Arabic Diacritic Recovery Using a Feature-rich biLSTM Model

KAREEM DARWISH, AHMED ABDELALI, HAMDY MUBARAK, and MOHAMED ELDESOUKI, Qatar Computing Research Institute, Hamad Bin Khalifa University

Diacritics (short vowels) are typically omitted when writing Arabic text, and readers have to reintroduce them to correctly pronounce words. There are two types of Arabic diacritics: The first are core-word diacritics (CW), which specify the lexical selection, and the second are case endings (CE), which typically appear at the end of word stems and generally specify their syntactic roles. Recovering CEs is relatively harder than recovering core-word diacritics due to inter-word dependencies, which are often distant. In this article, we use feature-rich recurrent neural network model that use a variety of linguistic and surface-level features to recover both core word diacritics and case endings. Our model surpasses all previous state-of-the-art systems with a CW error rate (CWER) of 2.9% and a CE error rate (CEER) of 3.7% for Modern Standard Arabic (MSA) and CWER of 2.2% and CEER of 2.5% for Classical Arabic (CA). When combining diacritized word cores with case endings, the resultant word error rates are 6.0% and 4.3% for MSA and CA, respectively. This highlights the effectiveness of feature engineering for such deep neural models.

CCS Concepts: • **Computing methodologies → Natural language processing**;

Additional Key Words and Phrases: Arabic, diacritization, text tagging

ACM Reference format:
Kareem Darwish, Ahmed Abdelali, Hamdy Mubarak, and Mohamed Eldesouki. 2021. Arabic Diacritic Recovery Using a Feature-rich biLSTM Model. *ACM Trans. Asian Low-Resour. Lang. Inf. Process.* 20, 2, Article 33 (April 2021), 18 pages.
https://doi.org/10.1145/3434235

1 INTRODUCTION

Modern Standard Arabic (MSA) and Classical Arabic (CA) have two types of vowels, namely long vowels, which are explicitly written, and short vowels, a.k.a. diacritics, which are typically omitted in writing but are reintroduced by readers to properly pronounce words. Since diacritics disambiguate the senses of the words in context and their syntactic roles in sentences, automatic diacritic recovery is essential for applications such as text-to-speech and educational tools for language learners, who may not know how to properly verbalize words. Diacritics have two types, namely core-word (CW) diacritics, which are internal to words and specify lexical selection, and case-endings (CE), which appear on the last letter of word stems, typically specifying their

Authors' address: K. Darwish, A. Abdelali, H. Mubarak, and M. Eldesouki, Qatar Computing Research Institute, Hamad Bin Khalifa University, P. O. Box 5825, Doha, Qatar; emails: {kdarwish, aabdelali, hmubarak, mohamohamed}@hbku.edu.qa.

2375-4699/2021/04-ART33 $15.00
https://doi.org/10.1145/3434235

syntactic role. For example, the word "ktb"[1] (كتب) can have multiple diacritized forms such as "k**a**t**a**b" (كَتَب – meaning "he wrote") or "k**u**t**u**b" (كُتُب – "books"). While "katab" can only assume one CE, namely "fatHa" ("a"), "kutub" can accept the CEs: "damma" ("u") (nominal – ex. subject), "a" (accusative – ex. object), "kasra" ("i") (genitive – ex. PP predicate) or their nunations. There are 14 diacritic combinations. When used as CEs, they typically convey specific syntactic information, namely **fatHa** "a" for accusative nouns, past tense verbs, and subjunctive present verbs; **kasra** "i" for genitive nouns; **damma** "u" for nominative nouns and indicative present verbs; and **sukun** "o" for jussive present verbs and imperative verbs. FatHa, kasra, and damma can be preceded by **shadda** "~" for gemination (consonant doubling) and/or converted to **nunation** forms following some grammar rules. In addition, according to Arabic orthography and phonology, some words take a **virtual** (null) "#" marker when they end with certain characters (e.g., long vowels). This applies also to all non-Arabic words (e.g., punctuation, digits, Latin words, etc.). Generally, function words, adverbs, and foreign named entities (NEs) have set CEs (sukun, fatHa, or virtual).

Similarly to other Semitic languages, Arabic allows flexible Verb-Subject-Object as well as Verb-Object-Subject constructs [6]. Such flexibility creates inherent ambiguity, which is resolved by diacritics as in "r>Y Emr <brAhym" (رأى عمر إبراهيم Omar saw Ibrahim/Ibrahim saw Omar). In the absence of diacritics it is not clear who saw whom. Similarly, in the sub-sentence "kAn Alm&tmr **AltAsE**" (كان المؤتمر التاسع), if the last word, is a predicate of the verb "kAn," then the sentence would mean "this conference was the ninth" and would receive a fatHa (a) as a case ending. Conversely, if it was an adjective to the "conference," then the sentence would mean "the ninth conference was ..." and would receive a damma (u) as a case ending. Thus, a consideration of context is required for proper disambiguation. Due to the inter-word dependence of CEs, they are typically harder to predict compared to core-word diacritics [5, 22, 23, 40], with CE Error Rate (CEER) of state-of-the-art systems being in double digits compared to nearly 3% for word-cores. Since recovering CEs is akin to shallow parsing [30] and requires morphological and syntactic processing, it is a difficult problem in Arabic NLP. In this article, we focus on recovering both CW diacritics and CEs. We employ two separate Deep Neural Network (DNN) architectures for recovering both kinds of diacritic types. We use character-level and word-level bidirectional Long-Short Term Memory–(biLSTM) based recurrent neural models for CW diacritic and CE recovery respectively. We train models for both MSA and CA. For CW diacritics, the model is informed using word segmentation information and a unigram language model. We also employ a unigram language model to perform post-correction on the model output. We achieve word error rates for CW diacritics of 2.9% and 2.2% for MSA and CA, respectively. The MSA word error rate is 6% lower than the best results in the literature (the RDI diacritizer [39]). The CE model is trained with a rich set of surface, morphological, and syntactic features. The proposed features would aid the biLSTM model in capturing syntactic dependencies indicated by Part-Of-Speech (POS) tags, gender and number features, morphological patterns, and affixes. We show that our model achieves a CEER of 3.7% for MSA and 2.5% for CA. For MSA, this CEER is more than 60% lower than other state-of-the-art systems such as Farasa, which is trained on the same dataset and achieve a CEER of 10.7%. The contributions of this article are as follows:

- We employ a character-level RNN model that is informed using word morphological information and a word unigram language model to recover CW diacritics. Our model beats the best state-of-the-art system by 6% for MSA.

[1]Buckwalter encoding is used in this article [10].

- We introduce a new feature-rich RNN-based CE recovery model that achieve an error rate that is 60% lower than the current state-of-the-art for MSA.
- We explore the effect of different features, which may potentially be exploited for Arabic parsing.
- We show the effectiveness of our approach for both MSA and CA.

