Arabic Diacritization Using Morphologically Informed Character-Level Model

Muhammad Elmallah*, Mahmoud Reda*, Kareem Darwish*, Abdelrahman El-Sheikh*, Ashraf Elneima*, Murtadha Aljubran†, Nouf Alsaeed†, Reem Mohammed†, Mohamed Al-Badrashiny*

*aiXplain Inc., San Jose, CA, USA
{muhammad.elmallah,mahmoud.reda, kareem.darwish,ashraf.hatim,mohamed}@aixplain.com

†SDAIA, Riyadh, Saudi Arabia
{maaljubran,nalsaeed, rmohammed}@sdaia.gov.sa

Abstract

Arabic diacritic recovery i.e. diacritization is necessary for proper vocalization and an enabler for downstream applications such as language learning and text to speech. Diacritics come in two varieties, namely: core-word diacritics and case endings. In this paper we introduce a highly effective morphologically informed character-level model that can recover both types of diacritics simultaneously. The model uses a Recurrent Neural Network (RNN) based architecture that takes in text as a sequence of characters, with markers for morphological segmentation, and outputs a sequence of diacritics. We also introduce a character-based morphological segmentation model that we train for Modern Standard Arabic (MSA) and dialectal Arabic. We demonstrate the efficacy of our diacritization model on Classical Arabic, MSA, and two dialectal (Moroccan and Tunisian) texts. We achieve the lowest reported word-level diacritization error rate for MSA (3.4%), match the best results for Classical Arabic (5.4%), and report competitive results for dialectal Arabic.

Keywords: Arabic diacritization, Sequence labeling, Character-level models

1. Introduction

Arabic words are composed of letters, either consonants or long vowels, along with diacritics that mostly follow consonants and long vowels, when they behave like consonants, to specify how they are vocalized. For example, the letter sequence كتب (ktb[1]) can be diacritized as كَتَبَ (kataba – "he wrote"), كُتِبَ (kutiba – "it was written"), كُتُب (kutub – "books"), etc. Diacritics are typically omitted in Arabic writings, and the readers need to guess the proper diacritics as they are reading. Downstream tasks such as text-to-speech and Arabic language learning may necessitate the automatic recovery of diacritics to enable proper vocalization. Other tasks such as machine translation may also benefit from diacritization as diacritization helps disambiguate the words in context. There are two types of diacritics, namely core-word diacritics, which disambiguate a word in context, and case-endings that commonly appear on the last letter of word to typically indicate syntactic role. While case-endings are important for Modern Standard Arabic (MSA) and Classical Arabic (CA), Arabic dialects typically use sukun, which indicates the absence of a vowel, for case-ending. As the names suggest, MSA is the contemporary formal Arabic that is commonly used now in official communication, CA appears in classical literary and religious texts, and dialectal Arabic is the spoken day-to-day language with different varieties in different locales. Further, the rich morphology of Arabic, which allows for the concatenation of prefixes and suffixes to words, directly affects the actual start and end of words and consequently diacritization. For example, the word بسكوتك (bskwtk) can be segmented as بـ+سكوت+ك (b+skwtk+k – "your silence") or as بسكوت+ك" (bskwt+k – "your biscuits"), where the diacritized forms would be بِسُكُوتِك (bisukuwtik) and بَسْكُوتك (basokuwtkThe diacritics after ت (t) and ك (k) are omitted as they depend on the context.) respectively.

Diacritics are commonly omitted in written text, and readers have to infer them as they read. Omission causes a challenge for language learners and for downstream tasks such as text-to-speech. Further, the inclusion of diacritics may increase the effectiveness of downstream tasks such as part of speech tagging and machine translation. The most common diacritization approaches either work: at word and/or clitic level typically decoupling case ending and core-word diacritics (Darwish et al., 2017; Hamed and Zesch, 2017); or at character level without discrimination between core-word diacritics and case endings (AlKhamissi et al., 2020; Fadel et al., 2019; Mubarak et al., 2019b). In this work, we build on top of the latter approach and introduce a recurrent neural network (RNN) character-based diacritization model that incorporates morphological segmentation information. The proposed model differs from prior character-level models in multiple aspects. Unlike the sequence to sequence model proposed by Mubarak et al.

[1] Buckwalter encoding is used throughout the paper.

LREC-COLING 2024, pages 1446–1454
20-25 May, 2024.

