Joint Diacritization, Lemmatization, Normalization, and Fine-Grained Morphological Tagging

Nasser Zalmout and **Nizar Habash**
Computational Approaches to Modeling Language Lab
New York University Abu Dhabi
United Arab Emirates
{nasser.zalmout,nizar.habash}@nyu.edu

Abstract

The written forms of Semitic languages are both highly ambiguous and morphologically rich: a word can have multiple interpretations and is one of many inflected forms of the same concept or lemma. This is further exacerbated for dialectal content, which is more prone to noise and lacks a standard orthography. The morphological features can be lexicalized, like lemmas and diacritized forms, or non-lexicalized, like gender, number, and part-of-speech tags, among others. Joint modeling of the lexicalized and non-lexicalized features can identify more intricate morphological patterns, which provide better context modeling, and further disambiguate ambiguous lexical choices. However, the different modeling granularity can make joint modeling more difficult. Our approach models the different features jointly, whether lexicalized (on the character-level), or non-lexicalized (on the word-level). We use Arabic as a test case, and achieve state-of-the-art results for Modern Standard Arabic with 20% relative error reduction, and Egyptian Arabic with 11% relative error reduction.

1 Introduction

Morphological modeling in Semitic languages is challenging. Their optional short vowels (diacritics) increase the overall ambiguity of surface forms; and their morphological richness results in large target spaces, which increase model sparsity. The different morphological features can be modeled through combined feature tags, using a single (but very large) target space, or through having separate models for each of the features. The combined features approach models the relationships between the different features explicitly, but the large target spaces for morphologically rich languages further increase sparsity. On the other hand, separate feature modeling guarantees smaller target spaces for the individual features, but the hard separation between the features prevents modeling any inter-feature dependencies. The set of morphological features includes lexicalized and non-lexicalized features, which further exacerbates joint modeling. Non-lexicalized features, like gender, and number, among others, have limited target spaces, and usually modeled as tagging tasks. Lexicalized features, like lemmas and diacritized forms (for Semitic languages), are open-ended, with large target vocabularies. Moreover, non-lexicalized features are modeled on the word level, whereas lexicalized features are optimally modeled on the character level. This difference in the modeling granularity can be challenging for joint models.

In this paper we present a model for handling lexicalized and non-lexicalized features jointly. We use a sequence-to-sequence architecture, with different parameter sharing strategies at the encoder and decoder sides for the different features. The non-lexicalized features are handled with a tagger, which shares several parameters with the encoder, and uses a multitask-learning architecture to model the different non-lexicalized features jointly. The lexicalized features, on the other hand, are handled with a specific decoder for each feature, sharing the same encoder. Our architecture models the non-lexicalized features on the word level, with a context representation that spans the entire sentence. The lexicalized features are modeled on the character level, with a fixed character context window. The character level modeling is also suitable for surface form normalization, which is important for noisy texts common in dialectal content.

We use Modern Standard Arabic (MSA) and Egyptian Arabic (EGY) as test cases. Our joint model achieves 20% relative error reduction (1.9% absolute improvement) for MSA, and 11% relative error reduction (2.5% absolute improvement) for EGY, compared to a baseline that models the different morphological features separately.

The rest of the paper is structured as follows. We present a brief background and a survey of related work in Section 2. We introduce the approach and various models in Section 3, and discuss the experimental setup and results in Section 4. We conclude and provide some directions for future work in Section 5.

2 Background and Related Work

In this section we present a brief linguistic overview of the challenges facing morphological modeling in Semitic and morphologically rich languages. We then discuss related contributions in literature, and how our model compares to them.

2.1 Linguistic Introduction

Morphologically rich languages (MRLs) tend to have more fully inflected words than other languages, realized through many morphemes that represent several morphological features. The target space for the combined morphological features therefore tends to be large, which increases sparsity. MRLs also can be highly ambiguous, with different interpretations of the same surface forms. Ambiguity is further exacerbated for Semitic languages, like Arabic and Hebrew, at which the short vowels (diacritics) can be kept or dropped. The high degree of ambiguity in Arabic results in having about 12 analyses per word on average (Pasha et al., 2014).[1]

Both morphological richness and ambiguity can be modeled with *morphological analyzers*, or morphological dictionaries, which are used to encode all potential word inflections in the language. Morphological analyzers should ideally return all the possible analyses of a surface word (to model ambiguity), and cover all the inflected forms of a word lemma (to model morphological richness), covering all related features. The best analysis can then be chosen through *morphological disambiguation*; by predicting the different morphological feature values and use them to rank the relevant analyses from the analyzer. The morphological features that we model for Arabic include:

- Lexicalized features: lemmas (lex) and diacritized forms (diac).
- Non-lexicalized features: aspect (asp), case (cas), gender (gen), person (per), part-of-speech (POS), number (num), mood (mod), state (stt), voice (vox).
- Clitics: enclitics, like pronominal enclitics, negative particle enclitics; proclitics, like article proclitic, preposition proclitics, conjunction proclitics, question proclitics.