2 BACKGROUND

Automatic diacritics restoration has been investigated for many different languages such as European languages (e.g., Romanian [31, 44], French [47], and Croatian [42]), African languages (e.g., Yorba [36]), Southeast Asian languages (e.g., Vietnamese [28]), Semitic language (e.g., Arabic and Hebrew [21]), and many others [18]. For many languages, diacritic (or accent restoration) is limited to a handful of letters. However, for Semitic languages, diacritic recovery extends to most letters. Many general approaches have been explored for this problem including linguistically motivated rule-based approaches, machine learning approaches, such as Hidden Markov Models (HMM) [21] and Conditional Random Fields [14], and lately deep learning approaches for languages such as Arabic [2, 24, 33], Slovak [26], and Yorba [36].

Arabic is a Semitic language with derivational morphology. Arabic nouns, adjectives, adverbs, and verbs are typically derived from a closed set of 10,000 roots of length 3, 4, or rarely 5. Arabic nouns and verbs are derived from roots by applying stem-templates to the roots to generate stems. Such templates may carry information that indicate morphological features of words such POS, gender, and number. For example, given a three-letter root with three consonants CCC, a valid template may be CwACC, where the infix "wA" (وا) is inserted, this template typically indicates an Arabic broken, or irregular, plural template for a noun of template CACC or CACCp if masculine or feminine, respectively. Further, stems may accept prefixes and/or suffixes to form words. Prefixes include coordinating conjunctions, determiners, and prepositions, and suffixes include attached pronouns and gender and number markers. Word stems specify the lexical selection and are typically unaffected by the attached affixes.

Aside from rule-based approaches [19], different methods were used to recover diacritics in Arabic text. Using an HMM [20, 21] with an input character sequence, the model attempts to find the best state sequence given previous observations. Reference [21] reported a 14% word error rate (WER) while Reference [20] achieved a 4.1% diacritic error rate (DER) on the Quran (CA). Reference [45] combined both morphological, acoustic, and contextual features to build a diacritizer trained on FBIS and LDC CallHome ECA collections. They reported a 9% DER without CE, and 28% DER with CE. [35] employed a cascade of finite-state transducers. The cascade stacked a word language model (LM), a charachter LM, and a morphological model. The model achieved an accuracy of 7.33% WER without CE and and 23.61% WER with CE. [46] employed a maximum entropy model for sequence classification. The system was trained on the LDC's Arabic Treebank (ATB) and evaluated on 600 articles from An-Nahar Newspaper (340K words) and achieved a 5.5% DER and a 18% WER on words without CE.

Reference [8] used a hybrid approach that utilizes the output of Alkhalil morphological Analyzer [32] to generate all possible out of context diacritizations of a word, which are passed to an HMM to guess the correct diacritized form. Similarly, Reference [5] used an n-gram language model to guess proper diacritization. The Microsoft Arabic Toolkit Services (ATKS) diacritizer [41] 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 WERs of 11.4% and 4.4% with and without CE. MADAMIRA [38] uses a combinations of morpho-syntactic features to rank a list of potential analyses provided by the Buckwalter Arabic Morphological Analyzer [11]. An SVM trained on ATB

selects the most probable analysis, including the diacritized form. MADAMIRA achieves 19.0% and 6.7% WER with and without CE, respectively [17]. Farasa [17] uses an HMM to guess CW diacritics and an SVM-rank based model trained on morphological and syntactic features to guess CEs. Farasa achieves WERs of 12.8% and 3.3% with and without CEs.

More recent work employed different neural architectures to model the diacritization problem. [2] used a biLSTM recurrent neural network trained on the same dataset as in Reference [46]. They explored one, two, and three biLSTM layers with 250 nodes in each layers, achieving a WER of 9.1% including CE on ATB. Similar architectures were used but achieved lower results [9, 39]. Reference [3] used a hybrid model that combines an RNN-based model with two biLSTM layers with a trigram word model. Reference [4] used a biLSTM based model that is trained on word segments instead of characters or words. Reference [34] modeled diacritization as a sequence-to-sequence translation problem, where undiacritized characters are mapped to diacritized characters. Reference [7] provide a comprehensive survey on Arabic diacritization. A more recent survey by Reference [37] concluded that reported results are often incomparable due to the usage of different test sets. They concluded that a large unigram LM for CW diacritic recovery is competitive with many of the systems in the literature, which prompted us to utilize a unigram language model for post-correction. As mentioned earlier, two conclusions can be drawn, namely restoring CEs is more challenging than CW diacritic restoration, and combining multiple features typically improves CE restoration.

In this article, we expand upon the work in the literature by introducing feature-rich DNN models for restoring both CW and CE diacritics. We compare our models to multiple systems on the same test set. We achieve results that reduce diacritization error rates by more than half compared to the best SOTA systems. We further conduct an ablation study to determine the relative effect of the different features.

3 OUR DIACRITIZER

3.1 Training and Test Corpora

For MSA, we acquired the diacritized corpus that was used to train the Farasa diacritizer [17]. The corpus contains 9.7M tokens with approximately 194K unique surface forms (excluding numbers and punctuation marks). The corpus covers multiple genres such as politics and sports and is a mix of MSA and CA. This corpus is considerably larger than the Arabic Treebank [29] and is more consistent in its diacritization. For testing, we used the freely available WikiNews test set [17], which is composed of 70 MSA WikiNews articles (18,300 tokens) and evenly covers a variety of genres including politics, economics, health, science and technology, sports, arts, and culture.

For CA, we obtained a large collection of fully diacritized classical texts (2.7M tokens) from a book publisher, and we held-out a small subset of 5,000 sentences (approximately 400k words) for testing. Then, we used the remaining sentences to train the CA models.

3.2 Core Word Diacritization

Features. We used four feature types, namely

- **CHAR:** the characters.
- **SEG:** the position of the character in a word segment. For example, given the word "wAlk-tAb" (والكتاب – and the book/writers), which is composed of three segments "w+Al+ktAb" (و+ال+كتاب). Letters were marked as "B" if they begin a segment, "M" if they are in the middle of a segment, "E" if they end a segment, and "S" if they are single letter segments. So for "w+Al+ktAb", the corresponding character positions are "S+BE+BMME." We used

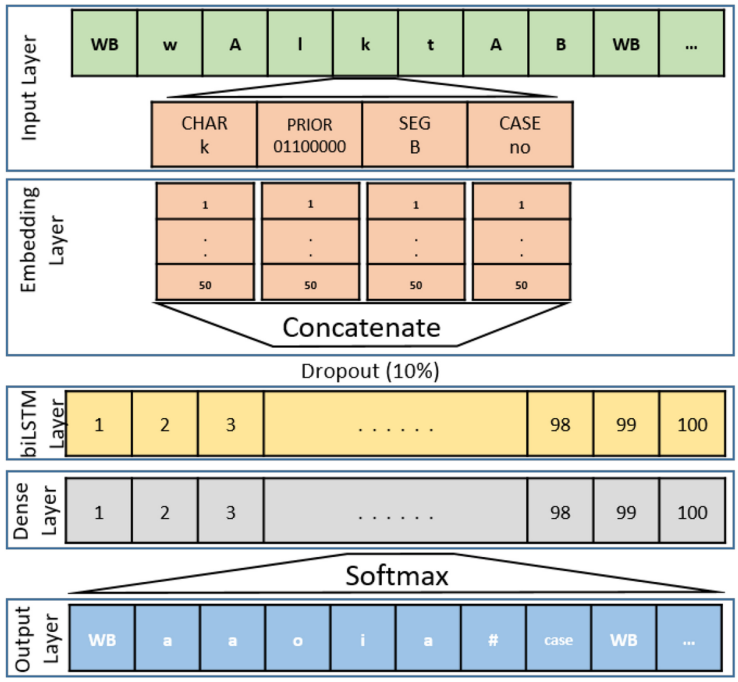

Fig. 1. DNN model for core word diacritics.

