

Enhancing Arabic Diacritization with Word-Ending Specific Models

Stanford CS224N Custom Project

Hassan Fahmy
Department of Computer Science
Stanford University
hassanf@stanford.edu

Ahmed Shuaibi
Department of Computer Science
Stanford University
ashuaibi@stanford.edu

Grading Scheme: Option 3 - GRADED

Abstract

The written Arabic language consists of a set of short vowel markings known as diacritics that are utilized to dictate the meaning and pronunciation of words. Ordinarily, these diacritical markings are omitted in written Arabic for the sake of time and brevity. Models and systems have been created over the years to automatically add these diacritics to unmarked text. A major challenge of Arabic text diacritization is properly assigning markings to word endings, as these ending characters hold syntactic, semantic, and pragmatic meaning for the word and sentence as a whole. In this work, we present a word-level bidirectional LSTM model for the prediction of diacritics on word endings. We evaluate our model on Fadel et. al's filtered Arabic dataset [1] constructed from the freely available Tashkeela corpus [2]. We compare our approach's prediction accuracy with that of character-level approaches that indistinctly model word endings and aim to evaluate its effectiveness assessed solely on word ending diacritics. Although our model yields relatively poor performance, it is able to capture important aspects of word-ending diacritics that are rarely captured by the character-level baseline. This provides valuable insight for future iterations of a word-level model to capture word-ending diacritics.

1 Key Information

- External collaborators (if you have any): None.
- Mentor (custom project only): Christopher Manning.
- Sharing project: No.

2 Introduction

The written Arabic language consists of a set of short vowel markings known as diacritics that are utilized to dictate the meaning and pronunciation of words. There are 8 basic Arabic diacritics and characters can be marked with up to two diacritics of certain allowable combinations (Figure 1). Ordinarily, these diacritical markings are omitted in written Arabic for the sake of time and brevity. The procedure of adding these diacritics to unmarked text is known as diacritization. While inferring the diacritics of words from unmarked text is a straightforward task for fluent speakers, this is very difficult for those newly learning the Arabic language.

Diacritic Name	Diacritic Shape
None	س
Fatḥah	سَ
Kasrah	سِ
Ḍammah	سُ
Sukun	سْ
Shaddah	سّ
Fatḥatan	سَـ
Kasratan	سِـ
Ḍammatan	سُـ

Diacritic Name	Diacritic Shape
Shaddah & Fatḥah	سَّ
Shaddah & Kasrah	سِّ
Shaddah & Ḍammah	سُّ
Shaddah & Fatḥatan	سَّـ
Shaddah & Kasratan	سِّـ
Shaddah & Ḍammatan	سُّـ

Figure 1: All 8 basic diacritics are depicted on the letter س. (*seen*) on the left. On the right are some of the allowable combinations of diacritics on a single character. [3]

As such, systems that automatically perform diacritization are significantly useful for Arabic learners and non-fluent speakers. Furthermore, there are many cases of modern Arabic texts in which identifying the correct diacritization is difficult even for fluent speakers. Very minute diacritical differences can contribute to significant phonetic, syntactic, and semantic changes in a word and its surrounding sentence. Consider the following individual words with slightly perturbed diacritics:

حَسَنُ حَسَنِ

LeftPronounced Hassanon and means the *name Hassan* when used as a subject in a sentence. [Right] Pronounced Hassanin and means the *name Hassan* when used as an indirect object in a sentence.

حُسِنَ حَسَّنَ

LeftPronounced hossina and means *to be improved*. [Right] Pronounced hassana and means *to improve*.

This therefore acts as a barrier to full comprehension of written Arabic that can only be overcome with proper diacritization. Automatic diacritization system is necessary to aid fluent and non-fluent speakers in written Arabic pronunciation and comprehension.

One challenging aspect of Arabic text diacritization is the prediction of the markings of word endings. The diacritics on the end of a words in Arabic hold information on their parts of speech and pronunciations. Unlike diacritics within a word, word-ending diacritics often change depending on the word level context in the sentence. As such, character level diacritization prediction models do not optimally perform on word-ending diacritic prediction. While the task of Arabic diacritization is typically approached at the character-level, we reason that the sentence-level approach is more suitable for the case of word endings.