(2019b), which has a tendency to hallucinate due to the fact that input and output do not necessarily correspond one-to-one, our proposed model does not hallucinate while being much faster with lower error rate. Further, although prior work incorporated word level information (AlKhamissi et al., 2020), none incorporated any morphological information. We show that the incorporation of morphological segmentation significantly improves diacritization accuracy with drops in word-level diacritization error rate ranging between 26% and 65% for different varieties of Arabic. To incorporate morphological segmentation, we introduce a character-level word segmentation model that does not rely on a predefined list of prefixes, suffixes, or stems, which makes it amenable for MSA as well as dialectal Arabic varieties, which lack widely accepted spelling conventions. The model uses a combination of RNN layers with a Conditional Random Fields (CRF) layer. The segmenter achieves an accuracy of 98.4% for MSA (using noisy training data) and 93.1% on dialectal Arabic.

The contribution of this paper is as follows:

- We introduce an RNN-based character-level Arabic diacritization model that incorporates morphological segmentation information.
- We introduce a new character-based word segmentation model that requires no prior linguistic features.
- We show that our model that incorporates word segmentation information beats all prior models, both word and character-based, on MSA, yields competitive results on classical Arabic, and achieves competitive results for dialectal Arabic.

2. Related Work

There has been much work on Arabic diacritic recovery (a.k.a. Arabic diacritization) dating back to the 1990's, with most of the early works being rule-based and proprietary, such as the diacritizer developed by `sakhr.com`. Rule-based diacritization required extensive morphological and syntactic rules (El-Imam, 2004; El-Sadany and Hashish, 1988). The rise of statistical and machine learning methods drove the work into two different directions, namely the development of diacritized corpora and utilization of different learning techniques.

2.1. Diacritization Datasets:

One of the earliest corpora that was utilized for diacritization was the Holy Quran (Gal, 2002). However, the characteristics of the Quran corpus could not be generalized to other genres and other varieties of Arabic, such as MSA. Later, the Linguistics Data Consortium developed the Arabic Penn Treebank (ATB) that was composed of hundreds of thousands of MSA tokens, mostly from the news domain, and included diacritization (Maamouri et al., 2004). The availability of the corpus ushered a wave of significant works on MSA processing including diacritization (Habash and Rambow, 2007; Nelken and Shieber, 2005; Zitouni et al., 2006). However, ATB had two shortcomings, namely it was relatively small and diacritization was often inconsistent (Darwish et al., 2017), with frequently missing diacritics. These two shortcomings stinted the generalization of models that used ATB for training with much higher levels of diacritization errors. There are much larger training datasets containing millions of words (Mubarak et al., 2019b; Noaman et al., 2018), which have led to much improved diacritization results. Unfortunately such datasets are proprietary, limiting the ability to properly compare different diacritization techniques. For Classical Arabic, there is a plethora of Classical Arabic books on the internet that are at least partially diacritized. This led to the development of large datasets such as Tashkeela Corpus (Zerrouki and Balla, 2017), which is composed of 75 million words. Though quite large, a smaller portion of the corpus is favorable for training diacritizers due to frequent partial diacritization (AlKhamissi et al., 2020; Fadel et al., 2019). We use a subset of Tashkeela in our work that is much smaller. For dialectal Arabic, there are some diacritized corpora for some dialects such as Tunisian, Moroccan, Egyptian, and Emirati (Alabbasi et al., 2022; Habash et al., 2012; Mubarak et al., 2019b). However, most are either genre specific (e.g. Bible translations into Tunisian and Moroccan) or too small (e.g. Emirati), limiting the development of broad coverage diacritizers. In this work, we use a proprietary MSA dataset composed of more than 7 million tokens, a subset of Tashkeela (Fadel et al., 2019), and the Tunisian and Moroccan Bible corpora. The availability of large publicly available diacritized corpora for diacritization continues to be an issue, particularly as we show in this paper that using larger corpora yields improved diacritization results.

2.2. Diacritization Techniques:

A variety of statistical and machine learning methods have been used for Arabic diacritization. The earliest ML methods utilized a Hidden Markov Model (HMM) (Gal, 2002) or a Finite State Transducer (FST) (Nelken and Shieber, 2005). Such methods relied on an inventory of diacritized forms for different words, which can be obtained from training data or a morphological analyzer. Later work started integrating morphological and syntactic features, such as prefix/suffix segmentation, part-of-speech tags, stem templates, gender, number, etc., to engineer features to improve diacritization (Darwish et al., 2017; Habash and Rambow, 2007; Obeid et al., 2020; Pasha et al., 2014; Roth et al., 2008; Zitouni et al., 2006). Later works replaced classical ML methods for deep learning ones, such as Recurrent Neural Networks (Abandah et al., 2015; Belinkov and Glass, 2015; Darwish et al., 2021; Rashwan et al., 2015). Though such methods, particularly deep learning ones, significantly improved diacritization results, they require the availability of morphological and syntactic analyzers, which may not be readily available (particularly for dialectal Arabic), and they often treated core-word diacritics separately from case-endings. Note that recovering case-endings is typically required for MSA and Classical Arabic, while dialects generally use sukun as a case-endings for the majority of words. However, one issue that greatly complicates dialectal diacritization is the ubiquity of sub-dialects. Thus, many dialectal words have diacritized forms in sub-dialects that differ from region to region and city to city (Alabbasi et al., 2022). A