Farasa to perform segmentation, which has a reported segmentation accuracy of 99% on the WikiNews dataset [16].

- **PRIOR:** diacritics seen in the training set per segment. Since we used a character-level model, this feature informed the model with word-level information. For example, the word "ktAb" (كتاب) was observed to have two diacritized forms in the training set, namely "k**i**t**a**Ab" (كِتَاب – book) and "k**u**t~**a**Ab" (كُتَّاب – writers). The first letter in the word ("k") accepted the diacritics "i" and "u." Thus, given a binary vector representing whether a character is allowed to assume any of the eight primitive Arabic diacritic marks (a, i, u, o, K, N, F, and ~ in order), the first letter would be given the following vector "01100000." If a word segment was never observed during training, then the vector for all letters therein would be set to 11111111. This feature borrows information from HMM models, which have been fairly successful in diacritizing word cores.
- **CASE:** whether the letter expects a core word diacritic or a case ending. Case endings are placed on only one letter in a word, which may or may not be the last letter in the word. This is a binary feature.

DNN Model. Using a DNN model, particularly with a biLSTM [43], is advantageous, because the model automatically explores the space of feature combinations and is able to capture distant dependencies. A number of studies have explored various biLSTM architectures [2, 9, 39] including stacked biLSTMs confirming their effectiveness. As shown in Figure 1, we employed a character-based biLSTM model with associated features for each character. Every input character had an associated list of m features, and we trained randomly initialized embeddings of size 50 for each feature. Then, we concatenated the feature embeddings vectors creating an $m \times 50$ vector for each character, which was fed into the biLSTM layer of size 100. The output of the biLSTM layer was

Table 1. Features with Examples and Motivation

Feature	Example	Explanation and Motivation
word	w+b+mktb+t+nA (و+ب+مكتبة+نا – and in our library)	Some words have a fixed set of observed CEs
word POS	CONJ+PREP+NOUN +NSUFF+PRON	Some POS combinations allow a closed set of CEs
gender/number	feminine/singular	Gender/number agreement (dis)allow certain attachments and may allow/exclude certain CEs
stem	mktb+p (مكتبة – library)	We attach gender and number noun suffixes such the singular feminine marker "p" (ـة), because CEs appear on them.
stem POS	NOUN+NSUFF	Same rationale as word POS
prefix(es) & POS	w+b+ (و+ب+) & CONJ+PREP	Certain prefixes affect CE directly. For example, the PREP "b+" (ب+) is a preposition causing their noun predicates to assume the genitive case
suffix(es) & POS	"+nA" (+نا) & PRON	Certain suffixes affect CE directly
stem template	mfEl+p (مفعلة – derived from the root 'ktb" كتب)	Some stem templates allow certain CEs and exclude others. Ex. the stem template ">fEl" (أفعل) disallows tanween ("N", "K", "F")
word/stem head/tail char uni/bi-grams	word: w (و), wb (و ب); stem: A (ا), nA (نا)	Such characters can capture some morphological and syntactic information. Ex. verbs in present tense start with "> (أ), n (ن), y (ي), or t (ت)".
sukun word	foreign NEs: ex. jwn (جون – John)	CE of certain words is strictly *sukun*. We built a list from training set.
named entities	NEs	Named entities are more likely to have *sukun* as CE. We extracted the named entity list from the Farasa named entity recognizer [13, 15].

fed directly into a dense layer of size 100. We used early stopping with patience of five epochs, a learning rate of 0.001, a batch size of 256, and an Adamax optimizer [27]. The input was the character sequence in a sentence with words being separated by word boundary markers, and we set the maximum sentence length to 1,250 characters.

3.3 Case Ending Diacritization

Features. Table 1 lists the features that we used for CE recovery. We used Farasa to perform segmentation and POS tagging and to determine stem-templates [17]. Farasa has a reported POS accuracy of 96% on the WikiNews dataset [17]. Though the Farasa diacritizer utilizes a combination of some the features presented herein, namely segmentation, POS tagging, and stem templates, Farasa's SVM-ranking approach requires explicit specification of feature combinations (ex. $Prob(CE||current\_word, prev\_word, prev\_CE)$). Manual exploration of the feature space is undesirable, and ideally we would want our learning algorithm to do so automatically. The flexibility of the DNN model allowed us to include many more surface-level features such as affixes, leading, and trailing characters in words and stems and the presence of words in large gazetteers of named entities. As we show later, these additional features significantly lowered CEER.

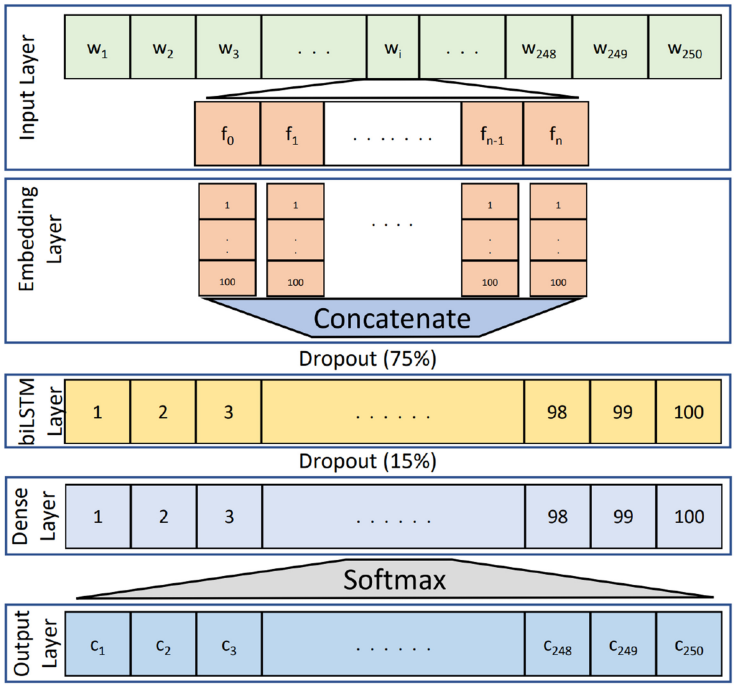

Fig. 2. DNN case ending model architecture.

