Highly Effective Arabic Diacritization using Sequence to Sequence Modeling

Hamdy Mubarak **Ahmed Abdelali** **Hassan Sajjad** **Younes Samih** **Kareem Darwish**
Qatar Computing Research Institute, HBKU Research Complex, Doha 5825, Qatar
{hmubarak,aabdelali,hsajjad,ysamih,kdarwish}@qf.org.qa

Abstract

Arabic text is typically written without short vowels (or diacritics). However, their presence is required for properly verbalizing Arabic and is hence essential for applications such as text to speech. There are two types of diacritics, namely core-word diacritics and case-endings. Most previous works on automatic Arabic diacritic recovery rely on a large number of manually engineered features, particularly for case-endings. In this work, we present a unified character level sequence-to-sequence deep learning model that recovers both types of diacritics without the use of explicit feature engineering. Specifically, we employ a standard neural machine translation setup on overlapping windows of words (broken down into characters), and then we use voting to select the most likely diacritized form of a word. The proposed model outperforms all previous state-of-the-art systems. Our best settings achieve a word error rate (WER) of 4.49% compared to the state-of-the-art of 12.25% on a standard dataset.

1 Introduction

Arabic uses two types of vowels, namely long vowels, which are explicitly placed in the text, and short vowels, which are diacritic marks that are typically omitted during writing. In order to read Arabic words properly, readers need to reintroduce the missing diacritics. Therefore, accurate diacritic recovery is essential for some applications such as text-to-speech. There are, in turn, two types of Arabic diacritics, namely core-word diacritics (CW), which specify lexical selection, and case endings (CE), which typically indicate syntactic role. For example, the word "AlElm"[1] can accept many possible core-word diacritics depending on the intended meaning such as: "AaloEalam" (the flag) and "AaloEilom" (the knowledge/science). In our training corpus, 17.1% of the word-cores have more than one valid diacritized form. In the sentence "Zahara AaloEalam**u**" (the flag appeared), "AaloEalam**u**" is the subject and takes the case ending "**u**", and in the sentence "ra>ayotu AaloEalam**a**" (I saw the flag), "AaloEalam**a**" is the object and takes the case ending "**a**". Aside from function words, past tense and accusative verb forms, and foreign names, most words can accept different case-endings depending on context.

In this paper, we introduce a unified model for both diacritic types while improving upon the state-of-the-art. Specifically, we approached the task as a sequence-to-sequence (seq2seq) problem (Cho et al., 2014); taking advantage of the recent advancements in Neural Machine Translation (NMT) (Britz et al., 2017; Kuchaiev et al., 2018) among other applications where seq2seq models made a breakthrough (Yu et al., 2016; Witten et al., 2016; Abadi et al., 2016). Using the analogy of translation which employs a sequential encoder and a sequential decoder, the input undiacritized text will be encoded and then decoded into diacritized form.

As we show later, directly applying a seq2seq model at sentence level using word or character representations produces nearly unusable results that are much worse than the state-of-the-art due to word insertions, omissions, and substitutions. Such problems are exaggerated when using word-based models due to Out-Of-Vocabulary words (OOVs). Conversely, character-based models suffer from not learning long-term dependencies. To avoid these problems, we train a seq2seq model on a sliding window of words that are represented using characters, and we employ voting to pick the best most likely diacritized form from different windows. In doing so, we provide suffi-

[1] In this paper, we use Buckwalter transliteration.

Proceedings of NAACL-HLT 2019, pages 2390–2395
Minneapolis, Minnesota, June 2 - June 7, 2019.

cient context to properly guess proper diacritized forms, while stinting the aforementioned undesirable word operations. Further, the use of voting has the effect of picking the most frequent diacritized form obtained from applying the model on different contexts. The resultant system makes full use of NMT machinery to achieve error rates that are 63.3% lower than the best state-of-the-art system[2].

The contributions in this paper are:

- Adaptation of neural machine translation for Arabic diacritic recovery with voting.
- Unified model to handle both core-diacritics and case-endings.
- Substantial improvement over state-of-the-art with 4.49% word error rate compared to 12.25%.