recent trend has focused on using character-level models that require no morphological or syntactic analysis (AlKhamissi et al., 2020; Fadel et al., 2019; Mubarak et al., 2019b). For example, Mubarak et al. (2019b) proposed a sequence-to-sequence model with either an RNN or a transformer model to "translate" an undiacritized character sequence to a diacritized character sequence. Though, their method improved upon the state-of-the-art, it suffered from a serious hallucination problem, where the decoder often generated output that did not completely match the input. They tried to alleviate this problem using overlapping sliding windows, which were computationally expensive and not completely reliable (Mubarak et al., 2019b). Fadel et al. (2019) used a character-level RNN, which guaranteed that the input and output completely match. AlKhamissi et al. (2020) utilized a hierarchical model that incorporates word-level as well as character-level information, leading to state-of-the-art results on the Tashkeela corpus. In this paper, we improve upon character-level models by using an RNN model that makes use of morphological segmentation. We show that the inclusion of such segmentation achieves significant improvements in diacritization accuracy, while eliminating hallucinations associated with sequence-to-sequence models. For more detailed background on Arabic diacritic recovery, we refer the readers to multiple comprehensive surveys of state-of-the-art methods (Al-Ayyoub et al., 2018; Almanea, 2021; Azmi and Almajed, 2015; Hamed and Zesch, 2017).

2.3. Arabic Morphological Segmentation:

Arabic words are typically derived from a closed set of roots that are fit into stem templates and may accept prefixes and suffixes to construct words (Abdelali et al., 2016; Obeid et al., 2020). Morphological segmentation involved segmenting words into their underlying clitics, which involves identifying the concatenation points between prefixes, suffixes, and the stem. For example, the word وكتاباتهم (wktAbAthm – and their writings) would be split into و+كتابات+هم (w+ktAb+At+hm). Such segmentation maybe performed using morphological analysis (Obeid et al., 2020) or using a classification model that attempts the most likely segmentation (Abdelali et al., 2016). Such models typically assume a closed set of prefixes and suffixes, which may be fine for MSA and Classical Arabic but rather problematic for dialectal Arabic. Arabic dialects lack commonly adopted spelling conventions and introduce affixes that do not exist in MSA and CA. Consider the dialectal word مالعبتش (mAlEbt$ – I/she did not play). Other common spelling variations include ملعبتش (mlEbt$) and ما لعبتش (mA lEbt$). Further, negation of the verb is done by introducing ما (mA) (or م (m)) as a prefix or independent token and the suffix ش ($). This negation construct does not exist in MSA or CA. Further, since dialectal Arabic is mostly used in non-formal communication, word concatenations, spelling mistakes, and repeated letters are ubiquitous. For example, the word شتسووي ($tswwy – what are you doing) is: actually of a fusion of the word إيش (>y$) and تسوي (tswy); إيش is contracted to ش ($); and تسووي (tswwy) has an additional character. All these properties complicate segmentation, where a segmentation model has to handle many forms of prefixes and suffixes along with misspellings and word concatenations.

In our work, we propose an RNN+CRF sequence labeling with no prior list of prefixes and suffixes that can also handle misspelled words and word concatenations. Such significantly simplifies the development of morphological segmentation.

3. Methodologies

3.1. Segmentation Model

Arabic words are segmented into their individual prefix(es), stem, and suffix components. For example, the word ومجلسي "wmjlsy" (gloss: and my sitting place) is composed of the prefix و "w" (and), stem مجلس "mjls" (sitting place), and the possessive pronoun ي "y" (my), the task of the tokenizer is to segment the word into و+مجلس+ي "W+mjls+y". Such morphological segmentation positively impacts a variety of downstream tasks such as machine translation, information retrieval, and, as we will show, Arabic diacritization also.