Table 1 shows an example highlighting the different morphological features. The example presents a subset of the possible analyses for the word لمتهم *lmthm*.[2] Disambiguation using the non-lexicalized features only is not conclusive enough, as we see in the last two analyses, where the lemma and diacritization only can disambiguate the right analysis.

Dialectal Arabic (DA) includes several dialects of Arabic, like EGY, that vary by the geographical location in the Arab world. DA is also Semitic and an MRL, but it is mainly spoken, and lacks a standard orthography (Habash et al., 2012a). The lack of a standard orthography further increases sparsity and ambiguity, hence requiring explicit normalization. Habash et al. (2012a, 2018) proposed CODA, a Conventional Orthography for Dialectal Arabic, which aims to provide a conventionalized orthography across the various Arabic dialects. We use CODA as the reference for the normalization task.

2.2 Morphological Tagging

Arabic morphological tagging and disambiguation have been studied extensively in literature, with contributions for MSA (Khalifa et al., 2016; Abdelali et al., 2016; Habash and Rambow, 2005; Diab et al., 2004), and DA (Habash et al., 2013; Al-Sabbagh and Girju, 2012; Duh and Kirchhoff, 2005). There are also several recent contributions that showed significant accuracy improvement using deep learning models (Zalmout et al., 2018; Inoue et al., 2017; Zalmout and Habash, 2017; Heigold et al., 2016). In addition to other deep learning contributions that showed limited success for Arabic (Shen et al., 2016). Most of these contributions model the different morphological features separately, or focus on a limited feature subset. We elaborate on the contributions with some joint modeling aspects later in the section.

2.3 Diacritization and Lemmatization

Diacritization and lemmatization are very useful for tasks like information retrieval, machine translation, and text-to-speech, among others.

[1]For more information on Arabic natural language processing, see (Habash, 2010).

[2]Arabic transliteration is presented in the Habash-Soudi-Buckwalter scheme (Habash et al., 2007).

Diacrtization	**Lemma**	**English**	**POS**	**Prc3**	**Prc2**	**Prc1**	**Prc0**	**Per**	**Asp**	**Vox**	**Mod**	**Gen**	**Num**	**Stt**	**Cas**	**Enc0**
lam∼at.hum	lam∼	she collected them	verb	0	0	0	0	3	p	a	i	f	s	na	na	dobj_{3mp}
lum.tahum	lAm	you [m.s.] blamed them	verb	0	0	0	0	2	p	a	i	m	s	na	na	dobj_{3mp}
lum.tihim	lAm	you [f.s.] blamed them	verb	0	0	0	0	2	p	a	i	f	s	na	na	dobj_{3mp}
lum.tuhum	lAm	I blamed them	verb	0	0	0	0	1	p	a	i	m	s	na	na	dobj_{3mp}
lam∼atuhum	lam∼a$\hbar$	their collection	noun	0	0	0	0	na	na	na	na	f	s	c	n	poss_{3mp}
limut∼ahamĩ	mut∼aham	for a suspect	noun	0	0	li (prep)	0	na	na	na	na	m	s	i	g	0
limut∼ahimĩ	mut∼ahim	for an accuser	noun	0	0	li (prep)	0	na	na	na	na	m	s	i	g	0

Table 1: A subset of all the possible analyses for the word لمتهم *lmthm*. Notice that in the last two analyses the words are disambiguated through the lemmas and diacritized forms only, and they share all the other features.

Diacritization has generally been an active area of research (Darwish et al., 2017; Zitouni et al., 2006; Nelken and Shieber, 2005). More recent contributions use Deep Learning models in different configurations; Belinkov and Glass (2015) model diacritization as a classification task, using Long Short Term Memory (LSTM) cells. And Abandah et al. (2015) use LSTMs to model diacritization as a sequence transcription task, similar to Mubarak et al. (2019) who model diacritization as a sequence-to-sequence task.

Early contributions for lemmatization used finite state machines (Schmid et al., 2004; Minnen et al., 2001), which had a limited capacity for modeling unseen words or lemmas. There were also several contributions that utilize a joint tagging and lemmatization approach, using CRFs and Maximum Entropy models (Müller et al., 2015; Chrupala et al., 2008). Other contributions approached lemmatization as a lemma selection task (Ezeiza et al., 1998), where the goal is to select the correct lemma from a set of lemmas provided by a morphological analyzer. Many of the lemmatization models for Arabic use a similar approach (Pasha et al., 2014; Roth et al., 2008). More recently, sequence-to-sequence models with attention (Bahdanau et al., 2014) have been shown useful in several NLP tasks, with several lemmatization contributions (Malaviya et al., 2019; Bergmanis and Goldwater, 2018; Pütz et al., 2018). Other contributions use additional morphosyntactic features as part of the modeling architecture (Kanerva et al., 2019; Kondratyuk et al., 2018), somewhat similar to our approach.