DNN Model. Figure 2 shows the architecture of our DNN algorithm. Every input word had an associated list of n features, and we trained randomly initialized embeddings of size 100 for each feature. Then, we concatenated the feature embeddings vectors creating an $n \times 100$ vector for each word. We fed these vectors into a biLSTM layer of 100 dimensions after applying a dropout of 75%, where dropout behaves like a regularlizer to avoid overfitting [25]. We conducted side experiments with lower dropout rates, but the higher dropout rate worked best. The output of the biLSTM layer was fed into a 100-dimensional dense layer with 15% dropout and softmax activation. We conducted side experiments where we added additional biLSTM layers and replaced softmax with a conditional random fields layer, but we did not observe improvements. Thus, we opted for a simpler model. We used a validation set to determine optimal parameters such as dropout rate. Again, we used the "Adamax" optimizer [27] with categorical cross entropy loss and a learning rate of 0.001. We also applied early stopping with patience of up to five consecutive epochs without improvement.

4 EXPERIMENTS AND RESULTS

4.1 Core Word

Experimental Setup. For all the experiments conducted herein, we used the Keras toolkit [12] with a TensorFlow backend [1]. We used the entirety of the training set as input, and we instructed Keras to use 5% of the data for tuning (validation). Aside from experiments on CA and MSA individually, we trained a unified model using both CA and MSA training data and using the best features. We included the CASE feature, which specifies whether the letter accepts a normal diacritic or case ending, in all our setups. We conducted multiple experiment using different features, namely

Table 2. Core Word Diacritization Results

	MSA				CA			
	DNN		DNN+Post		DNN		DNN+Post	
Model	WER	DER	WER	DER	WER	DER	WER	DER
CHAR	3.5	1.1	3.3	1.0	5.1	2.1	2.7	1.0
CHAR+SEG	3.3	1.1	3.2	1.0	4.7	1.9	2.6	1.0
CHAR+PRIOR	3.8	1.2	3.7	1.1	3.8	1.6	2.3	0.9
ALL	**3.0**	**1.0**	**2.9**	**0.9**	**3.6**	**1.5**	**2.2**	**0.9**

- **CHAR:** This is our baseline setup where we only used the characters as features.
- **CHAR+SEG:** This takes the characters and their segmentation information as features.
- **CHAR+PRIOR:** This takes the characters and their observed diacritized forms in the training set.
- **All:** This setup includes all the features.

We also optionally employed post-correction. For words that were seen in training, if the model produced a diacritized form that was not seen in the training data, we assumed it was an error and replaced it with the most frequently observed diacritized form (using a unigram language model). We report two error rates, namely WER (at word level) and DER (at character level). We used relaxed scoring where we assumed an empty case to be equivalent to *sukun*, and we removed default diacritics: *fatHa* followed by *alef*, *kasra* followed by *ya*, and *damma* followed by *wa*. Using such scoring would allow us to compare to other systems in the literature that may use different diacritization conventions.

Results and Error Analysis. For testing, we used the aforementioned WikiNews dataset to test the MSA diacritizer and the held-out 5,000 sentences for CA. Table 2 shows WER and DER results using different features with and without post-correction.

MSA Results: For MSA, though the CHAR+PRIOR feature led to worse results than using CHAR alone, the results show that combining all the features achieved the best results. Moreover, post-correction improved results overall. When analyzing why CHAR+PRIOR performed worse than CHAR alone, we found that most of the additional errors involved words that were not previously seen during training and hence all the different diacritics were likely. Roughly 5.3% of the words were not seen during training. When we added the segmentation information, such cases not only decreased, but also the classifier had other features to rely on. We compare our results to five other systems, namely Farasa [17], MADAMIRA [38], RDI [39], MIT (Belinkow and Glass, 2015), and Microsoft ATKS [41]. Table 3 compares our system with the aforementioned systems. As the results show, our results beat the current state of the art.

For error analysis, we analyzed all the errors (527 errors). The errors types along with examples of each are shown in Table 4. The most prominent error type arises from the selection of a valid diacritized form that does not match the context (40.8%). Perhaps, including POS tags as a feature or augmenting the PRIOR feature with POS tag information and a bigram language model may reduce the error rate further. The second most common error is due to transliterated foreign words including foreign named entities (23.5%). Such words were not observed during training. Further, Arabic Named entities account for 10.6% of the errors, where they were either not seen in training or they share identical non-diacritized forms with other words. Perhaps, building larger gazetteers of diacritized named entities may resolve NE related errors. In 10.8% of the cases, the diacritizer produced completely incorrect diacritized forms. In some of the cases (9.1%), though the diacritizer

Table 3. Comparing Our System to State-of-the-art Systems: Core Word Diacritics

System	Error Rate WER	DER
MSA		
Our system	**2.9**	**0.9**
[39]	3.0	1.0
Farasa	3.3	1.1
Microsoft ATKS	5.7	2.0
MADAMIRA	6.7	1.9
[9]	14.9	3.9
CA		
Our system	2.2	0.9
Our best MSA system on CA	8.5	3.7

Table 4. Error Analysis: Core Word Error Types for MSA

Error	Freq.	%	Explanation	Examples
Wrong selection	215	40.8	Homographs with different diacritized forms	"qaSor" (قَصْر – palace) vs. "qaSar" (قَصَر – he limited)
Foreign word	124	23.5	transliterated words including 96 foreign named entities	wiykiymaAnoyaA (وِيكِيمَانْيَا – Wikimania)
Invalid diacritized form	57	10.8	invalid form	ya*okur (يَذْكُر – he mentions) vs. ya*okar (يَذْكَر)
Named entity	56	10.6	Arabic named entities	"Eab~Adiy" (عَبَّادِي – name) vs. "EibAdiy" (عِبَادِي – my servants)
both correct	48	9.1	Some words have multiple valid diacritized forms	"wikAlap" (وِكَالَة) and "wakAlap" (وَكَالَة – agency)
Affix diacritization error	16	3.0	Some prefixes or suffixes are erroneously diacritized	b***a***Akt$Afihim (بَاكتشافِهِم – with their discovery)
Reference is wrong	10	1.9	the truth diacritics were incorrect	AlofiyfaA (الْفِيفَا – FIFA) vs. AlofayofaA (الْفَيْفَا)
dialectal word	1	0.2	dialectal word	mawaAyiliy (مَوَايِلِي – my chant)

produced a form that is different from the reference, both forms were in fact correct. Most of these cases were due to variations in diacritization conventions (ex. "bare alef" (A) at start of a word receiving a diacritic or not). Other cases include foreign words, and cases where both diacritized forms are equally valid.