3 Related Work

Several efforts have been made to diacritize Arabic text automatically, spanning from simple rule-based approaches to neural networks. While some approaches factor in linguistic features of the language in order to make daicritic predictions, others employ techniques without additional domain specific language knowledge. One very popular rule-based approach is contained within MADAMIRA [4], an extensive morphological analysis tool for Arabic . This tool was created to tackle NLP tasks in Arabic, including morphological disambiguation, POS tagging, tokenization, stemming, and diacritization. It tackles diacritization through quantifying and utilizing morphological characteristics of letters, words, and their underlying sentences as features in a rule-based manner for the prediction of letter diacritics. While MADAMIRA outperforms many other approaches in tasks such as POS tagging and morphological diambiguation, it has been shown to perform suboptimally on diacritization, as we will quantify in our results.

Beyond rule-based approaches come approaches that are more statistical in nature. Particularly, the work of Sarikaya et. al [5] and El-Shaefi et. al [6] approaches the task with a maximum entropy hidden markov model that attempts to predict the hidden states encapsulating the diacritics. Sarikaya et. al perform a simple morphological analysis of the input words to vowelize them (or in a distinct attempt split them into graphemes) to use as observed states prior to passing them into the model. Relative to MADAMIRA, this approach still takes advantage of a morphological analysis of the input and thereby couples linguistic features of the text to the task of diacritization.

Several neural network approaches that did not need linguistic features of the input have become more prevalent recently. Circumventing a preliminary morphological analysis through the utilization of powerful neural network models is appealing especially with the prevalence of increasing amounts of diacritized Arabic text. Among these is the work of Belinkov and Glass [?] in which they utilize multiple forms of stacked bidirectional character level LSTMs with a final softmax output layer to predict the diacritic class. The model was trained on the Arabic Treebank [7], comprised of 2.6 million words obtained from various web blogs. This approach yielded results near state of the art maximum entropy approaches in diacritization prediction. Finally, Fadel et. al [1] devise several feed forward and recurrent neural network (LSTM) models and train upon a much larger uniquely compiled corpus comprised of over 24.5 million total words pulled from ancient Arabic texts and modern articles. They uniquely utilize block-normalized gradients to accelerate the process and improve results with the LSTM model. Their attempts outperform Belinkov and Glass’s work while additionally outperforming all openly available rule and entropy based approaches on their dataset.

4 Approach

Among existing approaches for Arabic text diacritization, there does not exist one that aims to model the task of word-ending diacritic prediction separately. Instead, results in this field simply report accuracy rates with and without accounting for word-ending diacritics. Starting with Fadel et. al’s model as a baseline, tackle the more challenging task of word-ending diacritization separately. we devise a word-level bidirectional LSTM model to predict the diacritics of word endings. We train on unmarked Arabic text and predict the diacritics of the word ending for each word. We model our framework based on Fadel’s character-level LSTM for diacritization and thereby use a 256 unit LSTM, a 512 unit RELU, and 19 unit Softmax and the architecture is depicted below.

For our model, we obtain the input word embeddings from FastText with 100 dimensions to fit the memory restraints of the VM. The output of the softmax presents the probability the word ending takes one of the 19 possible diacritic combinations. Beyond the diacritic combinations enumerated above, this also includes 4 special tokens: <PAD>, <UNK>, <SOS>, <EOS>. These represent padding, unknown, start of sequence, and of sequence respectively. As a baseline, we use Fadel et. al’s character-level LSTM, with model architecture depicted below.

