A Multitask Learning Approach for Diacritic Restoration

Sawsan Alqahtani [1,2] and **Ajay Mishra**[1] and **Mona Diab**[2]
[1]AWS, Amazon AI
[2]The George Washington University
sawsa@amazon.com, misaja@amazon.com, diabmona@amazon.com

Abstract

In many languages like Arabic, diacritics are used to specify pronunciations as well as meanings. Such diacritics are often omitted in written text, increasing the number of possible pronunciations and meanings for a word. This results in a more ambiguous text making computational processing on such text more difficult. Diacritic restoration is the task of restoring missing diacritics in the written text. Most state-of-the-art diacritic restoration models are built on character level information which helps generalize the model to unseen data, but presumably lose useful information at the word level. Thus, to compensate for this loss, we investigate the use of multi-task learning to jointly optimize diacritic restoration with related NLP problems namely word segmentation, part-of-speech tagging, and syntactic diacritization. We use Arabic as a case study since it has sufficient data resources for tasks that we consider in our joint modeling. Our joint models significantly outperform the baselines and are comparable to the state-of-the-art models that are more complex relying on morphological analyzers and/or a lot more data (e.g. dialectal data).

1 Introduction

In contrast to English, some vowels in languages such as Arabic and Hebrew are not part of the alphabet and diacritics are used for vowel specification.[1] In addition to pertaining vowels, diacritics can also represent other features such as case marking and phonological gemination in Arabic. Not including diacritics in the written text in such languages increases the number of possible meanings as well as pronunciations. Humans rely on the surrounding context and their previous knowledge to infer the meanings and/or pronunciations of words. However, computational models, on the other hand, are inherently limited to deal with missing diacritics which pose a challenge for such models due to increased ambiguity.

Diacritic restoration (or diacritization) is the process of restoring these missing diacritics for every character in the written texts. It can specify pronunciation and can be viewed as a relaxed variant of word sense disambiguation. For example, the Arabic word علم *Elm*[2] can mean "flag" or "knowledge", but the meaning as well as pronunciation is specified when the word is diacritized (عَلَمٌ ***Ealamu*** means "flag" while عِلْمٌ ***Eilomo*** means "knowledge"). As an illustrative example in English, if we omit the vowels in the word *pn*, the word can be read as *pan*, *pin*, *pun*, and *pen*, each of these variants have different pronunciations and meanings if it composes a valid word in the language.

The state-of-the-art diacritic restoration models reached a decent performance over the years using recurrent or convolutional neural networks in terms of accuracy (Zalmout and Habash, 2017; Alqahtani et al., 2019; Orife, 2018) and/or efficiency (Alqahtani et al., 2019; Orife, 2018); yet, there is still room for further improvements. Most of these models are built on character level information which help generalize the model to unseen data, but presumably lose some useful information at the word level. Since word level resources are insufficient to be relied upon for training diacritic restoration models, we integrate additional linguistic information that considers word morphology as well as word relationships within a sentence to partially compensate for this loss.

In this paper, we improve the performance of

[1]Diacritics are marks that are added above, below, or in-between the letters to compose a new letter or characterize the letter with a different sound (Wells, 2000).

[2]We use Buckwalter Transliteration encoding http://www.qamus.org/transliteration.htm.

diacritic restoration by building a multitask learning model (i.e. joint modeling). Multitask learning refers to models that learn more than one task at the same time, and has recently been shown to provide good solutions for a number of NLP tasks (Hashimoto et al., 2016; Kendall et al., 2018).

The use of a multitask learning approach provides an end-to-end solution, in contrast to generating the linguistic features for diacritic restoration as a preprocessing step. In addition, it alleviates the reliance on other computational and/or data resources to generate these features. Furthermore, the proposed model is flexible such that a task can be added or removed depending on the data availability. This makes the model adaptable to other languages and dialects.

We consider the following auxiliary tasks to boost the performance of diacritic restoration: word segmentation, part-of-speech (POS) tagging, and syntactic diacritization. We use Arabic as a case study for our approach since it has sufficient data resources for tasks that we consider in our joint modeling.[3]

The contributions of this paper are twofold:

1. We investigate the benefits of automatically learning related tasks to boost the performance of diacritic restoration;
2. In doing so, we devise a state-of-the-art model for Arabic diacritic restoration as well as a framework for improving diacritic restoration in other languages that include diacritics.