CA Results: For CA results, the CHAR+SEG and CHAR+PRIOR performed better than using characters alone with CHAR+PRIOR performing better than CHAR+SEG. We suspect that CHAR+PRIOR performed better than CHAR alone, because the percentage of words that were not

Table 5. Error Analysis: Core Word Error Types for CA

Error	Freq.	%	Explanation	Examples
Invalid diacritized form	195	38.8	invalid form	"<aqosaAm" (إِقْسَام – portions) vs. ">aqasaAm" (أَقْسَام)
Wrong selection	157	31.4	Homographs with different diacritized forms	"rafoE" (رَفْع – lifting) vs. "rafaE" (رَفَعَ – he lifted)
Affix diacritization error	66	13.2	Some affixes are erroneously diacritized	"baladhu" (بَلَدُهُ – his country, where country is subject of verb) vs. "baladhi" (بَلَدِهِ – his country, where country is subject or object of preposition)
Named entities	44	8.8	Named entities	"Alr~ayob" (الرَّيْب – Suspicion) vs. "Alr~iyab" (الرِّيَب)
Problems with reference	22	4.4	Some words in the reference were partially diacritized	"nuEoTaY" (نُعْطَى – we are given) vs. "nETY" (نعطى)
Guess has no diacritics	9	1.8	system did not produce any diacritics	"mhnd" (مهند – sword or a person name) vs. "muhan~ad" (مُهَنَّد)
Different valid forms	7	1.4	Some words have multiple valid diacritized forms	"maA}op" (مَائْة – hundred) and "miA}op" (مِائَة)
Misspelled word	1	0.2		"lbAlmsjd" (لبالمسجد) vs. "bAlmsjd" (بالمسجد – in the mosque)

seen during training was relatively low (1.4% compared to 5.3% for MSA). As in the case of MSA, combining all the features led to the best results. Post correction had a significantly larger positive impact on results compared to what we observed for MSA. This may indicate that we need a larger training set. The best WER that we achieved for CW diacritics with post-corrections is 2.2%. Since we did not have access to any publicly available system that is tuned for CA, we compared our best system to using our best MSA system to diacritize the CA test set, and the MSA diacritizer produced significantly lower results with a WER of 8.5% (see Table 3). This highlights the large difference between MSA and CA and the need for systems that are specifically tuned for both.

We randomly selected and analyzed 500 errors (5.2% of the errors). The errors types along with examples of each are shown in Table 5. The two most common errors involve the system producing completely incorrect diacritized forms (38.8%) or correct forms that do not match the context (31.4%). As for MSA, we suspect that adding additional POS information and employing a word bigram to constrain the PRIOR feature may help reduce selection errors. Another prominent error is related to the diacritics that appear on attached suffixes, particularly pronouns, which depend on the choice of case ending (13.2%). Errors due to named entities are slightly fewer than those seen for MSA (8.8%). A noticeable number of mismatches between the guess and the reference are due to partial diacritization of the reference (4.4%). We plan to conduct an extra round of checks on the test set.

CA+MSA Results: For the unified CA+MSA model, we trained the model using all the features. As Table 6 shows, the results were consistently lower than training separate CA and MSA models,

Table 6. Unified MSA+CA Core Word Results

MSA				CA			
DNN		DNN+Post		DNN		DNN+Post	
WER	DER	WER	DER	WER	DER	WER	DER
4.8	1.5	4.6	1.5	3.8	1.5	3.2	1.3

with MSA results being impacted more than CA results with a 1.7% absolute increase in WER. We inspected the errors that occurred when using the combined model but not when using the individual models, and they fell in the same categories as those seen for CA and MSA. However, we suspect that the divergence in word usage between CA and MSA was the main contributor to the additional errors. For example, the word $bAk (شباك) can be diacritized as "$ibAk" to mean "nets" or "$ub~Ak" meaning window, with the later being an MSA diacritization as opposed to a CA one.

4.2 Case Ending

Experimental Setup. We conducted multiple experiments to determine the relative effect of the different features as follows:

- **word:** This is our baseline setup, which uses word surface forms only.
- **word-surface:** This setup uses the word surface form, stem, prefixes, and suffixes (including noun suffixes). This simulates the case when no POS tagging information is available.
- **word-POS:** This includes the word surface form and POS information, including gender and number of stem, prefixes, and suffixes.
- **word-morph:** This includes a word and its stem template to capture morphological patterns.
- **word-surface-POS-morph:** This setup uses all the features (surface, POS, and morphological).
- **all-misc:** This uses all features plus word and stem leading and trailing character unigrams and bigrams in addition to *sukun* words and named entities.

For testing MSA, we used the aforementioned WikiNews dataset. Again, we compared our results to five other systems, namely Farasa [17], MADAMIRA [38], RDI (Rashwan et al., 2015), MIT (Belinkow and Glass, 2015), and Microsoft ATKS [41]. For CA testing, we used the 5,000 sentences that we set aside. Again, we compared to our best MSA system.

Results and Error Analysis. Table 7 lists the results of our setups compared to other systems.

MSA Results: As the results show, our baseline DNN system outperforms all state-of-the-art systems. Further, adding more features yielded better results overall. Surface-level features resulted in the most gain, followed by POS tags, and last stem templates. Further, adding head and tail characters along with a list of *sukun* words and named entities led to further improvement. Our proposed feature-rich system has a CEER that is approximately 61% lower than any of the state-of-the-art systems.

Figure 3 shows CE distribution and prediction accuracy. For the four basic markers *kasra*, *fatHa*, *damma* and *sukun*, which appear 27%, 14%, 9% and 10% respectively, the system has CEER of ~1% for each. Detecting the virtual CE mark is a fairly easy task. All other CE markers represent 13% with almost negligible errors.

Table 7. MSA Results and Comparison to Other Systems

Setup	CEER%
MSA	
word (baseline)	9.1
word-surface	5.7
word-POS	7.0
word-morph	7.6
word-surface-POS-morph	5.2
all-misc	**3.7**
all-misc CA+MSA	**5.5**
Microsoft ATKS	9.5
Farasa	10.4
RDI [39]	14.0
MIT [9]	15.3
MADAMIRA [38]	15.9
CA	
word (baseline)	4.0
word-surface	3.3
word-POS	3.1
word-morph	3.7
word-surface-POS-morph	2.9
all-misc	**2.5**
all-misc CA+MSA	**3.0**
Our best MSA system on CA	8.9

Table 8 lists a thorough breakdown of all errors accounting for at least 1% of the errors along with the most common reasons for the errors and examples illustrating these reasons. For example, the most common error type involves guessing a fatHa (a) instead of damma (u) or vice versa (19.3%). The most common reasons for this error type, based on inspecting the errors, were due to: POS errors (ex. a word is tagged as a verb instead of a noun); and a noun is treated as a subject instead of an object or vice versa. The table details the rest of the error types. Overall, some of the errors are potentially fixable using better POS tagging, improved detection of non-Arabized foreign names, and detection of indeclinability. However, some errors are more difficult and require greater understanding of semantics such as improper attachment, incorrect idafa, and confusion between subject and object. Perhaps, such semantic errors can be resolved using parsing.

CA Results: The results show that the POS tagging features led to the greatest improvements followed by the surface features. Combining all features led to the best results with WER of 2.5%. As we saw for CW diacritics, using our best MSA system to diacritize CA led to significantly lower results with CEER of 8.9%.

Figure 4 shows CE distribution and prediction accuracy values. For the four basic markers *fatHa*, *kasra*, *damma*, and *sukun*, which appear 18%, 14%, 13%, and 8%, respectively, the system has CEERs of ~0.5% for each. Again, detecting the virtual CE mark was a fairly easy task. All other CE markers representing 20% have negligible errors.

Table 9 lists all the error types, which account for at least 1% of the errors, along with their most common causes and explanatory examples. The error types are similar to those observed for MSA.