2.4 Joint Morphological Modeling in Arabic

There are also several contributions for the joint modeling of the different morphological features in Arabic. However, most of these contributions use separate models for each of the features, and usually use a ranking step to select the best overall morphological analysis from an external morphological analyzer (Roth et al., 2008; Habash and Rambow, 2007). MADAMIRA (Pasha et al., 2014) is a popular system for Arabic morphological tagging and disambiguation. It uses SVMs for the different non-lexicalized features, and n-gram language models for the lemmas and diacritized forms. Zalmout and Habash (2017) presented a neural extension of this model, with LSTM taggers for the individual features, and neural language models for the lexicalized features. Inoue et al. (2017) used multi-task learning for fine-grained POS tagging, modeling the different morphological features jointly, but they do not model lemmas or diacritized forms. Zalmout and Habash (2019) also used multitask learning for the different non-lexicalized morphological features, and neural language models for lemmas and diacritized forms. This model currently provides state-of-the-art results for Arabic. In the models that rely on morphological analyzers (Zalmout and Habash, 2019, 2017; Pasha et al., 2014) surface form normalization are byproducts of selecting the correct analysis, rather than being explicitly modeled.

3 Approach

Non-lexicalized features are usually modeled on the word level, whereas lexicalized features are better handled through character level models. Moreover, the context representation for morphological tagging of the non-lexicalized features usually spans the entire sentence, using LSTMs for example. The optimal context representation for the lexicalized features, on the other hand, is through a fixed number of characters before and after the target word (Bergmanis and Goldwater, 2018). This difference in modeling granularity, in terms of context representation or word/character level modeling, can be very challenging for joint modeling.

We use a modified sequence-to-sequence architecture, where some components of the encoder are shared between a tagger, for the non-lexicalized features, and the encoder-decoder architecture, for the lexicalized features. We also use separate decoders for the different lexicalized features, that share the same encoder and trained jointly using a shared loss function. The remainder of this section discusses the architecture in more detail.

3.1 Tagger

The tagging architecture is similar to the architecture presented by Zalmout and Habash (2019). We use two Bi-LSTM layers on the word level to model the context for each direction of the target word. The context in the tagging network spans the entire input sentence. For each sentence of length L $\{w_1, w_2, ..., w_L\}$, every word w_j is represented by vector $\mathbf{v}_j$, which is comprised of the concatenation: $\mathbf{v}_j = [\mathbf{w}_j; \mathbf{s}_j; \mathbf{a}_j]$, where $\mathbf{w}_j$ is the word embedding vector, $\mathbf{s}_j$ is a vector representation of the characters within the word, and $\mathbf{a}_j$ is a vector representing all the candidate morphological tags (from an analyzer), for all the non-lexicalized morphological features.

To obtain the vector $\mathbf{s}_j$, we use an LSTM-based model, applied to the character sequence in each word separately. We use the last state vector as the embedding representation of the word's characters. Whereas to get the $\mathbf{a}_j$ vector, for each morphological feature f, we use a morphological analyzer to obtain all possible feature values of the word to be analyzed. We then embed each value separately (with separate embedding tensors for each feature, learnt within the model), then sum all the resulting vectors to to get $\mathbf{a}_j^f$ (since these tags are alternatives and do not constitute a sequence) (Zalmout and Habash, 2019). We concatenate the individual $\mathbf{a}_j^f$ vectors for each morphological feature f of each word, to get a single representation, $\mathbf{a}_j$, for all the features:

$$\mathbf{a}_j^f = \sum_{n=1}^{N_f} \mathbf{a}_{j,n}^f$$

$$\mathbf{a}_j = [\mathbf{a}_j^{pos}; ...; \mathbf{a}_j^{num}; ...; \mathbf{a}_j^{vox}]$$

Where N_f is the set of possible candidate values for each feature f (from the analyzer). The $\mathbf{a}_j$ vector does not constitute a hard constraint and can be discarded if a morphological analyzer is not used.

Several previous contributions for Arabic showed that pretraining the word embeddings is very useful (Erdmann et al., 2018; Watson et al., 2018; Zalmout and Habash, 2017), including the baselines used in this paper. We therefore pre-train the word embeddings with FastText (Bojanowski et al., 2017), using a large external dataset. The pre-trained embeddings are fixed during the model training. The character and tag embeddings are learnt within the model.

We use a multitask learning setup to train the different morphological features jointly, through sharing the parameters of the hidden layers in the Bi-LSTM network. The input is also shared, through the $\mathbf{v}_j$ vector. The output of the network is then fed to a separate non-linearity function, output layer, and softmax, for a probability distribution of each of the features separately. Figure 1 shows the overall tagging architecture.

3.2 Encoder

We share the character and word embeddings from the tagger network in the encoder. The input context is modeled through a sliding window of a fixed number of characters around the target word, as in the Lematus model (Bergmanis and Goldwater, 2018). We also use additional special symbols for the whitespace and target word boundaries. In addition to the character embeddings, we also condition on the word level embedding of the word containing the characters. We concatenate the word embedding vector with the input character embeddings. Each character embedding $\mathbf{c}_i$ is replaced by the concatenation $[\mathbf{c}_i; \mathbf{w}_j]$, where $\mathbf{w}_j$ is the d_w-dimensional word embedding of the word j in which character i appears in. Given the characters of input sentence c and its lemmatized equivalent y, the goal is to model $P(y_k|\mathbf{c}_i, \mathbf{w}_j)$.