2 Diacritization and Auxiliary Tasks

We formulate the problem of *(full)* diacritic restoration (*DIAC*) as follows: given a sequence of characters, we identify the diacritic corresponding to each character in that sequence from the following set of diacritics {a, u, i, o, K, F, N, ∼, ∼a, ∼u, ∼i, ∼F, ∼K, and ∼N}. We additionally consider three auxiliary tasks: syntactic diacritization, part-of-speech tagging, and word segmentation. Two of which operate at the word level (syntactic diacritization and POS tagging) and the remaining tasks (diacritic restoration and word segmentation) operate at the character level. This helps diacritic restoration utilize information from both character and word level information, bridging the gap between the two levels.

[3]Other languages that include diacritics lack such resources; however, the same multitask learning framework can be applied if data resources become available.

Syntactic Diacritization *(SYN)*: This refers to the task of retrieving diacritics related to the syntactic positions for each word in the sentence, which is a sub-task of full diacritic restoration. Arabic is a templatic language where words comprise roots and patterns in which patterns are typically reflective of diacritic distributions. Verb patterns are more or less predictable however nouns tend to be more complex. Arabic diacritics can be divided into lexical and inflectional (or syntactic) diacritics. Lexical diacritics change the meanings of words as well as their pronunciations and their distribution is bound by patterns/templates. In contrast, inflectional diacritics are related to the syntactic positions of words in the sentence and are added to the last letter of the main morphemes of words (word finally), changing their pronunciations.[4] Inflectional diacritics are also affected by word's root (e.g. weak roots) and semantic or morphological properties (e.g. with the same grammatical case, masculine and feminine plurals take different diacritics).

Thus, the same word can be assigned a different syntactic diacritic reflecting syntactic case, i.e. depending on its relations to the remaining words in the sentence (e.g. subject or object). For example, the diacritized variants عَلَمَ Ealam**a** and عَلَمُ Ealam**u** which both mean "flag" have the corresponding syntactic diacritics: **a** and **u**, respectively. That being said, the main trigger for accurate syntactic prediction is the relationships between words, capturing semantic and most importantly, syntactic information.

Because Arabic has a unique set of diacritics, this study formulates syntactic diacritization in the following way: each word in the input is tagged with a single diacritic representing its syntactic position in the sentence.[5] The set of diacritics in syntactic diacritization is the same as the set of diacritics for full diacritic restoration. Other languages that include diacritics can include syntactic related diacritics but in a different manner and complexity compared to Arabic.

[4]Diacritics that are added due to passivization are also syntactic in nature but are not considered in our syntactic diacritization task. That said, they are still considered in the full diacritic restoration model.

[5]Combinations of diacritics is possible but we combine valid possibilities together as one single unit in our model. For example, the diacritics ∼ and **a** are combined to form an additional diacritic ∼**a**.

Word segmentation *(SEG)*: This refers to the process of separating affixes from the main unit of the word. Word segmentation is commonly used as a preprocessing step for different NLP applications and its usefulness is apparent in morphologically rich languages. For example, the undiacritized word *whm* وهم might be diacritized as *waham~a* وَهَمَّ "and concerned", *waham* وَهَم "illusion", where the first diacritized word consists of two segments "wa ham~a" وَ هَمَّ while the second is composed of one word. Word segmentation can be formulated in the following way: each character in the input is tagged following IOB tagging scheme (*B*: beginning of a segment; *I*: inside a segment; *O*: out of the segment) (Diab et al., 2004).

Part-Of-Speech Tagging *(POS)*: This refers to the task of determining the syntactic role of a word (i.e. part of speech) within a sentence. POS tags are highly correlated with diacritics (both syntactic and lexical): knowing one helps determine or reduce the possible choices of the other. For instance, the word كتب *ktb* in the sentence *ktb [someone]* means "books" if we know it to be a noun whereas the word would be either كَتَب *katab* "someone wrote" or كَتَّب *kat~ab* "made someone write" if it is known to be a verb.

POS tagging can be formulated in the following way: each word in the input is assigned a POS tag from the Universal Dependencies tagset (Taji et al., 2017).[6]

3 Approach

We built a diacritic restoration joint model and studied the extent to which sharing information is plausible to improve diacritic restoration performance. Our joint model is motivated by the recent success of the hierarchical modeling proposed in (Hashimoto et al., 2016) such that information learned from an auxiliary task is passed as input to the diacritic restoration related layers.[7]

[6]Refer to https://universaldependencies.org/. This tagset is chosen because it includes essential POS tags in the language, and it is unified across different languages which makes it suitable to investigate more languages in the future.

[7]We also experimented with learning tasks sharing some levels and then diverging to specific layers for each tasks. However, this did not improve the performance compared to the diacritic restoration model when we don't consider any additional task.