2 Background

One of the first approaches to Arabic diacritization used a Hidden Markov Model (Gal, 2002; Elshafei et al., 2006) that was applied to the Qur'an achieving double digit word error rates (WER). Vergyri and Kirchhoff (2004) used acoustic features in conjunction with morphological and contextual constrains to train a diacritizer and reported a 9% and 28% diacritics error rate (DER) without and with CEs. Since words are composed of multiple letters with corresponding diacritics, DER values are typically lower than WER values. Nelken and Shieber (2005) used a cascade of word-level, character-level, and morphological models finite state transducers to attain a Word Error Rate (WER) of 7.33% without CE and and 23.61% WER with CE. Zitouni et al. (2006) trained a maximum entropy model for sequence classification using the LDCs Arabic Treebank (ATB) and attained a WER of 18% (with CE) on 600 articles from An-Nahar Newspaper.

Habash and Rambow (2005); Rashwan et al. (2011); Bebah et al. (2014); Pasha et al. (2014); Metwally et al. (2016); Darwish et al. (2017) combined morphological features along with POS tagging information and n-gram language models. MADA-D system (Habash and Rambow, 2007) achieved a 5.5% and 14.9% WER respectively without and with CE. MADAMIRA (Pasha et al., 2014) ranks a list of candidate analyses from the Buckwalter analyzer (Buckwalter, 2004) using an SVM classifier and achieves 19.0% and 6.7% WER with and without CE respectively. Similarly, Microsoft Arabic Toolkit Services (ATKS) diacritizer (Said et al., 2013) uses a rule-based morphological analyzer that produces possible analyses and an HMM in conjunction with rules to guess the most likely analysis. They report WER of 11.4% and 4.4% with and without CE. Darwish et al. (2017) used a Viterbi decoder to guess core word diacritics and SVM-rank to guess case-endings and their system achieves a WER of 3.29% and 12.77% for words without and with CE. They trained their system using a large corpus of roughly 4.5 million words.

More recent work employed different neural architectures to model the diacritization problem. Abandah et al. (2015) used a biLSTM-based recurrent neural network trained on the same dataset as (Zitouni et al., 2006), and they report a WER of 9.1% including CE on ATB. Similar architectures were explored but with lower results (Rashwan et al., 2015; Belinkov and Glass, 2015). Azmi and Almajed (2015); Osama Hamed (2017) survey recent work on Arabic diacritization. They concluded that: reported results are often incomparable due to the usage of different test sets; a large unigram LM for CW diacritic recovery is competitive with many of the systems in the literature. In this paper, we compare our results to those of Said et al. (2013); Pasha et al. (2014); Rashwan et al. (2015); Belinkov and Glass (2015); Darwish et al. (2017) on standard test set.

3 Methodology

Representation Unit. The diacritization of Arabic is a word-internal property dependent on both character and word-level contexts. Therefore, we consider characters as units of representation. We represent source sentences as a sequence of characters by adding a space after every character and a word boundary "_" between words. The target side, which is fully diacritized, is split into a sequence of subword units each consisting of a letter and its diacritic(s). For example, source word "AlElm" would be represented as "A/l/E/l/m" and its diacritized target "AaloEalamu" as "Aa/lo/Ea/la/mu".

The character-level representation has several benefits, such as reducing the vocabulary size and avoiding OOV words. The splitting of diacritized

[2]Patent pending.

words into subword units simplifies the problem as there will be identical number of source and target tokens in a parallel sentence. Later, we support our design decisions with results in the experiments section. Subwords (BPE (Sennrich et al., 2016)) have been used as a defacto standard in building NMT systems. They are a natural choice to handle unknown words. However, BPE does not fit in our scenario as it may create source and target segments of different lengths. In the Arabic diacritization problem, both source and target words and characters are strictly tied to each other and loosening it would result in sub-optimal performance and may generate unexpected errors.

Context Window. The diacritization of Arabic words is highly sensitive to context. Character representations significantly increase the size of the source and target sequences. This leads to a well known limitation of character-based LSTM-based models, namely poor handling of long range dependencies (Sennrich, 2017). An easy fix is to split sentences greater than a certain length into multiple lines. However, boundary words may loose context in the newly created sequences. To handle this, we propose to keep a fixed size context window c for every word. Given a sentence, we use a sliding context window to split it into segments of overlapping windows of size c as in Table 1. This fixes the problems of both long range dependencies and context of neighboring words. We are further aided by the fact that local context can conclusively determine the correct diacritization in the vast majority of cases.