We then feed the input vectors to a network of two Bi-LSTM layers for the hidden representation at the encoder.

3.3 Decoders

We use separate decoders for lemmatization and diacritization, with two LSTM layers for each. Both decoders share the same input and parameters of the encoder Bi-LSTM network. For each decoder, we condition on the decoder output of the previous step, along with Luong attention (Luong et al., 2015) over the encoder outputs h_i, and the predicted tags from the tagger. We use the last encoder output as the initial states for the decoder layers. We use scheduled sampling (Bengio et al., 2015) during training, and feed the d_c-dimensional character embeddings at every time step. But we found empirically that using a constant sampling probability instead of scheduling provides better results.

We also use dropout on the non-recurrent connections of both the encoder and decoder layers during training. The decoder outputs are fed to a softmax layer that reshapes the vectors to dimension d_{voc}, then argmax to yield an output sequence $\mathbf{y}$ one character at a time.

Conditioning on the Predicted Tags In addition to the attention distribution and the previous time step, we also condition on the predicted tags from the tagger during decoding. The goal is to provide an additional contextual signal to the decoders, and to disambiguate the possible lexical choices. We use the output of the argmax (over the softmax distribution) for each feature, and concatenate the different tags as in the $\mathbf{a}_j$ vector:

$$\hat{\mathbf{t}}_j = [\hat{\mathbf{t}}_j^{asp}; ...; \hat{\mathbf{t}}_j^{pos}; ...; \hat{\mathbf{t}}_j^{vox}]$$

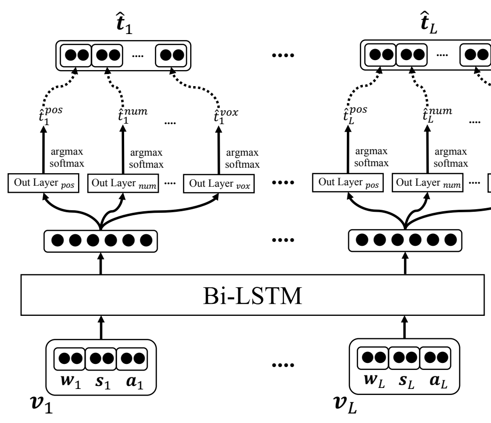

Figure 1: The tagger model, showing the multitask learning architecture for the features. The concatenated predicted tags are used to condition on, at the decoders.

Preventing Backpropagation to Tagger The decoder produces the lexicalized features at the character level, whereas the predicted tags are on the word level. The different granularities might create some biases, and we found that backpropagating gradients from the decoder to the tagger network leads to instability at the tagger. Therefore, we prevent the decoder from backpropagating gradients to the tagger during training. This is consistent with the model of Kondratyuk et al. (2018).

3.4 Surface Form Normalization

We use the term *normalization* in the sense of *enriched normalization* introduced by El Kholy and Habash (2012) for MSA; and in the sense of *spelling conventionalization* (into CODA) for DA as described by Eskander et al. (2013). Both

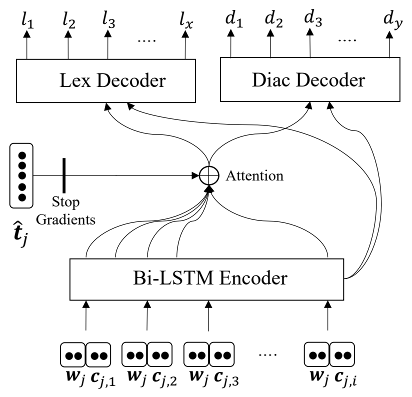

Figure 2: The sequence-to-sequence architecture for the lexicalized features, with a shared encoder, and separate decoders for lemmatization and diacritization. The figure does not show the fixed context window of 10 characters before and after the target word.

are non-trivial tasks comparable to true-casing or spelling correction for other languages.

The normalization task is particularly important for dialectal content, which lack a standardized orthography. The training data that we use has the diacritized annotations already in the CODA normalized form for EGY. So the output sequence of the diacritization task should be both the diacritized and CODA normalized version of the input sequence. This normalization is learnt explicitly in our character level sequence-to-sequence model. For MSA there is no need for CODA normalization, so the normalized output includes any error correction that might happen in the training dataset. Normalization is assessed as part of the overall diacritization accuracy.

3.5 Training Procedure

We use a small held out tuning set of about 5% of the training data to save the best model during training. We did not use the development set here to be consistent with other contributions in literature, where the development set is primarily used to evaluate high level design decisions only. We train the model for a fixed number of epochs and select the model that performs best on the tuning set. This method provided the most stable results, compared to early stopping or other methods.