3.1 Input Representation

Since our joint model may involve both character and word level based tasks, we began our investigation by asking the following question: *how to integrate information between these two levels?* Starting from the randomly initialized character embeddings as well as a pretrained set of embeddings for words, we follow two approaches (Figure 1 visually illustrates the two approaches with an example).

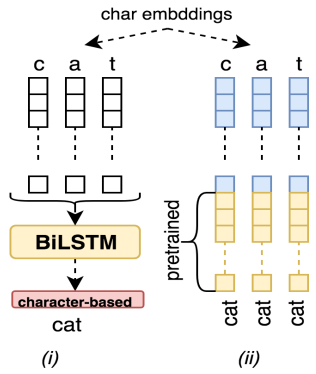

Figure 1: An example of embedding vectors for the word *cat* and its individual characters: c,a, and t. (i) A character-based representation for the word *cat* from its individual characters; (ii) A concatenation for the word embedding with each of its individual characters.

***(1) Character Based Representation*:** We pass information learned by character level tasks into word level tasks by composing a word embedding from the word's characters. We first concatenate the individual embeddings of characters in that word, and then apply a Bidirectional Long Short Term Memory (BiLSTM) layer to generate denser vectors.[8] This helps representing morphology and word composition into the model.

***(2) Word-To-Character Representation*:** To pass information learned by word level tasks into character level tasks, we concatenate each word with each of its composed characters during each pass, similar to what is described in Watson et al. (2018)'s study. This helps distinguishing the individual characters based on the surrounding context, implicitly capturing additional semantic and syntactic information.

[8]We also evaluated the use of a feedforward layer and unidirectional Long Short Term Memory (LSTM) but a BiLSTM layer yielded better results.

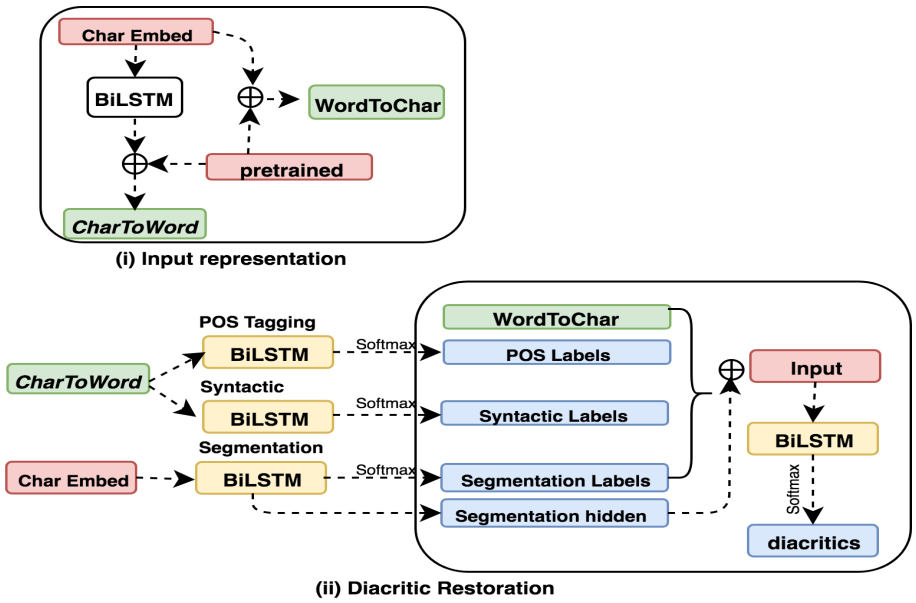

Figure 2: The diacritic restoration joint model. All *Char Embed* entities refer to the same randomly initialized character embedding learned during the training process. *Pretrained* embeddings refer to fixed word embeddings obtained from fastText (Bojanowski et al., 2017). *(i)* shows the input representation for CharToWord and WordToChar embedding which is the same as in Figure 1. *(ii)* represents the diacritic restoration joint model; output labels from each task are concatenated with WordToChar embedding and optionally with segmentation hidden.

3.2 The Joint Model

For all architectures, the main component is BiLSTM (Hochreiter and Schmidhuber, 1997; Schuster and Paliwal, 1997), which preserves the temporal order of the sequence and has been shown to provide the state-of-the-art performance in terms of accuracy (Zalmout and Habash, 2017; Alqahtani et al., 2019). After representing characters through random initialization and representing words using pretrained embeddings obtained from fastText (Bojanowski et al., 2017), the learning process for each batch runs as follows:

1. We extract the two additional input representation described in Section 3.1;
2. We apply BiLSTM for each of the different tasks separately to obtain their corresponding outputs;
3. We pass all outputs from all tasks as well as WordToChar embedding vectors as input to the diacritic restoration model and obtain our diacritic outputs.