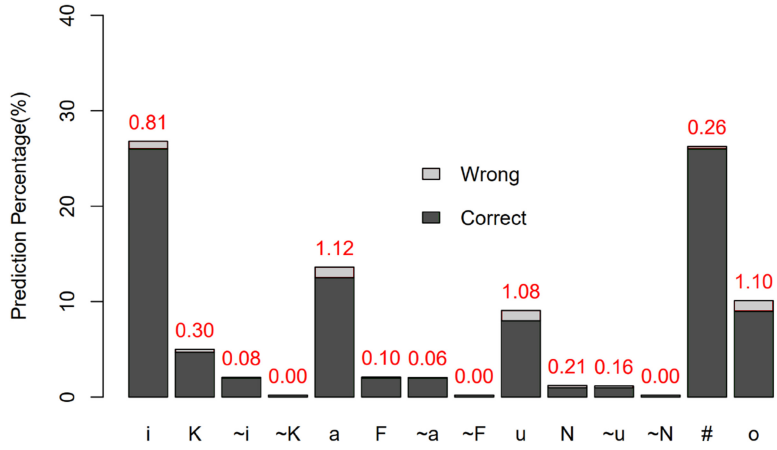

Fig. 3. Case endings distribution and prediction accuracy for MSA.

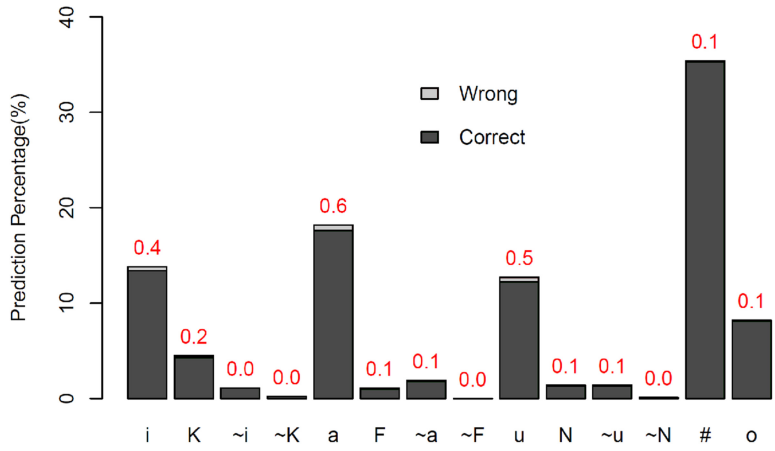

Fig. 4. Case endings distribution and prediction accuracy for CA.

Some errors are more syntactic and morphological in nature and can be addressed using better POS tagging and identification of indeclinability, particularly as they relate to named entities and nouns with feminine markers. Other errors such as incorrect attachment, incorrect idafa, false subject, and confusion between subject and object can perhaps benefit from the use of parsing. As with the core-word errors for CA, the reference had some errors (ex. {a,i,o} ⇒ #), and extra rounds of reviews of the reference are in order.

CA+MSA Results. The results show that combining both CA and MSA training data to create a unified model leads to results that are lower than using individual MSA and CA models, with results for MSA being impacted more than CA results with an increase of 1.8% (absolute) in CEER. Again, we inspected the errors that resulted from the unified model but not from the individual models, and it seems that a substantial portion of the errors resulted from divergent MSA and CA senses. For example, the word "tHdv" (تحدث) is commonly diacritized as "taHad~av**a**" in MSA (he

Table 8. MSA Case Errors Accounting from More than 1% of Errors

Error	Count	%	Most Common Causes
a ⇔ u	133	19.3	*POS error*: ex. "ka$afa" (كَشَفَ – he exposed) vs. "ka$ofu" (كَشْفُ – exposure) *& Subject vs. object*: ex. "tuwHy **mivolu**" (تُوحِي مِثْلُ – **such** indicates) vs. "tuwHy **mivola**" (تُوحِي مِثْلَ – she indicates **such**)
i ⇔ a	130	18.9	*Incorrect attachment* (due to coordinating conjunction or distant attachment): ex. "AlogAza Alomusay~ili lildumuwEi – **wa+AlraSASi** vs. **wa+AlraSASa** (الغَازَ الْمُسَيِّلَ لِلدُّمُوعِ والرَصَاص – tear gas and bullets) where bullets were attached incorrectly to tear instead of gas *& indeclinability such as foreign words and feminine names*: ex. "kaAnuwni" (كَانُونِ – Cyrillic month name) vs. "kaAuwna" (كَانُونَ)
i ⇔ u	95	13.8	*POS error of previous word*: ex. "tadahowuru **waDoEihi**" (تَدَهْوُرُ وَضْعِهِ – deterioration of his situation – situation is part of Idafa construct) vs. "tadahowara **waDoEuhu**" (تَدَهْوَرَ وَضْعُهُ – his situation deteriorated – situation is subject) *& Incorrect attachment* (due to coordinating conjunction or distant attachment): (as example for i ⇔ a)
a ⇔ o	60	8.7	*Foreign named entities*: ex. "siyraAloyuwna" (سِيرَالْيُونَ – Siera Leon) vs. "siyraAloyuwno" (سِيرَالْيُونْ)
i ⇔ K	27	4.0	*Incorrect Idafa*: "**liAt~ifaAqi**ha*aA Alo>usobuwE" (لِاتِفَاقِ هَذَا الأُسْبُوع – this week's agreement) vs. "**liAt~ifaAqK** ha*aA Alo>usobuwE" (لِاتِّفَاقٍ هَذَا الْأُسْبُوع – to an agreement this week)
K ⇔ N	29	4.2	*Subject vs. object* (as in a ⇔ u) and Incorrect attachment (as in i ⇔ a)
F ⇔ N	25	3.7	*Words ending with feminine marker "p" or "At"*: ex. "muHaADarap" (مُحَاضَرَة – lecture)
i ⇔ o	22	3.2	*Foreign named entities* (as in a ⇔ o)
F ⇔ a	16	2.3	*Incorrect Idafa* (as in i ⇔ K)
u ⇔ o	14	2.0	*Foreign named entities* (as in a ⇔ o)
F ⇔ K	9	1.3	*Words ending with feminine marker* (as in F ⇔ N)
K ⇔ a	8	1.2	*Incorrect Idafa* (as in i ⇔ K)

spoke) and rarely as "tuHad~iv**u**" (she/you narrated). The later sense is significantly more common in CA. Thus, a large proportion of the additional errors follow from core word diacritization errors.

4.3 Full Diacritization Results

Table 10 compares the full word diacritization (CW+CE) of our best setup to other systems in the literature. As the results show for MSA, our overall diacritization WER is 6.0% while the state-of-the-art system has a WER of 12.2%. As for CA, our best system produced an error rate of 4.3%, which is significantly better than using our best MSA system to diacritize CA.