The loss function is based on minimizing cross entropy H for each feature f. The overall loss is the average of the individual losses for the different features, whether lexicalized or non-lexicalized:

$$H(\hat{y}, y) = \frac{1}{|F|} \sum_{f \in F} H(\hat{y}^f, y^f)$$

Where F is the set of features that we model. y represents the true feature value, and $\hat{y}$ is the predicted value. We experimented with having different optimizers for the lexicalized and non-lexicalized features. We also experimented with a weighted average for the different features, where the weights are learnt as part of the end-to-end system. None of these modifications provided any improvement. We use Adam optimizer (Kingma and Ba, 2014) with a learning rate of 0.0005, and we run the various models for 50 epochs.

3.6 Full Morphological Disambiguation

Morphological disambiguation involves predicting the right combination of morphological features for each word in context. We can either present the predicted features from the model directly, or use a morphological analyzer to guarantee more consistent feature values. If a morphological analyzer is used, the disambiguation system selects the optimal analysis for the word from the set of analyses returned by the analyzer. We use the predicted tags to rank the analyses, and select the analysis with highest number of matched feature values. The different features can be assigned different weights during ranking. Refer to other contributions that use a similar approach for more details (Zalmout and Habash, 2019, 2017; Pasha et al., 2014).

4 Experiments and Results

4.1 Data

We use the Penn Arabic Treebank (PATB parts 1,2, and 3) (Maamouri et al., 2004) for MSA, and the ARZ dataset (Maamouri et al., 2012) from the Linguistic Data Consortium (LDC), parts 1–5, for EGY. We use the same datasets as used in MADAMIRA (Pasha et al., 2014), which involves synchronizing the datasets with morphological analyzers, using the process described by Habash and Rambow (2005). We follow the data splits recommended by Diab et al. (2013) for TRAIN, DEVTEST, and BLINDTEST.[3] Both datasets include gold annotations for the diacritized forms, lemmas, and the remaining 14 features. The diacritized forms are normalized following the CODA guidelines for EGY. We use Alif/Ya and Hamza normalization, which is commonly used for morphological modeling in Arabic (Zalmout et al., 2018; Pasha et al., 2014; Habash et al., 2013).

Table 2 shows the data sizes. The TUNE dataset is used during the model training process, for early stopping or to keep the best performing model. TUNE is extracted randomly from the original TRAIN split (almost 5% of TRAIN), so the other splits are consistent with the splits used in literature. The DEVTEST dataset is used during the system development to assess design choices. The BLINDTEST dataset is used to evaluate the system after finalizing the architecture design, and to report the overall performance.

	TRAIN	TUNE	DEVTEST	BLINDTEST
MSA	479K	23K	63K	63K
EGY	127K	6K	21K	20K

Table 2: Word count statistics for MSA and EGY.

We use the same morphological analyzers that were used in MADAMIRA (Pasha et al., 2014), and the other baselines, for both MSA and EGY. For MSA we use SAMA (Graff et al., 2009), and the combination of SAMA, CALIMA (Habash et al., 2012b), and ADAM (Salloum and Habash, 2014) for EGY. We use the LDC's Gigaword corpus (Parker et al., 2011) to pretrain the MSA word embeddings, and the BOLT Arabic Forum Discussions corpus (Tracey et al., 2018) for EGY, as used in the reported baselines. We preprocessed both datasets with Alif/Ya and Hamza normalization, as we did for the training dataset.

4.2 Experimental Setup

Tagger We use a similar setup as used by Zalmout and Habash (2019). We use two Bi-LSTM hidden layers of size 800, and dropout probability of 0.4, with peephole connections. The LSTM

[3]We use the LDC datasets because their annotations cover many of the tasks that are relevant to morphological disambiguation, and they are often used for benchmarking purposes. Other available datasets are usually limited to a particular task, like diacritization or POS tagging (Darwish et al., 2017, 2018; Abandah et al., 2015). Evaluating our model using these datasets is also not straightforward, since they often use different tagsets or representations (especially for diacritization), for which automatic conversion would require extensive post-processing.

character embedding architecture uses two LSTM layers of size 100, and embedding size 50. We use FastText (Bojanowski et al., 2017) to pretrain the word embeddings, with embedding dimension of 250, and an embedding window of size two.

Encoder-Decoder We use two LSTM layers of size 400 for both the encoder and decoder (bidirectional for the encoder), dropout value of 0.4, fixed sampling probability of 0.4 (Bengio et al., 2015). We use the same word and character embeddings as the tagger. We use beam decoding with beam size of 5, and a context window of 10 characters before and after the target word.

Metrics The evaluation metrics we use include:

- POS accuracy (POS): The accuracy of the POS tags, of a tagset comprised of 36 tags (Habash et al., 2013).
- Non-lexicalized morphological features accuracy (TAGS): The accuracy of the combined 14 morphological features we model, excluding lemmas and diacritized forms.
- Diacritization accuracy (DIAC): The accuracy of the diacritized forms, for MSA only.
- CODA-based normalization accuracy (CODA): The accuracy of the CODA-normalized, and diacritized, EGY forms. MSA does not need CODA normalization.
- Lemmatization accuracy (LEMMA): Lemma accuracy. The lemmas are also fully diacritized in the LDC datasets, so this metric reflects the fully diacritized lemmas.
- Full Analysis Accuracy (FULL): Accuracy over the full analysis – the strictest metric.