Figure 2 illustrates the diacritic restoration joint model. As can be seen, SYN as well as POS tagging are trained on top of CharToWord representation which is basically the concatenation of the pretrained embedding for each word with the character-based representations described in Figure 1. SEG is also trained separately on top of the character embeddings. We pass the outputs of all these tasks along with WordToChar representation to train the BiLSTM diacritic restoration model. Omitting a task is rather easy, we just remove the related components for that task to yield the appropriate model. We optionally pass the last hidden layer for SEG along with the remaining input to the diacritic restoration model.[9]

4 Experimental Setups

Dataset: We use the Arabic Treebank (ATB) dataset: parts 1, 2, and 3 and follow the same data division as Diab et al. (2013). Table 1 illustrates the data statistics. For word based tasks, we segment each sentence into space tokenized words. For character based tasks, we, in addition, add the special boundary "<w>" between these words, and then each word is further segmented into its characters, similar to that in (Alqahtani et al., 2019). We pass each word through the model along with a specific number of previous and future words (+/- 10 words).

Parameter Settings: For all tasks, we use 250 hidden units in each direction (500 units in both directions combined) and 300 as embedding size. We use 3 hidden layers for tasks except in SEG in

[9]Passing the last hidden layer for POS tagging and/or SYN did not improve the performance; the pretrained embeddings are sufficient to capture important linguistic signals.

Train	Test	Dev	OOV
502,938	63,168	63,126	7.3%

Table 1: Number of words and out of vocabulary (OOV) rate for Arabic. OOV rate indicates the percentage of undiacritized words in the test set that have not been observed during training.

which we use only one layer. We use Adam for learning optimization with a learning rate of 0.001. We use 20 for epoch size, 16 for batch size, 0.3 for hidden dropout, and 0.5 for embedding dropout. We initialize the embedding with a uniform distribution [-0.1,0.1] and the hidden layers with normal distribution. The loss scores for all considered tasks are combined and then normalized by the number of tasks in the model.

Evaluation metrics: We use accuracy for all tasks except diacritic restoration. For diacritic restoration, the two most typically used metrics are Word Error Rate (WER) and Diacritic Error Rate (DER), the percentages of incorrectly diacritized words and characters, respectively. In order to approximate errors in the syntactic diacritics, we use Last Diacritic Error Rate (LER), the percentage of words that have incorrect diacritics in the last positions of words. To evaluate the models' ability to generalize beyond observed data, we compute WER on OOV (out-of-vocabulary) words.[10]

Significance testing: We ran each experiment three times and reported the mean score.[11] We used the t-test with $p = 0.05$ to evaluate whether the difference between models' performance and the diacritic restoration is significant (Dror et al., 2018).

5 Results and Analysis

Table 2 shows the performance of joint diacritic restoration models when different tasks are considered. When we consider WordToChar as input to the diacritic restoration model, we observe statistically significant improvements for all evaluation metrics. This is justified by the ability of word embeddings to capture syntactic and semantic information at the sentence level. The same character is disambiguated in terms of the surrounding context as well as the word it appears in (e.g. the character *t* in the word *cat* would be represented slightly different than *t* in a related word *cats* or even a different word *table*). We consider both character based model as well as WordToChar based model as our baselines (BASE).

We use WordToChar representation rather than characters for all remaining models that jointly learn more than one task. For all experiments, we observe improvements compared to both baselines across all evaluation metrics. Furthermore, all models except DIAC+SEG outperform WordToChar diacritic restoration model in terms of WER, showing the benefits of considering output distributions for the other tasks. Despite leveraging tasks focused on syntax (SYN/POS) or morpheme boundaries (SEG), the improvements extend to lexical diacritics as well. Thus, the proposed joint diacritic restoration model is also helpful in settings beyond word final syntactic related diacritics. The best performance is achieved when we consider all auxiliary tasks within the diacritic restoration model.

Impact of Auxiliary Tasks: We discuss the impact of adding each investigated task towards the performance of the diacritic restoration model.

Word segmentation (DIAC+SEG): When morpheme boundaries as well as diacritics are learned jointly, the WER performance is slightly reduced on all and OOV words. This reduction is attributed mostly to lexical diacritics. As Arabic exhibits a non-concatenative fusional morphology, reducing its complexity to a segmentation task might inherently obscure morphological processes for each form.

Observing only slight improvement is surprising; we believe that this is due to our experimental setup and does not negate the importance of having morphemes that assign the appropriate diacritics. We speculate that the reason for this is that we do not capture the interaction between morphemes as an entity, losing some level of morphological information.