3.1.1. Data

For MSA, since there is no large publicly available alternative to the Arabic Penn Treebank (ATB) (Maamouri et al., 2004), we resorted to automatically build a training corpus for MSA by segmenting a large text corpus, composed of Arabic Wikipedia articles, using a state-of-the-art segmenter, namely Farasa (Abdelali et al., 2016). Using Farasa as a teacher model, the accuracy of our model would likely be capped by the accuracy of Farasa, namely 98.9% on the WikiNews corpus (Darwish and Mubarak, 2016), which we also used of evaluation. The breakdown of the train/test/validation sets is as follows:

Split	Size (tokens)
Train	1,500,000
Test (WikiNews)	18,301
Val	44,787

For dialectal segmentation, we used the set created by Darwish et al. (2018b) that includes 27,366 words across four dialects, namely Gulf, Egyptian, Levantine, and Maghrebi. Since there is significant overlap between the different dialects, we combined the data of all the dialects. We split the data into training, validation, and test sets. Further, since there is high overlap between dialects and MSA, we added the WikiNews dataset to the training set. Some example that illustrate the overlap include: the word الكتاب (AlktAb – the book), which exists in MSA and all dialects, and the word عشان (E$An – for), which is used in Egyptian, Levantine, and Gulf. Abdelali et al. (2020) show the overlap between dialects in detail. The splits were as follows:

Split	Size (tokens)
Train	39,970
Test	4,000
Val	1,697

3.1.2. Model and Results

We used the NCRFpp package, which is a neural sequence labeling package, to train the segmentation model (Yang and Zhang, 2018). We made the simplifying assumption that words don't depend on the context for proper segmentation. Though this assumption is not completely correct (see the example in the introduction), prior work has shown that this assumption holds for nearly 99% of the cases (Abdelali et al., 2016). We configured NCRFpp to have the following architecture:

- Input: characters (dim: 15)
- Character Embeddings Layer (dim: 50)
- 2 biLSTM layers (dim: 100)
- 4 biLSTM layers (dim: 400)
- CRF layer

We used ADAM optimizer and early stopping with patience equal to 5. For MSA, the trained model had an accuracy of 98.4%, which is 0.5% lower than the results achieved by Farasa. We suspect that our model would have achieved higher results had we used human annotated data (e.g. ATB). Our model has a distinct advantage over Farasa's model as it does not require a list of prefixes and suffixes and works directly at character level without feature engineering. We performed error analysis on 50 random errors from our segmentation model, and the errors we found were as follows:

- (20 errors) Mistakes in segmentation: فيلم (Ref: fylm – film; Guess: f+ylm)
- (17 errors) Named entities or foreign words: بوكاري (Ref: bwkAry – Bokary; Guess: b+wkAry)
- (10 errors) Ambiguous (reference & guess are both correct) باسم (Ref: b+Asm – in the name of; Guess: bAsm – Basem)
- (2 errors) Partial segmentation: واللاذقية (Ref: w+Al+l*qy+p – and Latakia; Guess: w+Al+l*qyp)
- (1 error) Error in reference: وكذلك (Ref: w+k+*lk – and also; Guess: w+k*lk)

For dialectal Arabic, the accuracy was 93.1%. Access to more training data would have likely yielded improved results. Again, we performed error analysis on 50 random errors, and they were as follows:

- (27 errors) Mistakes in segmentation: معكن (Ref: mE+kn – with you; Guess: mEkn)
- (7 errors) Partial segmentation: طعميني (Ref: TEm+y+ny – (you) feed me; Guess: TEmy+ny)
- (6 errors) Error in reference: فكرون (Ref: fkrwn – they think; Guess: fkr+wn)
- (6 errors) Named entities or foreign words: سمستر (Ref: smstr – semester; Guess: s+mstr)
- (4 errors) Ambiguous (reference & guess are both correct) مزكم (Ref: mzkm – stuffy nosed; Guess: mz+km – since when)

3.2. Diacritization Model

3.2.1. Data

We made use for 4 different dataset that cover Modern Standard Arabic, Classical Arabic, and two dialectal Arabic varieties, namely Moroccan and Tunisian. Table 1 shows the size of the different datasets.

MSA: For MSA we used a proprietary diacritized MSA corpus composed of 7.2 million tokens (234,845 sentences) and covers different subjects such as economy, religion, politics, society, etc. For testing, we used the WikiNews corpus, which is composed of 18,300 words (400 sentences).

Classical Arabic: For Classical Arabic, we used a subset of the Tashkeela Corpus (Zerrouki and Balla, 2017). The corpus is composed of 75 million diacritized tokens from 97 classical Arabic books that were scraped from the web. For proper comparison with prior work, we used the cleaned subset of Tashkeela along with the exact train/test/validation splits prepared by Fadel et al. (2019). The subset is composed of roughly 60k sentences containing 2.3 million tokens.

