart. The given input raw is in the form of s = {c1,c2,... cn} where s is the input diacritezedword , ci is two bytes Unicode ith character representation and n is the input word length. The final code value of the current character is based on whether this character has a diacritic sign attached to it or not. If exists, we have to make difference between signs with two and four bytes representation. As shown in the flowchart, the first character is read. Based on its Unicode value, it is decided whether it is an Arabic character or not. By default, we get rid off any non-Arabic characters or diacritic signs in the input text. In general, there are three common cases:

1- If the current character (ci) is an Arabic letter and the next character (ci+1) is another Arabic letter. This means that this character has no diacritic sign and its diacritic code will be zero and the current character position (i) is incremented by 1.
2- If the current character (ci) is an Arabic letter and the next character (ci+1) is an Arabic diacritic sign, and the character (ci+2) is an Arabic letter. This means that the current letter is diactertized by a diacritic sign that has 2 bytes Unicode representation. Its value, denoted by (dx), is assigned from one value between 1 and 9 from Table 2. The current character position (i) is then incremented by 2.
3- If the current character (ci) is an Arabic letter and the next two characters (ci+1, ci+2) are Arabic diacritic signs and the character (ci+3) is an Arabic letter. This means that the current letter is diactertized by a diacritic sign that has 4 bytes Unicode representation. Its value, denoted by (dy), is assigned from one values between 12 and 15 from table 2. The current character position (i) is then incremented by 3.

Table 2: The proposed codes and Unicode bytes number of the Arabic diacritic signs

Arabic diacritic	code	Unicode bytes
without diacritic	**0**	**0**
TanweenFathatan	**1**	**2**
TanweenDammatan	**2**	**2**
TanweenKasratan	**3**	**2**
Fatha	**4**	**2**
Dumma	**5**	**2**
Kasra	**6**	**2**
Shadda	**8**	**2**
Sukon	**9**	**2**
Shaddah + Fataha	**12**	**4**
Shaddah + Damma	**13**	**4**
Shaddah + Kasra	**14**	**4**
Shaddah + Dammatan	**16**	**4**
Shaddah + Fatahtan	**15**	**4**