Voting. As shown in Table 1, the sliding window approach replicates a word present in a sentence c times. At test time, a word may get different diacritized forms from different contexts. We use voting to choose the most frequent diacritized form from the c predictions. In case of a tie, we favor the context where the word appears in the middle.

Sentence	w_1	w_2	w_3	w_4	w_5
c=3	w_1	w_2	w_3		
		w_2	w_3	w_4	
			w_3	w_4	w_5

Table 1: Example sentence: w_1 w_2 w_3 w_4 w_5 with context window c of size 3.

	Train		Test		
	Total	Uniq	Total	Uniq	OOV
Diacritized	4.5M	333k	18.3k	7.9k	5.0%
Undiacritized		209k		6.8k	3.3%

Table 2: Number of tokens in training and test data.

Seq2Seq Model. We use a seq2seq model consisting of three main components: i) Encoder, ii) Decoder, and iii) Attention. Given a source sentence $s = w_1, ..., w_N$ and a target sentence $t = v_1, ..., v_N$, the encoder models the source sentence and computes a set of hidden states $h = h_1, ..., h_N$. The attention mechanism (Bahdanau et al., 2014) computes a weighted average of these hidden states from the previous decoder state, known as the context vector c_i, while decoder models the target sentence. The seq2seq model is trained jointly on a large parallel corpus by maximizing the log-likelihood of the data:

$$log\ p(t|s) = \sum_{i} \sum_{j=1}^{|t^i|} log\ p(v_j^i | v_1^i, ..., v_{j-1}^i, s^i) \quad (1)$$

where s^i and t^i are the i^{th} source and target sentences. In addition, we experiment with a Transformer model (Vaswani et al., 2017), which is an attention based architecture without LSTM, and compare its results to the aforementioned seq2seq model with attention.

4 Experimental Setup and Results

Data. We used a modern diacritized corpus of 4.5 million tokens that covers a wide range of topics such as politics, religion, sport, health, and economics. For testing, we used the freely available WikiNews corpus (18,300 words) (Darwish et al., 2017) as a test set, which covers a variety of genres. Table 2 reports the size of the training and test sets including the unique diacritized and undiacritized tokens and the percentage of OOVs in the test set that don't appear in the training set. We randomly used 10% of the train data for validation and the rest for training. We used a sequence length of 100, 500 and 7 tokens for word-, character-, and window-based systems respectively. The vocabulary is restricted to 100k words types and 1,000 character units.

System Settings. The settings for LSTM-based Seq2Seq model were: word embeddings and LSTM states = 512; 2 layer unidirectional LSTM;

Exp	Description	Core WER%	CE WER%	WER%	DER%	OOV%
01	Baseline Word	44.29	54.95	54.31	41.62	13.03
02	Baseline Char	41.29	41.95	48.31	36.62	0.00
03	Word 7g	14.83	19.01	20.69	18.92	11.04
04	Char 7g	2.78	6.11	8.32	2.19	0.00
05	Word 7g+overlap	14.50	16.57	18.05	18.14	10.97
06	Char 7g+overlap	2.04	3.23	4.94	1.34	0.00
07	Char 3g+overlap+voting	2.31	5.97	7.79	2.01	0.00
08	Char 5g+overlap+voting	2.37	3.57	5.49	1.49	0.00
*09	Char 7g+overlap+voting	1.99	3.07	4.77	1.30	0.00
10	Char 11g+overlap+voting	3.03	3.93	6.40	1.78	0.00
†11	Char 7g+overlap+voting (Transformer)	2.05	3.04	4.77	1.29	0.00
12	Combination *09 +† 11	**1.89**	**2.89**	**4.49**	**1.21**	0.00

Table 3: Diacritization results: *g represents ngram size e.g. 7g means 7-gram context. Experiment 09 and 11 are comparing NMT models – LSTM-based architecture with attention mechanism and Transformer model

and dropout rate = 0.3. The setting for the Transformer were: 6 encoder and 6 decoder layers each of size 512; number of attention heads = 8; feed forward dimension = 2048; and dropout = 0.1. We used the OpenNMT (Klein et al., 2017) implementation with tensorflow for all experiments.