For instances, the words ***waham***$\sim a$ versus ***wahum*** for the undiacritized words ***whm*** (bold letters refer to consonants distinguishing it from diacritics) would benefit from morpheme boundary identifications to tease apart ***wa*** from ***hum*** in the second variant (***wahum***), emphasizing that these are two words. But on the other hand, it adds an

[10] Words that appear in the training dataset but do not appear in the test dataset.

[11] Higher number of experiments provide more robust conclusion about the models' performance. We only considered the minimum acceptable number of times to run each experiment due to limited computational resources.

Task	WER	DER	LER/Lex	OOV WER
Zalmout and Habash (2017)	8.21	-	-	**20.2**
Zalmout and Habash (2019a)	**7.50**	-	-	-
Alqahtani and Diab (2019a)	7.6	2.7	-	32.1
BASE (Char)	8.51 (±0.01)	2.80	5.20/5.54	34.56
BASE (WordToChar)	8.09 (±0.05)	2.73	5.00/5.30	32.10
DIAC+SEG	8.35 (±0.02)	2.82	5.20/5.46	33.97
DIAC+SYN	7.70* (±0.02)	2.60	4.72/5.08	**30.94**
DIAC+POS	7.86* (±0.14)	2.65	4.72/5.20	32.28
DIAC+SEG+SYN	7.70* (±0.05)	2.59	4.65/5.03	31.33
DIAC+SEG+POS	7.73* (±0.08)	2.62	4.73/5.01	31.31
DIAC+SYN+POS	7.72* (±0.06)	2.61	4.62/5.06	31.05
ALL	**7.51*** (±0.09)	**2.54**	**4.54/4.91**	31.07

Table 2: Performance of the joint diacritic restoration model when different related tasks are considered. **Bold** numbers represent the highest score per column. Almost all scores are higher than the base model *BASE (char)*. * denotes statistically significant improvements compared to the baselines. *Lex* refers to the percentage of words that have incorrect lexical diacritics only, excluding syntactic diacritics.

additional layer of ambiguity for other cases like the morpheme ***ktb*** in the diacritic variants ***kataba***, ***kutubu***, *saya**kotubo*** - note that the underlined segment has the same consonants as the other variants - in which identifying morphemes increased the number of possible diacritic variants without learning the interactions between adjacent morphemes.

Furthermore, we found inconsistencies in the dataset for morphemes which might cause the drop in performance when we only consider SEG. When we consider all tasks together, these inconsistencies are reduced because of the combined information from different linguistic signals towards improving the performance of the diacritic restoration model.

Syntactic diacritization (DIAC+SYN): By enforcing inflectional diacritics through an additional focused layer within the diacritic restoration model, we observe improvements on WER compared to the baselines. We notice improvements on syntactic related diacritics (LER score), which is expected given the nature of syntactic diacritization in which it learns the underlying syntactic structure to assign the appropriate syntactic diacritics for each word. Improvements also extend to lexical diacritics, and this is because word relationships are captured during learning syntactic diacritics in which BiLSTM modeling for words is integrated.

POS tagging (DIAC+POS): When we jointly train POS tagging with full diacritic restoration, we notice improvements compared to both baselines. Compared to syntactic diacritization, we obtain *similar* findings across all evaluation metrics except for WER on OOV words in which POS tagging drops. Including POS tagging within diacritic restoration also captures important information about the words; the idea of POS tagging is to learn the underlying syntax of the sentence. In comparison to syntactic diacritization, it involves different types of information like passivization which could be essential in learning correct diacritics.

Ablation Analysis: Incorporating all the auxiliary tasks under study within the diacritic restoration model (ALL) provides the best performance across all measures except WER on OOV words in which the best performance was given by DIAC+SYN. We discuss the impact of removing one task at a time from ALL and examine whether its exclusion significantly impacts the performance. Excluding SEG from the process drops the performance of diacritic restoration. This shows that even though SEG did not help greatly when it was combined solely with diacritic restoration, the combinations of SEG and the other word based tasks filled in the gaps that were missing from just identifying morpheme boundaries. Excluding either POS tagging or syntactic diacritization also hurts the performance which shows that these tasks complement each other and, taken together, they improve the performance of diacritic restoration model.

Input Representation:

Impact of output labels: Table 3 shows the different models when we do not pass the labels of the investigated tasks (the input is only WordToChar representation) against the same models when we do. We noticed a drop in performance

across all models. Notice that all models - even when we do not consider the label have better performance than the baselines. This also supports the benefits of WordToChar representation.