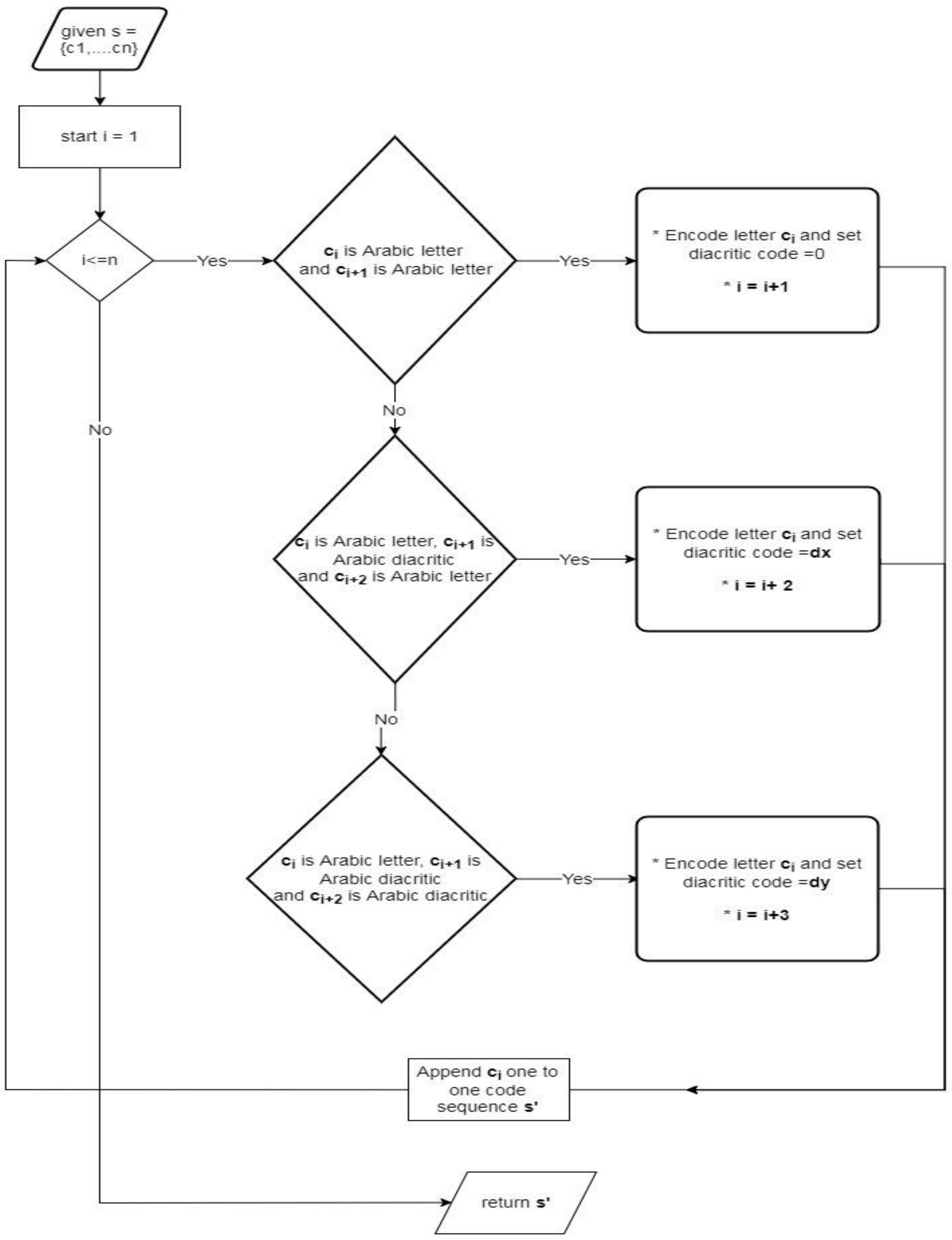
given s = {c1,....cn}
start i = 1
i<=n
Yes
No
c_i is Arabic letter and c_{i+1} is Arabic letter
Yes
* Encode letter c_i and set diacritic code =0
* i = i+1
No
c_i is Arabic letter, c_{i+1} is Arabic diacritic and c_{i+2} is Arabic letter
Yes
* Encode letter c_i and set diacritic code =dx
* i = i+ 2
No
c_i is Arabic letter, c_{i+1} is Arabic diacritic and c_{i+2} is Arabic diacritic
Yes
* Encode letter c_i and set diacritic code =dy
* i = i+3
Append c_i one to one code sequence s'
return s'

Figure 2: Proposed data preparation flowchart.

After applying the encoding flowchart shown in Figure 2, the output character sequence s'={c1',c2',…..cn'} where s' is the input word numeric representation, ci' is a numeric encoding that represents a given ith Arabic character with its diacritic sign as a single value and n is the input word length. Table 3 illustrates an example of the undiacertized word (علم Elm) and 4 of its different diacrtization styles. Table 3 also shows the corresponding Buckwalter transliteration, Unicode values and proposed single character encoding.

3.2 Proposed Model Architecture:

This work suggests using deep encoder-decoder neural networks to handle Arabic diacritization signs restoration problem, both input and output sequence may not be equal in size. Therefore, encoder part of the proposed model will convert the input sequence into context vector, then the context vector will be used as input to the proposed model decoder, both of the proposed encoder and decoder modules are deep LSTM neural network with 4 layers each layer consists of 1000 neuron. Network weights are randomly initialized to values between -1 and 1, and the starting learning rate is 0.01.

Table 3: The different diacrtization styles for undiacertized word (علم Elm), Buckwalter transliteration, Unicode values and proposed single character encoding representations.

Arabic word	Buckwalter transliteration	Unicode decimal two byterepresentation	Single char code representation
عَلِمَ	Ealima	[657, 678], [668, 680], [669, 678]	[457, 668, 469]
عُلِمَ	Eulima	[657, 679], [668, 680], [669, 678]	[557, 668, 469]
عِلْمُ	Eilomu	[657, 680], [668, 682], [669, 679]	[657, 968, 569]
عَلَّمَ	Eal~ama	[657, 678],[668, 681,678] , [669, 678]	[457 1268 469]

3.3 Encoder-Decoder Deep Neural Network training module:

After preparing the fixed length encoding scheme, presented in previous Section, two parallel files are created. The first one for the undiacrtized input word codes. The second file contains the diacritized output word codes. The encoder-decoder module first reads the source words using an encoder to build an encoder context vector. Based on this vector, the decoder will take it as an input and outputs diacritized word character sequence as illustrated in Figure 1. This iterates several epochs for the dataset. After completing a predefined number of epochs, the output of this module will be saved as the trained model. The general training framework is shown in Figure 3.