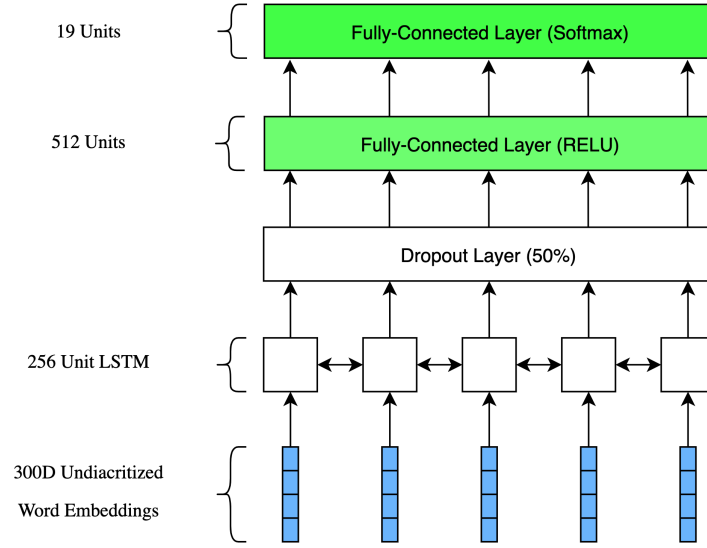


Figure 2: Our word-level LSTM for word ending diacritization

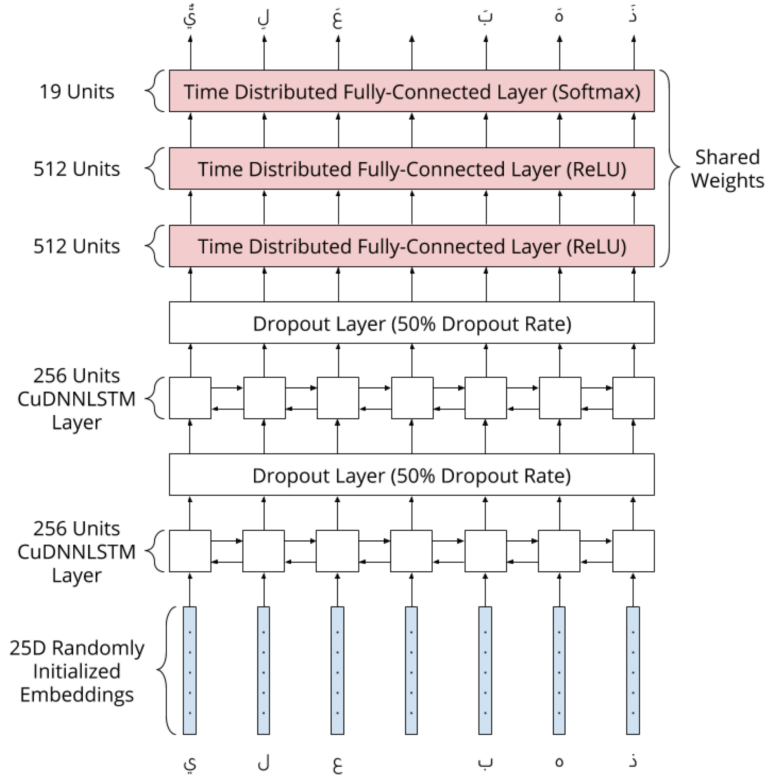


Figure 3: Fadel et. al's character-level LSTM for Arabic text diacritization. Note the utilization of stacked LSTMs and RELUs

5 Experiments

5.1 Data

We will be using the benchmark dataset created by Fadel et. al [8] for this task. This benchmark dataset was created through a subsampling and filtration of the largest fully diacritized Arabic dataset, *Tashkeela*, composed of 75 million words pulled from ancient Arabic texts, the Quran, and over 250 modern articles. Fadel et. al pinpointed and removed several irregularities in the dataset that were artifacts of the data extraction process. Simple dataset statistics are presented below:

Statistic	
Word Count	24.5M
Line Count	583K
Average Chars/Word	3.97
Average Words/Line	42.097
0 Diacritic Chars (%)	17.79
1 Diacritic Chars (%)	77.16
2 Diacritic Chars (%)	5.03

Figure 4: Simple statistics of Fadel et. al’s benchmark dataset. [1]

Since our task focused on word-ending diacritics, we needed to further process this dataset to make it suitable. As such, we stripped off all diacritics except those on word endings. This preprocessing is depicted below on a sample line from the input. Note that some word-endings do not have diacritics, in which case they would stay the same in the preprocessing.