5 CONCLUSION AND FUTURE WORK

In this article, we presented a feature-rich DNN approach for MSA, CW, and CE recovery that produces a word-level error for MSA of 6.0%, which is more than 50% lower than

Table 9. CA Case Errors Accounting from More than 1% of Errors

Error	Count	%	Most Common Causes
a ⇔ u	2,907	28.4	*Subject vs. object*: ex. "wafaqa **yawoma**" (وَفَقَ يَوْمَ – he matches the day) vs. ex. "wafaqa **yawomu**" (وَفَقَ يَوْمُ – the day matches) & *False subject* (object behaves like subject in passive tense): ex. "yufar~iqu **AloqaDaA'a**" (يُفَرِّقُ الْقَضَاءَ – he separates the make up) vs. "yufar~aqu **AloqaDaA'u**" (يُفَرَّقُ الْقَضَاءُ – the make up is separated) & *Incorrect attachment* (due to coordinating conjunction): ex. "f+a>aEohadu" (فَأَعْهَدَ – so I entrust) vs. "f+a>aEohidu" (فَأَعْهِدُ)
i ⇔ u	1,316	12.9	*Incorrect attachment* (due to coordinating conjunctions or distant attachment): (as in a ⇔ u)
i ⇔ a	1,019	10.0	*Incorrect attachment* (as in a ⇔ u) & *Indeclinability such as foreign words and feminine names*: ex. ">ajoyaAdiyni" (أَجْيَادِينِ – Ajyadeen (city name)) vs. ">ajoyaAdiyna" (أَجْيَادِينَ)
a ⇔ #	480	4.7	*Problem with reference where the case for some words, particularly non-Arabic names, is not provided in the reference:* ex. "<isoHaAq" (إِسْحَاق – Issac) vs. "<isoHaAqa" (إِسْحَاقَ)
u ⇔ #	426	4.2	same problems as in a ⇔ #
K ⇔ i	371	3.6	*Incorrect Idafa*: ex. "**EaTaA'i** Alofaqiyh" (عَطَاءِ الْفَقِيه – the providence of the jurist) vs. "**EaTaA'K** Alofaqiyh" (عَطَاءٍ الْفَقِيه – Ataa the jurist)
K ⇔ a	328	3.2	*words ending with feminine marker*: ex. "tayomiyap" (تَيْمِيَة –Taymiya) & *Indeclinability:* ex. "wabi<i$obiyliy~ap" (وَ بِإِشْبِيلِيَّة – and in Lisbon)
u ⇔ o	300	2.9	*confusion between past, present, and imperative moods of verbs and preceding markers (imperative "laA" vs. negation "laA):* ex. "laA tano$ariHu" (لا تَنْشَرِحُ – does not open up) vs. "laA tano$ariHo" (لا تَنْشَرِحْ – do not open up)
a ⇔ o	278	2.7	*confusion between past, present, and imperative moods of verbs* (as in u ⇔ o)
K ⇔ N	253	2.5	*Incorrect attachment* (as in i ⇒ u)
N ⇔ u	254	2.5	*Incorrect Idafa* (as in K ⇒ i)
F ⇔ N	235	2.3	*words ending with feminine marker* (as in K ⇒ a)
i ⇔ o	195	1.9	*Differing conventions concerning handling two consecutive letters with sukun*: ex. "**Eano** Aboni" (عَنْ ابْنِ – on the authority of the son of) vs. "**Eani** Aboni" (عَنِ ابْنِ)
i ⇔ #	178	1.7	same errors as for a ⇒ #
o ⇔ #	143	1.4	same errors as for a ⇒ #

state-of-the-art systems (6.0% compared to 12.2%) and a word error rate of 4.3% for CA. Specifically, we used a biLSTM-based model with a variety of surface, morphological, and syntactic features. Reliable NLP tools may be required to generate some of these features, and such tools may not be readily available for other language varieties, such as dialectal Arabic. However, we showed the efficacy of different varieties of features, such as surface-level features, and they can help improve

Table 10. Comparison to Other Systems for Full Diacritization

Setup	WER%
MSA	
Our System	**6.0**
Microsoft ATKS	12.2
Farasa	12.8
RDI [39]	16.0
MADAMIRA [38]	19.0
MIT [9]	30.5
CA	
Our system	**4.3**
Our best MSA system on CA	14.7

diacritization individually. Further, though some errors may be overcome using improved NLP tools (ex. better POS tagging), semantic errors, such incorrect attachments, are more difficult to fix. Perhaps, using dependency parsing may help overcome some semantic errors. As for feature engineering, the broad categories of features, such as surface, syntactic, and morphological features, may likely carry-over to other languages, and language-specific feature engineering may be require to handle the specificity of each language. Last, since multiple diacritization conventions may exist, as in the case of Arabic, adopting one convention consistently is important for training a good system and for properly testing it. Though we have mostly achieved this for MSA, the CA dataset requires more checks to insure greater consistency.

For future work, we want to explore the effectiveness of augmenting our CW model with POS tagging information and a bigram language model. Further, we plan to create a multi-reference diacritization test set to handle words that have multiple valid diacritized forms. For CE, we want to examine the effectiveness of the proposed features for Arabic parsing. We plan to explore character-level convolutional neural networks that may capture sub-word morphological features, pre-trained embeddings, and attention mechanisms to focus on salient features. We also plan to explore joint modeling for both core word and case ending diacritics.

REFERENCES

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. Retrieved from https://www.tensorflow.org/.

[2] 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. *Int. J. Doc. Anal. Recogn.* 18, 2 (2015), 183–197.

[3] Hamza Abbad and Shengwu Xiong. 2020. Multi-components system for automatic Arabic diacritization. In *Proceedings of the European Conference on Information Retrieval*. Springer, 341–355.

[4] Sawsan Alqahtani and Mona Diab. 2019. Investigating input and output units in diacritic restoration. In *Proceedings of the 2019 18th IEEE International Conference on Machine Learning and Applications (ICMLA'19)*. IEEE, 811–817.

[5] Mohamed Seghir Hadj Ameur, Youcef Moulahoum, and Ahmed Guessoum. 2015. Restoration of Arabic diacritics using a multilevel statistical model. In *Proceedings of the IFIP International Conference on Computer Science and Its Applications_x000D_*. Springer, 181–192.