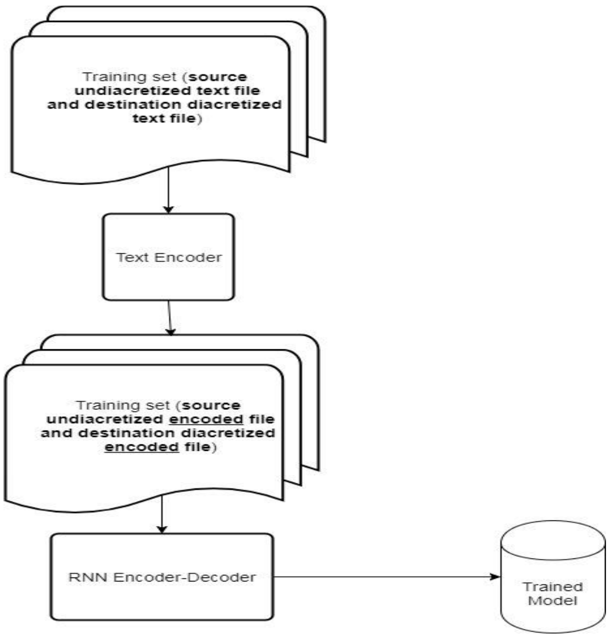
Training set (source undiacretized text file and destination diacretized text file)
Text Encoder
Training set (source undiacretized encoded file and destination diacretized encoded file)
RNN Encoder-Decoder
Trained Model

Figure 3: Training module flowchart.

3.4 Encoder-Decoder Deep Neural Network testing module:

After building the training model, word diacrtization sequence can be inferred by converting input test text into its one-to-one encoding as given in Section 3.1. Based on the trained model and decoder neural network, we will get the system output in the form of sequence of characters one-to-one codes. This code will have different diacritic values based on the proposed system selection for each character. Sometimes the character codes produced by the proposed system are replaced. In this case, we restore the modified characters to its original form. This step will not affect the diacrtization accuracy, because our concern will be the final diacrtization signs accuracy.

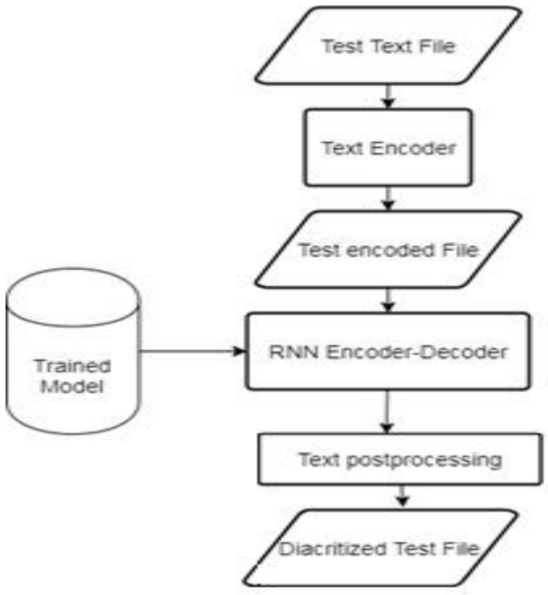

Figure 4 Test Model flowchart

Some postprocessing text correction operations are also applied on the final output text based on some well-known fixed Arabic diacrtization rules. Although the system output may be true without applying this set of rules, we include them for reliability and robustness of our method. It is noticed that the system performance and final output are enhanced after applying these rules as shown in the results section. These operations are:

1- Sukun correction: There are two writing styles for sukundiacritic. One adds sukun and the other omits it. Both styles are correct and give the same meaning [12]. If there is sukun in the proposed model output, we omit it from the test set and output sequences to unify dealing with this case.