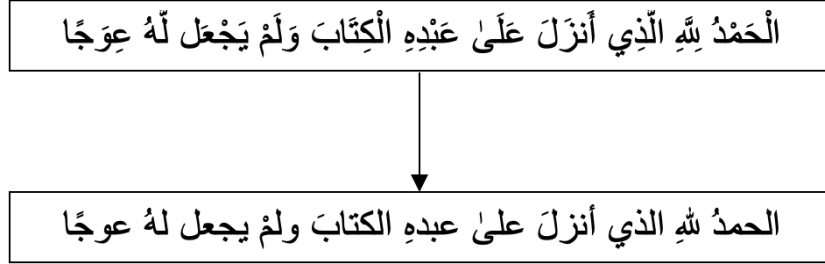


Figure 5: Preprocessing example Fadel et. al dataset to obtain Arabic text with only word-endings diacritized to use as model input

Thereafter, we performed an 80/10/10 split of this dataset into training, validation, and test sets. At evaluation time, we fully stripped the diacritics from word-endings and fed this into the model to predict word-ending diacritics.

5.2 Evaluation method

We use Diacritization Error Rate (DER) as our metric for evaluation. DER is a measure of the number of incorrectly diacritized letters among all the input letters. In our case, we are only concerned with the DER of word-endings. Diacritization error is binary and a prediction is considered incorrect if it does not belong to one of the 19 possible diacritic classes. As such, we do not consider diacritization to be partially correct for predicting the component of a diacritic class (ex. predicting Kasrah when belonging to the class Shaddah & Kasrah).

5.3 Experimental details

We trained the model for 50 epochs on Google Cloud using Nvidia K80 GPUs. The model was configured to utilize an Adam optimizer with a learning rate of 0.001, a β_1 of 0.9, and a β_2 of 0.999. Furthermore, we utilize batches of size 256 and categorical cross-entropy loss. Using this configuration, the baseline character-level LSTM model provided by Fadel et. al trained in 57 hours. Our newly created word-level LSTM for the task of word-ending diacritization trained in 43 hours. To yield a relatively accelerated training time, both runs utilized Block-Normalized Gradients applied to each batch. Block-Normalized Gradients (BNGs) is an approach introduced by Yu et. al [9] that normalizes the gradients within each batch in the training process.

5.4 Results

	Diacritization Error Rate
Character-Level LSTM (Baseline Model)	13.64%
Word-Level LSTM (Our Model)	81.63%

Figure 6: Diacritization Error Rate results of word-endings using our model against the baseline model.

These results are much worse than we anticipated. There are several insights this gives us about our approach. Most notably, while modeling word-ending diacritization through the use of word-level relationships and positions made sense in theory, it did not perform well in practice. This is due to a variety of reasons:

- Great diversity of consecutive word-ending diacritic classes relative to consecutive character diacritic classes. Consider the Arabic word: *يَصَدَّعْنَ*. This word has two consecutive characters that have the diacritic class of *Shaddah + Fathah*. However, this construction is extremely rare in the Arabic language. There are a few occurrences of this case in the entire Quran (comprised of roughly 75,000 words). As such, the near impossibility of these diacritic combinations is learned in a character-level model. Conversely, the order of word-ending diacritics is not well defined and there is a greater level of diversity in what is allowable. With regards to word order topology, Arabic is typically a Verb-Subject-Object (VSO) order language. However, Object-Subject-Verb (OSV) and Object-Verb-Subject (OVS) order are allowable in Arabic and occur with notable frequency in ancient Arabic texts (which the training corpus is largely composed of). As such, our model could not sufficiently learn word-ending diacritics due to this inherent variety
- Weaker model. Fadel et. al’s model utilized stacked LSTMs and stacked RELU layers with the reasoning that it worked sufficiently for their use case. We created our model initially using one LSTM and one RELU layer, notably weaker than the baseline character-level counterpart. As such, experimenting with more complex stacked LSTM models may yield better results.

Overall, these reasons in addition to those expressed in the Analysis section explain the poor performance and grants us an understanding oh how to better model word-ending diacritics in Arabic in a manner that is not solely word-level.

6 Analysis

As explained above, our model encountered several shortcomings due to the nature of Arabic sentence construction relative to inherent diversity of component word-ending diacritics. However, while the character-level model far outperforms our word-level model in capturing word-ending diacritics, there several cases in which our model captures a word-ending diacritic that is not predicted by the

character-level model. Namely, consider the example below with full diacritics, paying particular attention to the final word.