System Runs. We conducted a variety of experiments as follows, namely: **Word-level experiments** where the input is a sequence of words and the output is a sequence of diacritized words:
– Baseline Word: uses the full sentences and shows the deficiency of using NMT directly.
– Word 7g: uses non-overlapping windows of 7 words to compare to our best character-level model, which also uses a window of length 7.
– Word 7g+overlap: uses a sliding window of 7 words.
Character-level experiments where the input is represented as a sequence of character and the output as a sequence of diacritized characters:
– Baseline Char: uses the full sentence.
– Char 7g: uses non-overlapping sequences of 7 words.
– Char 7g+overlap: uses a sliding window of 7 words without voting.
– Char ng-overlap+voting: uses a sliding window of n words with voting, where we varied n to equal 3, 5, 7, and 11. When $n = 7$, we experimented with a seq2seq model with attention, a Transformer model, and a combination of both.

Results. Table 3 summarizes the results of our experiments. As the results clearly show, using an NMT model at word or character level produced unusable results. Both Baselines suffer also from excessive repetition of characters that are often meaningless hallucination (e.g. "AalofaA}iti AaloHaAdiy waAlt∼awaAliy Aaloayoiy AloanohaAti AaloanohaAti"). When we limited the context to 7 words, the results improved dramatically, nonetheless, the output still suffered from a high ratio of OOVs. Using characters instead alleviated the OOV problem. The results improved dramatically with contexts of length 7 yielding the best results. Using voting lowered WER rate further, leading to a 4.77% WER. Using a Transformer model led to nearly identical WER to using our NMT model with attention. However their results are somewhat complimentary. Thus, voting on the predictions across both systems improved the results further with a 4.49% WER. Table 4 compares our best system with other systems. The WER of our best system is 63.3% lower than the state-of-the-art.

Setup	WER%
Our System	**04.49**
Microsoft ATKS (Said et al., 2013)	12.25
Farasa (Darwish et al., 2017)	12.76
RDI (Rashwan et al., 2015)	15.95
MADAMIRA (Pasha et al., 2014)	19.02
MIT (Belinkov and Glass, 2015)	30.50

Table 4: Comparison to other systems for full diacritization (Darwish et al., 2017).

Error Analysis. we randomly selected 100 errors word-core and 100 case-ending errors we ascertain the most common error types. For case-ending, the top 4 error types were: long-distance dependency (e.g. coordination or verb subj/obj), which is an artifact of using limited context – 24%

of errors; confusion between different syntactic functions (e.g. N N vs. N ADJ or V Subj vs. V Obj) – 22%; wrong selection of morphological analysis (e.g. present tense vs. past tense) – 20%; and named entities (NEs) – 16%. For long distance dependencies, increasing context size may help in some case, but may introduce additional errors (see Table 3). Perhaps combining multiple context sizes may help. As for word-core, the top 4 errors were: incorrect selection for ambiguous words, where most of these errors were related to active vs. passive voice – 60%; NEs – 32%; borrowed words – 4%; and words with multiple valid diacritized words – 4%.

5 Conclusion

In this paper, we adapted a seq2seq model to build a unified model for Arabic diacritic recovery. We trained the model on a fixed length sliding window of n words that are represented using their characters. We further employed voting to pick the most common diacritized form of a word in different contexts. The adaptation yielded a word error rate of 4.49%, which is 63.3% lower than the best state-of-the-art system. One possible future direction is to use a system combination with varying context sizes with a weighted voting scheme. Further, the explicit inclusion of a large gazetteer of diacritized NEs in the training set would help diacritize them properly. We also want to examine the effect of training data size to determine if more data would yield better results.

References

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. 2016. Tensorflow: a system for large-scale machine learning. In *OSDI*, volume 16, pages 265–283.

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 (IJDAR)*, 18(2):183–197.

Aqil M Azmi and Reham S Almajed. 2015. A survey of automatic arabic diacritization techniques. *Natural Language Engineering*, 21(03):477–495.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Mohamed Bebah, Chennoufi Amine, Mazroui Azzeddine, and Lakhouaja Abdelhak. 2014. Hybrid approaches for automatic vowelization of arabic texts. *arXiv preprint arXiv:1410.2646*.