2- Missing diacritic sign before Madd letters (Alef, Waw, and Yaa): If the output letter is madd letter (Alef, Waw, or Yaa) , we change diacritic for the letter before this letter to the suitable diacritic with this madd letter, Fatha for Alef, Damma for Waw and Kasra for Yaa.

3- Alef+Lam Type correction: In the Arabic language there are two types of Alef+Lam letters combination in the start of words: Alef+LamShamsia and Alef+LamQamria. With Alef+LamShamsia, the next letter must have Shadda diacritic sign combined with this letter diacritic sign. Alef letter must have Fatha and Lam is left without any sign. In the case of Alef+LamQamria, the next letter cannot have Shadda diacritic sign.

4- Latin Words Ambiguity:

5- Some words which are arabized from other languages may have more than one correct diacritization form as (حَزِيرَان ،حُزِيرَان). We choose to consider that both forms are true diacritization for such words. Some Arabic words have the same case. In this work, a list of 83 words was prepared that appear in the test set and have more than one possible correct diacritization from.

4. Implementation

The proposed System was implemented using two different languages. C# 3.5 was used for text encoding/decoding and postprocessing modules. Python 3 was used to build encoder-decoder neural network model based on TensorFlow 1.4.1® deep learning framework. Model training and testing were carried out on Intel(R) Xeon(R) CPU X5650 2.67GHz with 64GB RAM installed with GeForce GTX 980 Ti GPU card. Input and output sequences are presented as plain text file.

5. Experimental Results

5.1 Accuracy measures

In this paper, we present the performance using two measures: Diacritization Error Rate (DER) and Word Error Rate (WER). The DER represents how many letters diacritics are incorrectly restored. The WER represents the rate of words that have any incorrect diacritization error. These two measures are reported for morphological (core word) where the diacritic assigned to the last character of the core of the word (the stem) is not considered.

5.2 Experiment's Data Set

Reported results in this paper were obtained by training the proposed approach using fully diacritized Arabic corpus that contains about 1 million words from "Aljazeera" news annotated by RDI company in Egypt. In next subsections results were obtained based on test data proposed by Darwish et al. (2017) [23] which contains about 18300 words extracted from the Wikinews articles, published between 2013 and 2014 and covers 7 different themes (politics, economics, health, science and technology, sports, arts, and culture).

5.3 Performance Enhancement

To improve our proposed system Diacritization Error Rate (DER), some postprocessing and correction rules are applied to the output. Table 4 summarizes the Word Error Rate (WER) and Diacritic-Error-Rate (DER) with error reduction ratio through each correction step and finally after applying all the steps as the final system results in the final row

From results in Table 4, the proposed system Word Error Rate (WER) results are improved by 5.88 % and Diacritic Error Rate (DER) about 1.99 for morphological diacritization. Latin words correction steps improves Word Error Rate (WER) results from 8.12% to 3.85% which shows that words with original Arabic origins can be handled with good accuracy by the proposed system. Other words that have ambiguity in their diacritization because they are not original Arabic words and words that have more than correct diacritization confuse the proposed algorithm about how to decide the correct diacritization. Hence, we list these words out and consider both the words diacritizations are true.

Table 4. Diacritic-Error-Rate (DER) reduction based on post-processing and correction rules

	WER (%)	**Error Reduction**	**DER(%)**	**Error Reduction**
Without Correction	**9.73**	**-----**	**3.11**	**-------**
+ Sukun Correction	**9.57**	**0.16**	**2.98**	**0.13**
+ Missing diac before Madd letters correction	**8.62**	**1.11**	**2.69**	**0. 42**
+ Alef+Lam Type correction	**8.12**	**1.61**	**2.45**	**0.66**
+Latin words correction	**3.85**	**5.88**	**1.12**	**1.99**

5.4 Final Results and comparisons

Table 5 shows the comparison between our proposed approach and the state-of-the-art systems, on Wikinews test set, in terms of morphological diacritization Word Error Rate (WER) and Diacritic Error Rate (DER).

Table 5: Baseline, related work, and proposed system Word-Error-Rate (WER) and Diacritic-Error-Rate (DER) for core word diacritization results.