قَالَ لِي مَالِكُ

Our model produces identical output to the fully diacritized form while the baseline model produces the output:

قَالَ لِي مَالِكُ

In Arabic, subjects of a sentence possess the diacritic class *Dammah* or *Shaddah + Dammah*. Additionally, if a proper noun is the subject of a sentence, it possesses the class *Shaddah + Dammah* over *Dammah* an overwhelming majority of the time. Our word-level model is able to capture this information with regards to the proper nouns as subjects of sentences and thus utilizes the correct class. The character-level baseline model very frequently incorrectly diacritizes proper noun subjects just like with this case and many others. Since it works at the character level, it is unable to capture the proper noun property of words and thereby yields many errors in these cases.

7 Conclusion & Future Work

This project explores the utilization of a word-level model to predict the diacritic classes of word-endings in Arabic. Our model yields very poor performance on the task as a whole. This poor performance yields valuable insight for the task of word-ending diacritic prediction as a whole. Firstly, it prompts us to consider the word order topology of the Arabic language, especially in ancient Arabic texts. Since word-ending diacritics is closely tied to the part of speech of a word, the diversity in allowable word order topology interfered significantly with word-ending diacritic prediction. Furthermore, modeling this task at the character level allows us to learn most probable and nearly impossible consecutive diacritic combinations. At the word-level, there is less structure in allowable word-ending diacritic combinations. As such, prediction at the word-level is less effective overall. One benefit of our work was its capacity to capture the diacritic class of particular words. Most notably, it nearly always correctly captured the diacritic class of proper nouns when used as subjects in a sentence. This capacity is lost at the character level.

We aim to take several avenues to improve this model. First, we aim to experiment with more complex models through stacked LSTMs and evaluate the relative improvement in performance. Second, we aim to use higher dimensional word embeddings to better capture underlying word property information. To tackle the variety in word topology order, we aim to develop an embedding representation of a word that contains all its diacritics except the final diacritic. Doing so will help to better capture the role of a word in a sentence in addition to capture the specific word topology order of the sentence. We plan to do this by enumerating all diacritic class combinations for words and learning representations of all these diacritic class combinations. We will thereby use these improved representations to train the word-level word-ending diacritic prediction model.

References

- [1] Ali Fadel, Ibraheem Tuffaha, Bara' Al-Jawarneh, and Mahmoud Al-Ayyoub. Neural arabic text diacritization: State of the art results and a novel approach for machine translation. *arXiv preprint arXiv:1911.03531*, 2019.
- [2] Taha Zerrouki and Amar Balla. Tashkeela: Novel corpus of arabic vocalized texts, data for auto-diacritization systems. *Data in Brief*, 11:147 – 151, 2017.
- [3] Mark A. Foster. Arabic diacritics.
- [4] Arfath Pasha, Mohamed Al-Badrashiny, Mona Diab, Ahmed El Kholy, Ramy Eskander, Nizar Habash, Manoj Pooleery, Owen Rambow, and Ryan Roth. 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, May 2014. European Language Resources Association (ELRA).

- [5] Ruhi Sarikaya, Ossama Emam, Imed Zitouni, and Yuqing Gao. Maximum entropy modeling for diacritization of arabic text. In *Ninth International Conference on Spoken Language Processing*, 2006.
- [6] Moustafa Elshafei, Husni Al-Muhtaseb, Mansour Al-Ghamdi, et al. Machine generation of arabic diacritical marks. *MLMTA*, 2006:128–133, 2006.
- [7] Mohamed Maamouri, Ann Bies, Tim Buckwalter, and Wigdan Mekki. The penn arabic treebank: Building a large-scale annotated arabic corpus. In *NEMLAR conference on Arabic language resources and tools*, volume 27, pages 466–467. Cairo, 2004.
- [8] Ali Fadel, Ibraheem Tuffaha, Mahmoud Al-Ayyoub, et al. Arabic text diacritization using deep neural networks. In *2019 2nd International Conference on Computer Applications & Information Security (ICCAIS)*, pages 1–7. IEEE, 2019.
- [9] Adams Wei Yu, Lei Huang, Qihang Lin, Ruslan Salakhutdinov, and Jaime Carbonell. Block-normalized gradient method: An empirical study for training deep neural network. *arXiv preprint arXiv:1707.04822*, 2017.