Tasks	With Labels	Without Labels
DIAC+SYN	**7.70**	7.99
DIAC+POS	**7.86**	7.93
DIAC+SEG+SYN	**7.70**	7.93
DIAC+SEG+POS	**7.73**	7.99
DIAC+SYN+POS	**7.72**	7.97
ALL	**7.51**	7.91

Table 3: WER performance when we do not consider the output labels for the investigated tasks. **Bold** numbers represent the highest score per row.

Last hidden layer of SEG: Identifying morpheme boundaries did not increase accuracy as we expected. Therefore, we examined whether information learned from the BiLSTM layer would help us learn morpheme interactions by passing the output of last BiLSTM layer to the diacritic restoration model along with segmentation labels. We did not observe any improvements towards predicting accurate diacritics when we pass information regarding the last BiLSTM layer. For ALL, the WER score increased by 0.22%. Thus, it is sufficient to only utilize the segment labels for diacritic restoration.

Passive and active verbs: Passivation in Arabic is denoted through diacritics and missing such diacritic can cause ambiguity in some cases (Hermena et al., 2015; Diab et al., 2007). To examine its impact, we further divide verbs in the POS tagset into passive and active, increasing the size by one. Table 4 shows the diacritic restoration performance with and without considering passivation. We notice improvements, in some combinations of tasks, across all evaluation metrics compared to the pure POS tagging, showing its importance in diacritic restoration models.

Task	**With Pass**	**Without Pass**
DIAC+POS	**7.65**	7.86
DIAC+SEG+POS	**7.65**	7.73
DIAC+SYN+POS	7.78	**7.72**
ALL	7.62	**7.51**

Table 4: WER performance for different diacritic restoration models when passivation is considered. **Bold** numbers represent the highest score per row.

Level of linguistic information: The joint diacritic restoration model were built empirically and tested against the development set. We noticed that to improve the performance, soft parameter sharing in a hierarchical fashion performs better on diacritic restoration. We experimented with building a joint diacritic restoration model that jointly learns segmentation and diacritics through hard parameter sharing. To learn segmentation with diacritic restoration, we shared the embedding layer between the two tasks as well as sharing some or all layers of BiLSTM. We got WER on all words (8.53∼9.35) in which no improvements were shown compared to character based diacritic restoration. To learn word based tasks with diacritic restoration, we pass WordToChar representation to the diacritic restoration and/or CharToWord representation for word-based tasks. The best that we could get for both tasks is 8.23%∼9.6%; no statistically significant improvements were found. This shows the importance of hierarchical structure for appropriate diacritic assignments.

Qualitative analysis: We compared random errors that are correct in DIAC (character-based diacritic restoration) with ALL in which we consider all investigated tasks. Although ALL provides accurate results for more words, it introduces errors in other words that have been correctly diacritized by DIAC. The patterns of such words are not clear. We did not find a particular category that occurs in one model but not the other. Rather, the types and quantity of errors differ in each of these categories.

State-of-the-art Comparison: Table 2 also shows the performance of the state-of-the-art models. ALL model surpass the performance of Zalmout and Habash (2017). However, Zalmout and Habash (2017)'s model performs significantly better on OOV words. Zalmout and Habash (2019a) provides comparable performance to ALL model. The difference between their work and that in (Zalmout and Habash, 2017) is the use of a joint model to learn morphological features other than diacritics (or features at the word level), rather than learning these features individually. Zalmout and Habash (2019a) obtained an additional boost in performance (0.3% improvement over ours) when they add a dialect variant of Arabic in the learning process, sharing information between both languages.

Alqahtani and Diab (2019a) provides comparable performance to ALL and better performance on some task combinations in terms of WER on all and OOV words. The difference between their model and our BASE model is the addition of a

CRF (Conditional Random Fields) layer which incorporate dependencies in the output space at the cost of model's computational efficiency (memory and speed).

Zalmout and Habash (2019b) provides the current state-of-the-art performance in which they build a morphological disambiguation framework in Arabic similar to (Zalmout and Habash, 2017, 2019a). They reported their scores based on the development set which was not used for tuning. In the development set, they obtained 93.9% which significantly outperforms our best model (ALL) by 1.4%. Our approach is similar to (Zalmout and Habash, 2019b). We both follow WordToChar as well as CharToWord input representations discussed in Section 3.1, regardless of the specifics. Furthermore, we both consider the morphological outputs as features in our diacritic restoration model. In Zalmout and Habash (2019b), morphological feature space that are considered is larger, making use of all morphological features in Arabic. Furthermore, Zalmout and Habash (2019b) use sequence-to-sequence modeling rather than sequence classification as ours. Unlike Zalmout and Habash (2019b), our model is more flexible allowing additional tasks to be added when sufficient resources are available.