Baselines The first baseline is MADAMIRA (Pasha et al., 2014), which is one of the most commonly used morphological disambiguation models for Arabic. We also use the model suggested by Zalmout and Habash (2017), which is based on a similar architecture, but uses LSTM taggers instead of the SVM models in MADAMIRA, and LSTM-based language models instead of the n-gram models. The last baseline uses a multitask learning architecture to model the different non-lexicalized features jointly, but neural language models for the lexicalized features (Zalmout and Habash, 2019). We use the same feature weights during the disambiguation process as this baseline.

4.3 Results

Table 3 presents the results for the baselines, and the joint modeling architecture. The results show a significant accuracy improvement for the joint modeling approach, compared to all baselines.

Diacritization The diacritization task seems to have benefited the most of the joint modeling architecture, with about 16% relative error reduction for MSA. This is probably due to the relatively large target space for diacritized forms when using the language modeling approach in the baseline, compared to lemmatization for example, which has a smaller overall types count. The character level sequence-to-sequence architecture is more suitable to this task, with a small character target space.

Normalization In the baseline model normalization is a byproduct of selecting the right analysis, rather than a modeling goal. However, character level models provide for an explicit and direct normalization capability, as the model learns to map the erroneous sequence to the normalized target sequence. Our model results in 12% relative error reduction for EGY.

Overall Feature Consistency An analysis is consistent if all the feature values are linguistically acceptable to co-occur with each other. For example, case is undefined for verbs, so if a verb analysis had a defined case value, this analysis is inconsistent. The same applies to consistency between the tags and the corresponding lemma (or diacritized form). The TAGS metric, which represents the accuracy of the combined non-lexicalized features, also shows noticeable improvement for MSA. The fact that TAGS improved, along with FULL, while the POS accuracy remained somewhat similar, indicates that the model is now producing more consistent morphological predictions. This improved consistency is probably the result of enhanced diacritization and lemmatization models, which provide a better signal to the overall analysis ranking. The improvement in TAGS for EGY, on the other hand, is limited. This indicates that the model was probably already producing more consistent non-lexicalized morphological features, and the improvement in the FULL metric is due to improved diacritization and lemmatization only.

The Role of Morphological Analyzers Morphological analyzers are also used to guarantee consistency in the predicted features. The base-

		Model	FULL	TAGS	DIAC	LEX	POS
MSA	(a)	MADAMIRA (SVM models + analyzer) (Pasha et al., 2014)	85.6	87.1	87.7	96.3	97.1
	(b)	LSTM models + analyzer (Zalmout and Habash, 2017)	90.4	92.3	92.4	96.9	97.9
	(c)	+ Multitask learning for the tags (Zalmout and Habash, 2019)	90.8	92.7	92.7	96.9	97.9
	(d)	Joint modeling + analyzer	**92.3**	**93.5**	**93.9**	**97.6**	**98.1**
	(e)	Joint modeling without analyzer	90.3	92.7	92.8	96.3	97.7

		Model	FULL	TAGS	CODA	LEX	POS
EGY	(a)	MADAMIRA (SVM models + analyzer) (Pasha et al., 2014)	76.2	86.7	82.4	86.4	91.7
	(b)	LSTM models + analyzer (Zalmout and Habash, 2017)	77.0	88.8	82.9	87.6	92.9
	(c)	+ Multitask learning for the tags (Zalmout and Habash, 2019)	77.2	88.8	82.9	87.6	**93.1**
	(d)	Joint modeling + analyzer	**79.5**	**89.0**	**85.0**	**88.5**	**93.1**
	(e)	Joint modeling without analyzer	73.2	84.9	81.5	84.4	91.1

Table 3: The results of the various models on the **DEVTEST** for MSA and EGY. The first and second baselines, (a) and (b), use separate models for the features, and the third, (c), uses a multitask learning architecture for the non-lexicalized features only.

lines and our best performing model all use morphological analyzers, to get the candidate tags at the input, and to produce the best analysis through the ranking process. We train our model without using the analyzer – without the **t** vector and without ranking – to evaluate its role in the morphological disambiguation task. The results are lower, both for MSA and EGY. However, the result for MSA is very close to the (Zalmout and Habash, 2017) baseline, which uses separate feature models (with the analyzer). This indicates that our model can match the accuracy of a strong baseline, without relying on expensive external resources. This does not apply to EGY, probably due to the lower training data size and noisier content. Even with a better model, morphological analyzers still provide additional consistency between the different features.

BLINDTEST Results The results for the BLINDTEST dataset were consistent with the DEVTEST. The accuracy for EGY using the strongest baseline is 78.1, based on the multitask learning architecture for the tags. The accuracy of the best system, using the joint modeling architecture along with the morphological analyzer, is 80.3. We also observed the same behavior for MSA, with somewhat similar values to DEVTEST. The strongest baseline had an accuracy of 90.8, whereas the best model had an accuracy of 92.6.