	WER	DER
MADAMIRA (2014) [7][23]	6.73	1.91
Abandah et al. (2015) [26]	4.34	1.38
Rashwan et al. (2015) [19][23]	3.04	0.95
Belinkov and Glass (2015) [1][23]	14.87	3.89
Darwish et al. (2017) [23]	3.29	1.06
Proposed Approach	3.85	1.12

From the listed results in Table 5, it can be observed that the proposed approach results are better than MADAMIRA [7] and Belinkov and Glass [1] results according to Diacritic Error Rate(DER) morphological diacritization while they are very close to Rashwan et al. [19] (2015) and Darwish et al. (2017) [23] in the case of morphological diacritization Word Error Rate(WER) and Diacritic Error Rate (DER) using WikiNews test set. It was noticed that the proposed system results are improved remarkably compared to Belinkov and Glass [1] which is very similar to the proposed approach. However, their work is based on the recurrent neural networks without adding any morphological or syntactical features for input text. Also, the proposed approach outperformed MADAMIRA [7] where diacritic signs are restored based on the morphological interpretation of the word. Abandah et al. (2015) [26] tried to use deep bidirectional long short-term memory (LSTM) network to handle Arabic language diacrtization problem, comparing their results to the proposed model results shows that results of the proposed model achieves about 0.5% for Word Error Rate (WER) and 0.26% Diacritic Error Rate (DER) enhancement. Although Rashwan et al. [19] and Darwish et al. (2017) [23] results slightly outperform our proposed model results, it is worth noting that their models were designed to contain different levels of word features to help in deciding the appropriate word diacrtization form.

6. Conclusion

In this paper, we propose a deep encoder-decoder neural networks based model, followed by text post-processing steps that handle the problem of Arabic text diacritization. The proposed approach can restore the missing morphological diacritics at high accuracy without any additional resource of data except the fully diacritized dataset. The Proposed approach achieves full coverage of Arabic words with no need to handle the Out Of Vocabulary (OOV) problem. The WER of the core word diacritization is 3.85% and DER is 1.12% which are very competitive to the state-of-the-art techniques. As a future work, we will try to handle syntactic diacritization (full word). Also we intend to add word context features to improve diacritization accuracy.

ACKNOWLEDGMENTS

We would like to thank Kareem Darwish, Hamdy Mubarak, and Ahmed Abdelali for providing us with Wikinews test set.

7. REFERENCES

[1] Belinkov, Y., & Glass, J. (2015). Arabic diacritization with recurrent neural networks. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (pp. 2281-2285).

[2] Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. In Advances in neural information processing systems (pp. 3104-3112).

[3] Azmi, A. M., &Almajed, R. S. (2015). A survey of automatic Arabic diacritization techniques. Natural Language Engineering, 21(3), 477-495.

[4] El-Sadany, T., & Hashish, M. (1988, April). Semi-automatic vowelization of Arabic verbs. In 10th NC Conference, Jeddah, Saudi Arabia.

[5] El-Imam, Y. A. (2004). Phonetization of Arabic: rules and algorithms. Computer Speech & Language, 18(4), 339-373.

[6] Shaalan, K. (2010). Rule-based approach in Arabic natural language processing. The International Journal on Information and Communication Technologies (IJICT), 3(3), 11-19.

[7] Pasha, A., Al-Badrashiny, M., Diab, M. T., El Kholy, A., Eskander, R., Habash, N., ...& Roth, R. (2014, May). MADAMIRA: A Fast, Comprehensive Tool for Morphological Analysis and Disambiguation of Arabic. In LREC (Vol. 14, pp. 1094-1101).

[8] Gal, Y. A. (2002, July). An HMM approach to vowel restoration in Arabic and Hebrew. In Proceedings of the ACL-02 workshop on Computational approaches to semitic languages (pp. 1-7). Association for Computational Linguistics.

[9] Kirchhoff, K., Bilmes, J., Das, S., Duta, N., Egan, M., Ji, G. &Schone, P. (2003, April). Novel approaches to Arabic speech recognition: report from the 2002 Johns-Hopkins summer workshop. In Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03). 2003 IEEE International Conference on (Vol. 1, pp. I-I). IEEE.

[10] Hifny, Y. (2012). Smoothing techniques for Arabic diacritics restoration. In Proceedings of the 12th Conference Lang. Eng.(ESOLEC'12) (No. 1, pp. 6-12).