We believe that neither the underlying architecture nor the consideration of all possible features were the *crucial* factor that led to the significant reduction in WER performance. Rather, morphological analyzers is crucial in such significant improvement. As a matter of fact, in Zalmout and Habash (2019b), the performance significantly drops to 7.2 when they, similar to our approach, take the highest probabilistic value as a solution. Thus, we believe that the use of morphological analyzers enforces valid word composition in the language and filter out invalid words (a side effect of using characters as input representation). This also justifies the significant improvement on OOV words obtained by (Zalmout and Habash, 2017). Thus, we believe that a global knowledge of words and internal constraints within words are captured.

Auxiliary tasks: We compared the base model of the auxiliary tasks to the state-of-the-art (SOTA). For SEG, BiLSTM model has comparable performance to that in (Zalmout and Habash, 2017) (SEG yields 99.88% F1 compared to SOTA 99.6%). For POS, we use a shallower tag set (16 number of tags compared to $\sim$70) than typically used in previous models hence we do not have a valid comparison set. For SYN, we compare our results with (Hifny, 2018) which uses a hybrid network of BiLSTM and Maximum Entropy to solve syntactic diacritization. The SYN yields results comparable to SOTA (our model performs 94.22 vs. SOTA 94.70).

6 Related Work

The problem of diacritization has been addressed using classical machine learning approaches (e.g. Maximum Entropy and Support Vector Machine) (Zitouni and Sarikaya, 2009; Pasha et al., 2014) or neural based approaches for different languages that include diacritics such as Arabic, Vietnamese, and Yoruba. Neural based approaches yield state-of-the-art performance for diacritic restoration by using Bidirectional LSTM or temporal convolutional networks (Zalmout and Habash, 2017; Orife, 2018; Alqahtani et al., 2019; Alqahtani and Diab, 2019a).

Arabic syntactic diacritization has been consistently reported to be difficult, degrading the performance of full diacritic restoration (Zitouni et al., 2006; Habash et al., 2007; Said et al., 2013; Shaalan et al., 2009; Shahrour et al., 2015; Darwish et al., 2017). To improve the performance of syntactic diacritization or full diacritic restoration in general, previous studies followed different approaches. Some studies separate lexical from syntactic diacritization (Shaalan et al., 2009; Darwish et al., 2017). Other studies consider additional linguistic features such as POS tags and word segmentation (i.e. tokens or morphemes) (Ananthakrishnan et al., 2005; Zitouni et al., 2006; Zitouni and Sarikaya, 2009; Shaalan et al., 2009).

Hifny (2018) addresses syntactic diacritization by building BiLSTM model in which its input embeddings are augmented with manually generated features of context, POS tags, and word segments. Rashwan et al. (2015) use deep belief network to build a diacritization model for Arabic that focuses on improving syntactic diacritization and build sub-classifiers based on the analysis of a confusion matrix and POS tags.

Regarding incorporating linguistic features into the model, previous studies have either used morphological features as a preprocessing step or as a ranking step for building diacritic restoration models. As a preprocessing step, the words are converted to their constituents (e.g. morphemes, lemmas, or n-grams) and then diacritic restoration

models are built on top of that (Ananthakrishnan et al., 2005; Alqahtani and Diab, 2019b). Ananthakrishnan et al. (2005) use POS tags to improve diacritic restoration at the syntax level assuming that POS tags are known at inference time.

As a ranking procedure, all possible analyses of words are generated and then the most probable analysis is chosen (Pasha et al., 2014; Zalmout and Habash, 2017, 2019a,b). Zalmout and Habash (2017) develop a morphological disambiguation model to determine Arabic morphological features including diacritization. They train the model using BiLSTM and consult with a LSTM-based language model as well as other morphological features to rank and score the output analysis. Similar methodology can be found in (Pasha et al., 2014) but using Support Vector Machines. This methodology shows better performance on out of vocabulary (OOV) words compared to pure character models.

7 Discussion & Conclusion

We present a diacritic restoration joint model that considers the output distributions for different related tasks to improve the performance of diacritic restoration. Our results shows statistically significant improvements across all evaluation metrics. This shows the importance of considering additional linguistic information at morphological and/or sentence levels. Including semantic information through pretrained word embeddings within the diacritic restoration model also helped boosting the diacritic restoration performance. Although we apply our joint model on Arabic, this model provides a framework for other languages that include diacritics whenever resources become available. Although we observed improvements in terms of generalizing beyond observed data when using the proposed linguistic features, the OOV performance is still an issue for diacritic restoration.

References

Sawsan Alqahtani and Mona Diab. 2019a. Investigating input and output units in diacritic restoration. In *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*. IEEE.