4.4 Error Analysis

The Role of Morphological Analyzers The goal is to assess the role of morphological analyzers in the consistency (following the consistency definition mentioned earlier) of the predicted features. We took a sample of 1000 words from the MSA DEVTEST, and ran it through the joint model that does not use a morphological analyzer, and checked the errors in the predictions. There were 110 errors (11% of the sample), for an accuracy of 89%, which is close to the reported accuracy over the entire dataset. About 62% of the errors had consistent feature predictions, but the predicted analysis did not match the gold. And around 13% of the errors are due to gold errors. Around 25% of the errors (2.8% of sample) had inconsistent predictions. This roughly matches the accuracy gap between the joint model with and without the morphological analyzer, which is also around 2%. This indicates that the accuracy boost that the morphological analyzer provides is to a large extent due to the consistency it conveys. We also observed that 37% of the inconsistent predictions (1% of the sample) had a correct lemma, but the lemma was inconsistent with the analysis. The remaining 63% (1.7% of sample), had an invalid lemma.

Joint Modeling vs Separate Modeling We also investigated the distribution of errors over the different features for the joint model against the baseline of separate feature models, both using the morphological analyzer. We annotated the errors in a 1000-word sample from DEVTEST, for both MSA and EGY, with the main erroneous feature. For example, if the predicted analysis is a verb inflection of a gold noun, the main erroneous feature would be the POS tag, even if other features ended up being wrong as a result. For MSA, the error distribution for the baseline is: case 27%, diacritization 22%, POS 18%, lemmatization 13%, gold errors 11%, and smaller percentages for state, voice, person, and enclitics. Whereas the distribution for the joint model is: case 26%, POS 21%, lemmatization 18%, gold errors 14%, diacritization 13%, and small percentages for state, voice, and person. In both models, case dominates the error distribution, since identifying the case ending in MSA is particularly challenging. The main difference between the models in terms of error distribution is the diacritization, where we observe a significant boost when we use the joint model. The apparent increase in the error percentages of the other error types at the joint model is due to the drop in the overall errors count, while many have a lower drop rate.

For EGY, a notable error pattern is when the prediction matches the MSA-equivalent analysis of the dialectal word, like having an MSA-like diacritization, or having a case ending (DA, like EGY, does not have case ending). This happens due to code-switching with MSA in the dialectal content, which is also reflected at the analyzer. This error type is not an error per se, but we do include it in the analysis. The error distribution for the separate features baseline is: gold errors 23%, MSA-equivalents 21%, POS 17%, lemmatization 14%, diacritization 12%, and smaller percentages for several other error types. Whereas the distribution for the joint model is: gold errors 27%, MSA-equivalents 21%, lemmatization 18%, POS 14%, diacritization 7%, and smaller frequencies for the other errors. Gold errors are frequent, but this is consistent with other contributions that use the same dataset (Zalmout et al., 2018). Like MSA, the percentage increase of the other error types is due to lower drop rates.

5 Conclusions and Future Work

We presented a joint modeling approach for the lexicalized and non-lexicalized features in morphologically rich and Semitic languages. Our model achieves a significant improvement over several baselines for Arabic, and matches the baseline for MSA without having to use an expensive morphological analyzer. The results highlight the benefits of joint modeling, where diacritization seems to have benefitted the most. We observe, however, that further research is needed to enhance the overall consistency of the predicted features, without relying on external morphological analyzers.

6 Acknowledgment

The first author was supported by the New York University Abu Dhabi Global PhD Student Fellowship program. The support and resources from the High Performance Computing Center at New York University Abu Dhabi are gratefully acknowledged.

References

Gheith A Abandah, Alex Graves, Balkees Al-Shagoor, Alaa Arabiyat, Fuad Jamour, and Majid Al-Taee. 2015. Automatic diacritization of Arabic text using recurrent neural networks. *International Journal on Document Analysis and Recognition (IJDAR)*, 18(2):183–197.

Ahmed Abdelali, Kareem Darwish, Nadir Durrani, and Hamdy Mubarak. 2016. Farasa: A fast and furious segmenter for Arabic. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, pages 11–16, San Diego, California.

Rania Al-Sabbagh and Roxana Girju. 2012. A supervised POS tagger for written Arabic social networking corpora. In *Proceedings of KONVENS 2012*, pages 39–52. OGAI. Main track: oral presentations.

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

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

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. In *Proceedings of the 28th International Conference on Neural Information Processing Systems-Volume 1*, pages 1171–1179. MIT Press.

Toms Bergmanis and Sharon Goldwater. 2018. Context sensitive neural lemmatization with lematus. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, volume 1, pages 1391–1400.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Grzegorz Chrupala, Georgiana Dinu, and Josef van Genabith. 2008. Learning morphology with morfette. *LREC 2008*, pages 2362–2367.