Dialectal Arabic We had two dialectal datasets composed of two fully diacritized dialectal translations of the Bible into two Maghrebi dialects, namely Moroccan[2] and Tunisian[3]. Each of them had 1,600 verses with 134,323 and 131,870 tokens for the Moroccan and Tunisian versions respectively. For proper comparison with prior work, we used 5 fold cross validation using the same splits of Mubarak et al. (2019a).

Data	Tashkeela	Moroccan	Tunisian	MSA
Train	2,458,113	116,400	114,037	4,157,656
Test	125,098	29,130	28,501	18,300
Val	119,958	12,933	12,671	18,017

Table 1: Number of tokens for each dataset (average is reported for the 5 folds for Moroccan and Tunisian)

3.3. Diacritization Model

For diacritization, we used the RNN model depicted in Figure 1 that has a character embeddings layer, followed by 3 biLSTM layers, 3 dense layers, and a softmax layer. To avoid over fitting, we used dropout and early stopping with patience 5. We experimented with a variety of other architectures including a transformer model and a CRF output layer instead of a softmax, but none of them led to improved results.

We experimented with two input types, namely with and without clitic segmentation. Given the sequence وقال الرجل (wqAl Alrjl – and the man said), the input S would be {w, +, q, A, l, _, A, l, +, r, j, l} and {w, q, A, l, _, A, l, r, j, l} with and without segmentation respectively,

[2] https://www.biblesociety.ma/
[3] https://www.bible.com/bible/1304/

	w/o Segmentation				w/ Segmentation			
	WER		CER		WER		CER	
	w/o CE	w/ CE	w/o CE	w/ CE	w/o CE	w/ CE	w/o CE	w/ CE
MSA	3.4	5.7	1.1	1.4	1.9	3.4	1.3	1.4
Classical	4.5	7.3	2.0	2.4	2.7	5.4	2.5	2.8
Moroccan	4.1	-	1.5	-	2.0	-	1.6	-
Tunisian	9.6	-	4.0	-	3.1	-	2.6	-

Table 2: Diacritization Results. Since dialects generally use sukun for case endings, there are no case ending results for dialects.

MSA System	WER
(Belinkov and Glass, 2015)	30.5
(Pasha et al., 2014)	19.0
(Obeid et al., 2020)	15.6
(Rashwan et al., 2015)	16.0
(Darwish et al., 2017)	12.8
(Mubarak et al., 2019b)	4.5
Ours	**3.4**
Classical Arabic	**WER**
(Fadel et al., 2019)	11.2
(AlKhamissi et al., 2020) (d2/d3)	5.5/**5.3**
Ours	5.4
Moroccan	**WER**
(Darwish et al., 2018a)	2.9
(Mubarak et al., 2019b)	**1.4**
Ours	2.0
Tunisian	**WER**
(Darwish et al., 2018a)	3.8
(Mubarak et al., 2019b)	**2.5**
Ours	3.1

Table 3: Comparing our results to other systems – WER w/ case-ending. Note that dialects generally use sukun for case ending.

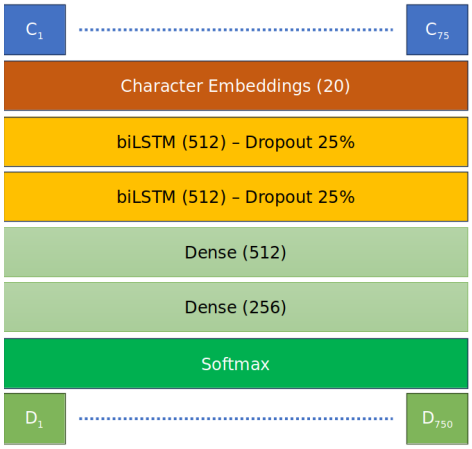

Figure 1: Diacritization model

where + indicates the morphological split between clitics that make a word and _ indicates a space. The output sequence would correspond to the appropriate diacritics per character, with + and _ getting a null diacritic.

3.4. Experiments and Results

We used two evaluation metrics, namely Word Error Rate (WER), which is the percentage of words with one or more diacritization errors, and Character Error Rate (CER), which is the percentage of characters with incorrect diacritics, with and without Case Endings (CE). Note that when using word segmentation, all segments are concatenated prior to computing WER, so it is computed at word level (exactly as in the case without segmentation). Similarly, the "+" symbol is omitted when computing CER. We relaxed our evaluation scripts to account for varying diacritization standards such as the emission of short vowels followed by matching long vowels (e.g. a (fatha) followed by A (alef)) and common omission of sukun (Darwish et al., 2017). Table 2 reports on all our experiments for the variations of Arabic, and Table 3 compares the results of our best model (WER with case-ending) that utilizes segmentation with the the results of prior works. Multiple notable observations can be drawn from the results, namely:

- Incorporating morphological segmentation information led to improvements across all varieties of Arabic (relative drop in WER: MSA: 40%; Classical: 26%; Moroccan: 51%; and Tunisian: 68%). This suggests that incorporating additional morphological or syntactic information, such word-level information (AlKhamissi et al., 2020), may further improve results.
- Our results on MSA reduced WER by 24% (relative) compared to the best reported results on the same test set by Mubarak et al. (2019a), which was trained on a corpus of roughly 4 million words and had a tendency to hallucinate. Our results are the best reported results on the WikiNews test set. We suspect that part of the improvement is due to using more training data. This highlights the importance of having large publicly available diacritized corpora, as the only publicly available one, namely ATB (Maamouri et al., 2004) is not sufficient.
- Our Classical Arabic results are at par with the best reported results (AlKhamissi et al., 2020), which were based on a model that incorporates both character and word information. We suspect that further incorporation of word-level information into our model would lead to further improvements.
- Though our results lag behind those of Mubarak et al. (2019b) for Moroccan and Tunisian, our system does

Error Type	Count	Example
Named entity/foreign word	12	Ref: أَنْدُرُويِدْ (>anodoruwyido – Android) Guess: أَنْدِرُوِيدْ (>anodiruwyido)
Ambiguous attachment	10	Ref: أَكَّدَ حَارِسُ مَرْمَى الْمُنْتَخَبِ الْبُرْتُغَالِيِّ (>ak∼ada HaArisu maromaY Alomunotaxabi AloburotugaAliyi– the goalkeeper of the Portuguese team indicated); Guess: أَكَّدَ حَارِسُ مَرْمَى الْمُنْتَخَبِ الْبُرْتُغَالِيُّ (>ak∼ada HaArisu maromaY Alomunotaxabi AloburotugaAliyu– the Poruguese goalkeeper of the team indicated)
Wrong part-of-speech	8	Ref: وَتَعَرَّضَ (wataEar∼aDa – and he was exposed to); Guess: وَتَعَرُّضِ (wataEar∼uDi – and exposure)
Active vs. passive voice	6	Ref: تُلْغَى مَرَاسِمُ (tulogaY maraAsimu – formalities to canceled); Guess: تَلْغِي مَرَاسِمَ (talogiy maraAsima – she cancels formalities)
Both correct	4	Ref: الْأَسْبَانِيُّ (Alo>asobaniy∼u – the Spanish); Guess: الْإِسْبَانِيُّ (Alo>isobaniy∼u)
Unexplained errors	5	Ref: فَعَالِيَّاتِهِ (faE∼aAliy∼aAtihi – its functions); فَعَالِيَّاتِهِ (faEaAliy∼aAtihi)
Tanween (added/missing)	3	Ref: بِشَكْلٍ (bi$akoli – in the form); بِشَكْلِ (bi$akolK)
Number (e.g. plural vs. dual)	2	Ref: الطَّالِبَيْنِ (AlT∼aAlibayoni – the two students); Guess: الطَّالِبِينَ (AlT∼aAlibiyna – those who are asking)

Table 4: Analysis of 50 random diacritization errors for MSA.

Error Type	Count	Example
Unexplained error	18	Ref: لِحَقِّهِ(liHaq∼ihi – to his right); Guess: لِحِقَّهُ(liHiq∼ihu)
Disambiguation error	11	Ref: اللُّبْسُ (All∼ubosu – the cloth); Guess: اللَّبْسُ (All∼abosu – the ambiguity)
Ambiguous attachment	11	Ref: أَرْسَلَ لَهُمْ عُثْمَانَ (>arosala lahumo EuvomaAna – he sent them Othman); Guess: أَرْسَلَ لَهُمْ عُثْمَانُ (>arosala lahumo EuvomaAnu – Othman sent to them)
active vs. passive voice	10	Ref: فَيَحْكُمُ (fayaHokumu – so he rules); Guess: فَيُحْكَمُ (fayuHokamu – so it is tightly closed)

Table 5: Analysis of 50 random diacritization errors for CA.

Error Type	Count	Example
Shadda+sukun vs. sukun or fatha	20	Ref: وْبْضَلّْ (woboDol∼o – and it stays); Guess: وْبْضَلْ (woboDolo) & Ref: وْحْفَرْ (woHofaro – and he dug); Guess: وْحْفْرْ (woHof∼oro)
Sukun vs. other diacritics	17	Ref: وَلَا (walaA – and not); Guess: وْلَا (wolaA)
Wrong diacritization	13	Ref: حُريَّة (Hury∼ap – freedom); Guess: حَرّيَّة (Har∼y∼ap)

Table 6: Analysis of 50 random diacritization errors for Moroccan.