Sawsan Alqahtani and Mona Diab. 2019b. Investigating input and output units in diacritic restoration. In *2019 18th IEEE International Conference on Machine Learning and Applications (ICMLA)*.

Sawsan Alqahtani, Ajay Mishra, and Mona Diab. 2019. Convolutional neural networks for diacritic restoration. In *EMNLP*.

Sankaranarayanan Ananthakrishnan, Shrikanth Narayanan, and Srinivas Bangalore. 2005. Automatic diacritization of arabic transcripts for automatic speech recognition. In *Proceedings of the 4th International Conference on Natural Language Processing*, pages 47–54.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*.

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.

Mona Diab, Mahmoud Ghoneim, and Nizar Habash. 2007. Arabic diacritization in the context of statistical machine translation. In *Proceedings of MT-Summit*.

Mona Diab, Nizar Habash, Owen Rambow, and Ryan Roth. 2013. Ldc arabic treebanks and associated corpora: Data divisions manual. *arXiv preprint arXiv:1309.5652*.

Mona Diab, Kadri Hacioglu, and Daniel Jurafsky. 2004. Automatic tagging of arabic text: From raw text to base phrase chunks. In *Proceedings of HLT-NAACL 2004: Short papers*. Association for Computational Linguistics.

Rotem Dror, Gili Baumer, Segev Shlomov, and Roi Reichart. 2018. The hitchhikers guide to testing statistical significance in natural language processing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 1383–1392.

Nizar Habash, Ryan Gabbard, Owen Rambow, Seth Kulick, and Mitch Marcus. 2007. Determining case in arabic: Learning complex linguistic behavior requires complex linguistic features. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*.

Kazuma Hashimoto, Caiming Xiong, Yoshimasa Tsuruoka, and Richard Socher. 2016. A joint many-task model: Growing a neural network for multiple nlp tasks. *arXiv preprint arXiv:1611.01587*.

Ehab W Hermena, Denis Drieghe, Sam Hellmuth, and Simon P Liversedge. 2015. Processing of arabic diacritical marks: Phonological–syntactic disambiguation of homographic verbs and visual crowding effects. *Journal of Experimental Psychology: Human Perception and Performance*, 41(2):494.

Yasser Hifny. 2018. Hybrid lstm/maxent networks for arabic syntactic diacritics restoration. *IEEE Signal Processing Letters*, 25(10):1515–1519.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Alex Kendall, Yarin Gal, and Roberto Cipolla. 2018. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7482–7491.

Iroro Orife. 2018. Attentive sequence-to-sequence learning for diacritic restoration of yor\ub\'a language text. *arXiv preprint arXiv:1804.00832*.

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 (TASLP)*, 23(3):505–516.

Ahmed Said, Mohamed El-Sharqwi, Achraf Chalabi, and Eslam Kamal. 2013. A hybrid approach for arabic diacritization. In *International Conference on Application of Natural Language to Information Systems*, pages 53–64. Springer.

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.

Khaled Shaalan, Hitham M Abo Bakr, and Ibrahim Ziedan. 2009. A hybrid approach for building arabic diacritizer. In *Proceedings of the EACL 2009 workshop on computational approaches to semitic languages*, pages 27–35. Association for Computational Linguistics.

Anas Shahrour, Salam Khalifa, and Nizar Habash. 2015. Improving arabic diacritization through syntactic analysis. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1309–1315.

Dima Taji, Nizar Habash, and Daniel Zeman. 2017. Universal dependencies for arabic. In *Proceedings of the Third Arabic Natural Language Processing Workshop*, pages 166–176.

Daniel Watson, Nasser Zalmout, and Nizar Habash. 2018. Utilizing character and word embeddings for text normalization with sequence-to-sequence models. *arXiv preprint arXiv:1809.01534*.

JC Wells. 2000. Orthographic diacritics and multilingual computing. *Language problems and language planning*, 24(3):249–272.

Nasser Zalmout and Nizar Habash. 2017. Don't throw those morphological analyzers away just yet: Neural morphological disambiguation for arabic. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 704–713.

Nasser Zalmout and Nizar Habash. 2019a. Adversarial multitask learning for joint multi-feature and multi-dialect morphological modeling. *arXiv preprint arXiv:1910.12702*.

Nasser Zalmout and Nizar Habash. 2019b. Joint diacritization, lemmatization, normalization, and fine-grained morphological tagging. *arXiv preprint arXiv:1910.02267*.

Imed Zitouni and Ruhi Sarikaya. 2009. Arabic diacritic restoration approach based on maximum entropy models. *Computer Speech & Language*, 23(3):257–276.

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