Yonatan Belinkov and James Glass. 2015. Arabic diacritization with recurrent neural networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2281–2285, Lisbon, Portugal.

Denny Britz, Anna Goldie, Thang Luong, and Quoc Le. 2017. Massive Exploration of Neural Machine Translation Architectures. *ArXiv e-prints*.

Tim Buckwalter. 2004. Buckwalter arabic morphological analyzer version 2.0. *LDC catalog number LDC2004L02, ISBN 1-58563-324-0.*

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Kareem Darwish, Hamdy Mubarak, and Ahmed Abdelali. 2017. Arabic diacritization: Stats, rules, and hacks. In *Proceedings of the Third Arabic Natural Language Processing Workshop*, pages 9–17.

Moustafa Elshafei, Husni Al-Muhtaseb, and Mansour Alghamdi. 2006. Statistical methods for automatic diacritization of arabic text. In *The Saudi 18th National Computer Conference. Riyadh*, volume 18, pages 301–306.

Ya'akov Gal. 2002. An hmm approach to vowel restoration in arabic and hebrew. In *Proceedings of the ACL-02 workshop on Computational approaches to Semitic languages*, pages 1–7. Association for Computational Linguistics.

Nizar Habash and Owen Rambow. 2005. Arabic tokenization, part-of-speech tagging and morphological disambiguation in one fell swoop. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 573–580. Association for Computational Linguistics.

Nizar Habash and Owen Rambow. 2007. 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*, pages 53–56. Association for Computational Linguistics.

G. Klein, Y. Kim, Y. Deng, J. Senellart, and A. M. Rush. 2017. OpenNMT: Open-Source Toolkit for Neural Machine Translation. *ArXiv e-prints*.

Oleksii Kuchaiev, Boris Ginsburg, Igor Gitman, Vitaly Lavrukhin, Carl Case, and Paulius Micikevicius. 2018. Openseq2seq: extensible toolkit for distributed and mixed precision training of sequence-to-sequence models. *arXiv preprint arXiv:1805.10387*.

Aya S Metwally, Mohsen A Rashwan, and Amir F Atiya. 2016. A multi-layered approach for arabic text diacritization. In *Cloud Computing and Big Data Analysis (ICCCBDA), 2016 IEEE International Conference on*, pages 389–393. IEEE.

Rani Nelken and Stuart M Shieber. 2005. Arabic diacritization using weighted finite-state transducers. In *Proceedings of the ACL Workshop on Computational Approaches to Semitic Languages*, pages 79–86. Association for Computational Linguistics.

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

Arfath Pasha, Mohamed Al-Badrashiny, Mona Diab, Ahmed El Kholy, Ramy Eskander, Nizar Habash, Manoj Pooleery, Owen Rambow, and Ryan M Roth. 2014. Madamira: A fast, comprehensive tool for morphological analysis and disambiguation of arabic. In *LREC-2014*, Reykjavik, Iceland.

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

Mohsen AA Rashwan, Mohamed ASAA Al-Badrashiny, Mohamed Attia, Sherif M Abdou, and Ahmed Rafea. 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.

Ahmed Said, Mohamed El-Sharqwi, Achraf Chalabi, and Eslam Kamal. 2013. A hybrid approach for arabic diacritization. In *Natural Language Processing and Information Systems*, pages 53–64, Berlin, Heidelberg. Springer Berlin Heidelberg.

Rico Sennrich. 2017. How grammatical is character-level neural machine translation? assessing mt quality with contrastive translation pairs. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural Machine Translation of Rare Words with Subword Units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, ukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Dimitra Vergyri and Katrin Kirchhoff. 2004. Automatic diacritization of arabic for acoustic modeling in speech recognition. In *Proceedings of the workshop on computational approaches to Arabic script-based languages, COLING'04*, pages 66–73, Geneva, Switzerland. Association for Computational Linguistics.

Ian H Witten, Eibe Frank, Mark A Hall, and Christopher J Pal. 2016. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann.

Haonan Yu, Jiang Wang, Zhiheng Huang, Yi Yang, and Wei Xu. 2016. Video paragraph captioning using hierarchical recurrent neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4584–4593.

Imed Zitouni, Jeffrey S Sorensen, and Ruhi Sarikaya. 2006. 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*, pages 577–584. Association for Computational Linguistics.