[11] Azim, A. S., Wang, X., & Chai, S. K. (2012). A weighted combination of speech with text-based models for Arabic diacritization. In Thirteenth Annual Conference of the International Speech Communication Association.

[12] Abandah, G. A., Graves, A., Al-Shagoor, B., Arabiyat, A., Jamour, F., & Al-Taee, M. (2015). Automatic diacritization of Arabic text using recurrent neural networks. International Journal on Document Analysis and Recognition (IJDAR), 18(2), 183-197.

[13] Vergyri, D., & Kirchhoff, K. (2004, August). Automatic diacritization of Arabic for acoustic modeling in speech recognition. In Proceedings of the workshop on computational approaches to Arabic script-based languages (pp. 66-73). Association for Computational Linguistics.

[14] Buckwalter, T. (2004). Buckwalter Arabic Morphological Analyzer, v2. 0 edn. Linguistic Data Consortium, Philadelphia.

[15] Zitouni, I., Sorensen, J. S., &Sarikaya, R. (2006, July). Maximum entropy based restoration of Arabic diacritics. In Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics (pp. 577-584). Association for Computational Linguistics.

[16] Habash, N., &Rambow, O. (2007, April). Arabic diacritization through full morphological tagging. In Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Companion Volume, Short Papers (pp. 53-56). Association for Computational Linguistics.

[17] Rashwan, M. A., Al-Badrashiny, M. A., Attia, M., Abdou, S. M., &Rafea, A. (2011). A stochastic Arabic

diacritizer based on a hybrid of factorized and unfactorized textual features. IEEE Transactions on Audio, Speech, and Language Processing, 19(1), 166-175.

[18] Said, A., El-Sharqwi, M., Chalabi, A., & Kamal, E. (2013, June). A hybrid approach for Arabic diacritization. In International Conference on Application of Natural Language to Information Systems (pp. 53-64). Springer, Berlin, Heidelberg.

[19] Rashwan, M. A., Al Sallab, A. A., Raafat, H. M., &Rafea, A. (2015). Deep learning framework with confused sub-set resolution architecture for automatic Arabic diacritization. IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP), 23(3), 505-516.

[20] Elshafei, M., Al-Muhtaseb, H., &Alghamdi, M. (2006). Statistical methods for automatic diacritization of Arabic text. In The Saudi 18th National Computer Conference. Riyadh (Vol. 18, pp. 301-306).

[21] Schlippe, T., Nguyen, T., & Vogel, S. (2008, October). Diacritization as a machine translation problem and as a sequence labeling problem. In AMTA-2008. MT at work: In Proceedings of the Eighth Conference of the Association for Machine Translation in the Americas (pp. 270-278).

[22] Aya S. M. Hussein, Mohsen A. A. Rashwanand Amir F. Atiya (2016), Arabic Full Text Diacritization using Light Layered Approach. Artificial Intelligence and Machine Learning Journal, ISSN: 1687-4846, Vo. 16, No. 1, Delaware, USA, December 2016

[23] Darwish, K., Mubarak, H., &Abdelali, A. (2017). Arabic Diacritization: Stats, Rules, and Hacks. In Proceedings of the Third Arabic Natural Language Processing Workshop (pp. 9-17).

[24] Fashwan, A. &Alansary, S. (2017). SHAKKIL: An Automatic Diacritization System for Modern Standard. In Proceedings of the Third Arabic Natural Language Processing Workshop (pp. 84-93).

[25] Al-Badrashiny, M., Hawwari, A., &Diab, M. (2017). A Layered Language Model based Hybrid Approach to Automatic Full Diacritization of Arabic. In Proceedings of the Third Arabic Natural Language Processing Workshop (pp. 177-184).

[26] Abandah, G. A., Graves, A., Al-Shagoor, B., Arabiyat, A., Jamour, F., & Al-Taee, M. (2015). Automatic diacritization of Arabic text using recurrent neural networks. International Journal on Document Analysis and Recognition (IJDAR), 18(2), 183-197.

[27] Khorsheed, M. S. (2018). Diacritizing Arabic Text Using a Single Hidden Markov Model. IEEE Access, 6.