[6] Mohammed Attia. 2008. *Handling Arabic Morphological and Syntactic Ambiguity within the LFG Framework with a View to Machine Translation*. Ph.D. Thesis. School of Languages, Linguistics and Cultures, The University of Manchester, UK.
[7] Aqil M. Azmi and Reham S. Almajed. 2015. A survey of automatic Arabic diacritization techniques. *Nat. Lang. Eng.* 21, 03 (2015), 477–495.
[8] Mohamed Bebah, Chennoufi Amine, Mazroui Azzeddine, and Lakhouaja Abdelhak. 2014. Hybrid approaches for automatic vowelization of Arabic texts. *arXiv:1410.2646*. Retrieved from https://arxiv.org/abs/1410.2646.
[9] 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*. 2281–2285.
[10] Tim Buckwalter. 2002. Buckwalter {Arabic} morphological analyzer version 1.0.
[11] Tim Buckwalter. 2004. Buckwalter Arabic morphological analyzer version 2.0. *LDC Catalog Number LDC2004L02*.
[12] François Chollet et al. 2015. Keras. Retrieved from https://keras.io.
[13] Kareem Darwish. 2013. Named entity recognition using cross-lingual resources: Arabic as an example. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Vol. 1. 1558–1567.
[14] Kareem Darwish, Ahmed Abdelali, Hamdy Mubarak, Younes Samih, and Mohammed Attia. 2018. Diacritization of Moroccan and Tunisian Arabic dialects: A CRF approach. In *Proceedings of the 3rd Workshop on Open-Source Arabic Corpora and Processing Tools (OSACT 3)*. 62.
[15] Kareem Darwish and Wei Gao. 2014. Simple effective microblog named entity recognition: Arabic as an example. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC'14)*. 2513–2517.
[16] Kareem Darwish and Hamdy Mubarak. 2016. Farasa: A new fast and accurate Arabic word segmenter. In *Proceedings of the 10th International Conference on Language Resources and Evaluation (LREC'16)*. European Language Resources Association (ELRA).
[17] Kareem Darwish, Hamdy Mubarak, and Ahmed Abdelali. 2017. Arabic diacritization: Stats, rules, and hacks. In *Proceedings of the 3rd Arabic Natural Language Processing Workshop*. 9–17.
[18] Guy De Pauw, Peter W. Wagacha, and Gilles-Maurice De Schryver. 2007. Automatic diacritic restoration for resource-scarce languages. In *Proceedings of the International Conference on Text, Speech and Dialogue*. Springer, 170–179.
[19] Tarek A. El-Sadany and Mohamed A. Hashish. 1989. An Arabic morphological system. *IBM Syst. J.* 28, 4 (1989), 600–612.
[20] Moustafa Elshafei, Husni Al-Muhtaseb, and Mansour Alghamdi. 2006. Statistical methods for automatic diacritization of Arabic text. In *Proceedings of the Saudi 18th National Computer Conference*, Vol. 18. 301–306.
[21] 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*. Association for Computational Linguistics, 1–7.
[22] Nizar Habash and Owen Rambow. 2007. Arabic diacritization through full morphological tagging. In *Proceedings of the Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Companion Volume, Short Papers*. Association for Computational Linguistics, 53–56.
[23] Salima Harrat, Mourad Abbas, Karima Meftouh, and Kamel Smaili. 2013. Diacritics restoration for Arabic dialects. In *Proceedings of the 14th Annual Conference of the International Speech Communication Association (INTERSPEECH'13)*. ISCA.
[24] Y. Hifny. 2018. Hybrid LSTM/MaxEnt networks for Arabic syntactic diacritics restoration. *IEEE Sign. Process. Lett.* 25, 10 (Oct. 2018), 1515–1519. DOI: https://doi.org/10.1109/LSP.2018.2865098
[25] Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R. Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv:1207.0580*. Retrieved from https://arxiv.org/abs/1207.0580.
[26] A. Hucko and P. Lacko. 2018. Diacritics restoration using deep neural networks. In *Proceedings of the 2018 World Symposium on Digital Intelligence for Systems and Machines (DISA'18)*. 195–200. DOI: https://doi.org/10.1109/DISA.2018.8490624
[27] Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. arxiv:cs.LG/1412.6980. Retrieved from https://arxiv.org/abs/1412.6980.
[28] Tuan Anh Luu and Kazuhide Yamamoto. 2012. A pointwise approach for Vietnamese diacritics restoration. In *Proceedings of the 2012 International Conference on Asian Language Processing*. IEEE, 189–192.
[29] Mohammed Maamouri, Ann Bies, Tim Buckwalter, and Wigdan Mekki. 2004. The Penn Arabic treebank: Building a large-scale annotated Arabic corpus. In *Proceedings of the NEMLAR Conference on Arabic Language Resources and Tools*. 102–109.
[30] Yuval Marton, Nizar Habash, and Owen Rambow. 2010. Improving Arabic dependency parsing with lexical and inflectional morphological features. In *Proceedings of the NAACL HLT 2010 First Workshop on Statistical Parsing of Morphologically-Rich Languages*. Association for Computational Linguistics, 13–21.

[31] Rada F. Mihalcea. 2002. Diacritics restoration: Learning from letters versus learning from words. In *Proceedings of the International Conference on Intelligent Text Processing and Computational Linguistics*. Springer, 339–348.
[32] Bebah Mohamed Ould Abdallahi Ould, Abdelouafi Meziane, Azzeddine Mazroui, and Abdelhak Lakhouaja. 2011. Alkhalil MorphoSys. In *Proceedings of the 7th International Computing Conference in Arabic*. 66–73.
[33] Hamdy Mubarak, Ahmed Abdelali, Hassan Sajjad, Younes Samih, and Kareem Darwish. 2019. Highly effective Arabic diacritization using sequence to sequence modeling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Association for Computational Linguistics, 2390–2395.
[34] Hamdy Mubarak and Kareem Darwish. 2014. Automatic correction of Arabic text: A cascaded approach. In *Proceedings of the Arabic NLP Workshop (EMNLP'14)*.
[35] 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*. Association for Computational Linguistics, 79–86.
[36] Iroro Orife. 2018. Attentive sequence-to-sequence learning for diacritic restoration of Yorùbá language text. *arXiv:1804.00832*. Retrieved from https://arxiv.org/abs/1804.00832.
[37] Osama Hamed and Torsten Zesch. 2017. A survey and comparative study of arabic diacritization tools. *J. Lang. Technol. Comput. Ling.* 32, 1 (2017), 27–47.
[38] 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 *Proceedings of the International Conference on Language Resources and Evaluation (LREC'14)*.
[39] Mohsen Rashwan, Ahmad Al Sallab, M. Raafat, and Ahmed Rafea. 2015. Deep learning framework with confused sub-set resolution architecture for automatic Arabic diacritization. *IEEE Trans. Aud. Speech Lang. Process.* 23, 3 (2015), 505–516.
[40] 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*. Association for Computational Linguistics, 117–120.
[41] Ahmed Said, Mohamed El-Sharqwi, Achraf Chalabi, and Eslam Kamal. 2013. A hybrid approach for Arabic diacritization. In *Natural Language Processing and Information Systems*, Elisabeth Métais, Farid Meziane, Mohamad Saraee, Vijayan Sugumaran, and Sunil Vadera (Eds.). Springer, Berlin, 53–64.
[42] Nikola Šantić, Jan Šnajder, and Bojana Dalbelo Bašić. 2009. Automatic diacritics restoration in Croatian texts. In *INFuture2009: Digital Resources and Knowledge Sharing* (2009), 309–318.
[43] Mike Schuster and Kuldip K. Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Trans. Sign. Process.* 45, 11 (1997), 2673–2681.
[44] Dan Tufiş and Alexandru Ceauşu. 2008. DIAC+: A professional diacritics recovering system. *Proceedings of the International Conference on Language Resources and Evaluation (LREC'08)*.
[45] 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)*. Association for Computational Linguistics, 66–73.
[46] 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*. Association for Computational Linguistics, 577–584.
[47] Pierre Zweigenbaum and Natalia Grabar. 2002. Restoring accents in unknown biomedical words: Application to the French MeSH thesaurus. *Int. J. Med. Inf.* 67, 1–3 (2002), 113–126.

Received February 2020; revised November 2020; accepted November 2020