Error Type	Count	Example
Wrong diacritization	27	Ref: البْسوا (AlboswA – (you pl.) wear); Guess: البُسُوا (AlbusowA)
Sukun vs. other diacritics	17	Ref: وِقْعَدْ (wiqoEado – and he sat); Guess: وْقْعَدْ (woqoEado)
Shadda+sukun vs. sukun	3	Ref: نْحِبْكُمْش (noHibokumo$o – we don't like you); Guess: نْحِبْكُمْش (noHib∼okumo$o)
Both correct	3	Ref: نِحْفِرْ (niHofiro – we dig); Guess: نَحْفِرْ (naHofiro)

Table 7: Analysis of 50 random diacritization errors for Tunisian.

not hallucinate and is computationally more efficient, making it more practical to deploy.

For error analysis, we extracted 50 random errors from each of our best systems for MSA, CA, Moroccan, and Tunisian. As shown in Table 3.2.1, the most common errors for MSA were mainly due to named entities (or foreign words), ambiguous attachment (or long distance attachment), and the use of active versus passive verbs. A small number of errors can be attributed to producing a completely wrong diacritized forms (Unexplained errors). For classical Arabic (Table 5) on the other hand, the most common errors were due to completely wrong diacritized forms. Further, a large number of errors can be attributed to incorrect disambiguation of a word in context, where the diacritizer picked the wrong diacritized forms. Like for MSA, ambiguous attachment and active versus passive voice were also common problems.

For Moroccan and Tunisian, the types of errors were very different from MSA and classical Arabic, as there were no errors related to ambiguous attachment or disambiguation in context. For Moroccan (Table 6), the vast majority of errors were due to substitution of sukun with shadda+sukun (or vice versa) and replacing sukun with another diacritic (or vice versa). The same problems were also observed for Tunisian (Table 7) but to a lesser degree. Overall, we observed significantly different distributions of error types for the different varieties of Arabic, with greater similarity between MSA and CA on one hand and Moroccan and Tunisian on the other. For example, MSA and CA exhibited many more errors relating to context as compared to dialectal datasets.

4. Conclusion

In this paper, we presented a character-level Arabic diacritization model that incorporates morphological segmentation information. We show that the inclusion of morphological segmentation significantly reduces diacritization word error rate for MSA, Classical Arabic, and dialectal by up to 68% (relative). Further, our MSA model achieves the best (lowest) word error rate (3.4%) on the WikiNews dataset, which is 24% lower than the previous state-of-the-art results (Mubarak et al., 2019b). This may hint that further incorporation of morphological and syntactic features may further improve character-level diacritization models. We plan to further investigate this direction for future work. Our work also highlights the importance of a community effort to develop a large publicly available corpus for Arabic diacritization. Improvements in WER when comparing models trained on ATB compared to those trained on more data is dramatic, and our results suggest that additional data continues to lead to lower WER. Futher, the paper introduced an RNN-based character-level word segmentation model that is capable of handing MSA, CA, and dialectal Arabic without the need for a preset list of prefixes and suffixes. This is particularly significant for dialectal Arabic varieties, where due to the lack of commonly accepted spelling conventions have arbitrarily large sets of prefixes and suffixes and words are often concatenated.

For future work, we plan to pursue three directions, namely: investigating the inclusion of additional morphological and syntactic features within our character-level model; performing multi-task learning where the model jointly performs segmentation and diacritization; and testing our dialectal diacritizers on social media text to ascertain their efficacy in naturally occurring dialectal texts. The texts of the Moroccan and Tunisian bibles are of a specific genre and are well curated, which may not match other sources of dialectal Moroccan and Tunisian texts.

5. 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 (IJDAR)*, 18:183–197.

Ahmed Abdelali, Kareem Darwish, Nadir Durrani, and Hamdy Mubarak. 2016. Farasa: A fast and furious segmenter for arabic. In *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: Demonstrations*, pages 11–16.

Ahmed Abdelali, Hamdy Mubarak, Younes Samih, Sabit Hassan, and Kareem Darwish. 2020. Arabic dialect identification in the wild. *arXiv preprint arXiv:2005.06557*.

Mahmoud Al-Ayyoub, Aya Nuseir, Kholoud Alsmearat, Yaser Jararweh, and Brij Gupta. 2018. Deep learning for arabic nlp: A survey. *Journal of computational science*, 26:522–531.

Nouf Alabbasi, Mohamed Al-Badrashiny, Maryam Aldahmani, Ahmed AlDhanhani, Abdullah Saleh Alhashmi, Fawaghy Ahmed Alhashmi, Khalid Al Hashemi, Rama Emad Alkhobbi, Shamma T Al Maazmi, Mohammed Ali Alyafeai, et al. 2022. Gulf arabic diacritization: Guidelines, initial dataset, and results. In *Proceedings of the The Seventh Arabic Natural Language Processing Workshop (WANLP)*, pages 356–360.

Badr AlKhamissi, Muhammad ElNokrashy, and Mohamed Gabr. 2020. Deep diacritization: Efficient hierarchical recurrence for improved Arabic diacritization. In *Proceedings of the Fifth Arabic Natural Language Processing Workshop*, pages 38–48, Barcelona, Spain (Online). Association for Computational Linguistics.

Manar M Almanea. 2021. Automatic methods and neural networks in arabic texts diacritization: a comprehensive survey. *IEEE Access*, 9:145012–145032.

Aqil M Azmi and Reham S Almajed. 2015. A survey of automatic arabic diacritization techniques. *Natural Language Engineering*, 21(3):477–495.

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.

Kareem Darwish, Ahmed Abdelali, Hamdy Mubarak, and Mohamed Eldesouki. 2021. Arabic diacritic recovery using a feature-rich bilstm model. *Transactions on Asian and Low-Resource Language Information Processing*, 20(2):1–18.

Kareem Darwish, Ahmed Abdelali, Hamdy Mubarak, Younes Samih, and Mohammed Attia. 2018a. Diacritization of moroccan and tunisian arabic dialects: A crf approach. *OSACT*, 3:62.

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.

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.

Kareem Darwish, Hamdy Mubarak, Ahmed Abdelali, Mohamed Eldesouki, Younes Samih, Randah Alharbi, Mohammed Attia, Walid Magdy, and Laura Kallmeyer. 2018b. Multi-dialect arabic pos tagging: A crf approach. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Paris, France. European Language Resources Association (ELRA).

Yousif A El-Imam. 2004. Phonetization of arabic: rules and algorithms. *Computer Speech & Language*, 18(4):339–373.

T El-Sadany and M Hashish. 1988. Semi-automatic vowelization of arabic verbs. In *10th NC Conference, Jeddah, Saudi Arabia*.

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

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*.

Nizar Habash, Ramy Eskander, and Abdelati Hawwari. 2012. A morphological analyzer for egyptian arabic. In *Proceedings of the twelfth meeting of the special interest group on computational morphology and phonology*, pages 1–9.

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.

Osaama Hamed and Torsten Zesch. 2017. A survey and comparative study of arabic diacritization tools. *Journal for Language Technology and Computational Linguistics*, 32(1):27–47.

Mohamed Maamouri, Ann Bies, Tim Buckwalter, and Wigdan Mekki. 2004. 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.

Hamdy Mubarak, Ahmed Abdelali, Kareem Darwish, Mohamed Eldesouki, Younes Samih, and Hassan Sajjad. 2019a. A system for diacritizing four varieties of arabic. 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): System Demonstrations*, pages 217–222.

Hamdy Mubarak, Ahmed Abdelali, Hassan Sajjad, Younes Samih, and Kareem Darwish. 2019b. 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 Papers)*, pages 2390–2395.

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

Hatem M Noaman, Shahenda S Sarhan, and MAA Rashwan. 2018. A hybrid approach for automatic morphological diacritizationof arabic text.

Ossama Obeid, Nasser Zalmout, Salam Khalifa, Dima Taji, Mai Oudah, Bashar Alhafni, Go Inoue, Fadhl Eryani, Alexander Erdmann, and Nizar Habash. 2020. Camel tools: An open source python toolkit for arabic natural language processing. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 7022–7032.

Arfath Pasha, Mohamed Al-Badrashiny, Mona T 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 *Lrec*, volume 14, pages 1094–1101.

Mohsen AA 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/ACM Transactions on Audio, Speech, And Language Processing*, 23(3):505–516.

Ryan Roth, Owen Rambow, Nizar Habash, Mona Diab, and Cynthia Rudin. 2008. Arabic morphological tagging, diacritization, and lemmatization using lexeme models and feature ranking. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies: Short Papers*, pages 117–120.

Jie Yang and Yue Zhang. 2018. Ncrf++: An open-source neural sequence labeling toolkit. In *Proceedings of*

the 56th Annual Meeting of the Association for Computational Linguistics.

Taha Zerrouki and Amar Balla. 2017. Tashkeela: Novel corpus of arabic vocalized texts, data for auto-diacritization systems. *Data in brief*, 11:147–151.

Imed Zitouni, Jeffrey Sorensen, and Ruhi Sarikaya. 2006. Maximum entropy based restoration of arabic diacritics. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 577–584.