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

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

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 the 5th Meeting of the North American Chapter of the Association for Computational Linguistics/Human Language Technologies Conference (HLT-NAACL04)*, pages 149–152, Boston, MA.

Kevin Duh and Katrin Kirchhoff. 2005. POS tagging of dialectal Arabic: a minimally supervised approach. In *Proceedings of the ACL Workshop on Computational Approaches to Semitic Languages*, Semitic '05, pages 55–62, Ann Arbor, Michigan.

Ahmed El Kholy and Nizar Habash. 2012. Orthographic and morphological processing for English–Arabic statistical machine translation. *Machine Translation*, 26(1-2):25–45.

Alexander Erdmann, Nasser Zalmout, and Nizar Habash. 2018. Addressing noise in multidialectal word embeddings. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 558–565, Melbourne, Australia. Association for Computational Linguistics.

Ramy Eskander, Nizar Habash, Owen Rambow, and Nadi Tomeh. 2013. Processing Spontaneous Orthography. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, Atlanta, GA.

Nerea Ezeiza, Iñaki Alegria, José María Arriola, Rubén Urizar, and Itziar Aduriz. 1998. Combining stochastic and rule-based methods for disambiguation in agglutinative languages. In *Proceedings of the 17th international conference on Computational linguistics-Volume 1*, pages 380–384. Association for Computational Linguistics.

David Graff, Mohamed Maamouri, Basma Bouziri, Sondos Krouna, Seth Kulick, and Tim Buckwalter. 2009. Standard Arabic Morphological Analyzer (SAMA) Version 3.1. Linguistic Data Consortium LDC2009E73.

Nizar Habash, Mona Diab, and Owen Rambow. 2012a. Conventional Orthography for Dialectal Arabic: Principles and Guidelines – Egyptian Arabic. Technical Report CCLS-12-02, Columbia University Center for Computational Learning Systems.

Nizar Habash, Fadhl Eryani, Salam Khalifa, Owen Rambow, Dana Abdulrahim, Alexander Erdmann, Reem Faraj, Wajdi Zaghouani, Houda Bouamor, Nasser Zalmout, Sara Hassan, Faisal Al-Shargi, Sakhar Alkhereyf, Basma Abdulkareem, Ramy Eskander, Mohammad Salameh, and Hind Saddiki. 2018. Unified guidelines and resources for Arabic dialect orthography. In *Proceedings of the 11th Language Resources and Evaluation Conference*, Miyazaki, Japan. European Language Resource Association.

Nizar Habash, Ramy Eskander, and Abdelati Hawwari. 2012b. A Morphological Analyzer for Egyptian Arabic. In *Proceedings of the Twelfth Meeting of the Special Interest Group on Computational Morphology and Phonology*, pages 1–9, Montréal, Canada.

Nizar Habash and Owen Rambow. 2005. Arabic Tokenization, Part-of-Speech Tagging and Morphological Disambiguation in One Fell Swoop. In *Proceedings of the 43rd Annual Meeting of the ACL*, pages 573–580, Ann Arbor, Michigan.

Nizar Habash and Owen Rambow. 2007. Arabic diacritization through full morphological tagging. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Companion Volume, Short Papers*, pages 53–56, Rochester, New York. Association for Computational Linguistics.

Nizar Habash, Ryan Roth, Owen Rambow, Ramy Eskander, and Nadi Tomeh. 2013. Morphological analysis and disambiguation for dialectal Arabic. In *Proceedings of NAACL-HLT*, pages 426–432, Atlanta, Georgia.

Nizar Habash, Abdelhadi Soudi, and Tim Buckwalter. 2007. On Arabic Transliteration. In A. van den Bosch and A. Soudi, editors, *Arabic Computational Morphology: Knowledge-based and Empirical Methods*. Springer.

Nizar Y Habash. 2010. *Introduction to Arabic natural language processing*, volume 3. Morgan & Claypool Publishers.

Georg Heigold, Josef van Genabith, and Günter Neumann. 2016. Scaling character-based morphological tagging to fourteen languages. In *2016 IEEE International Conference on Big Data (Big Data)*, pages 3895–3902.

Go Inoue, Hiroyuki Shindo, and Yuji Matsumoto. 2017. Joint prediction of morphosyntactic categories for fine-grained Arabic part-of-speech tagging exploiting tag dictionary information. In *Proceedings of the 21st SIGNLL Conference on Computational Natural Language Learning (CoNLL)*, Vancouver, Canada.

Jenna Kanerva, Filip Ginter, and Tapio Salakoski. 2019. Universal lemmatizer: A sequence to sequence model for lemmatizing universal dependencies treebanks. *arXiv preprint arXiv:1902.00972*.

Salam Khalifa, Nasser Zalmout, and Nizar Habash. 2016. Yamama: Yet another multi-dialect Arabic morphological analyzer. In *Proceedings of the International Conference on Computational Linguistics (COLING): System Demonstrations*, pages 223–227.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Daniel Kondratyuk, Tomáš Gavenčiak, Milan Straka, and Jan Hajič. 2018. Lemmatag: Jointly tagging and lemmatizing for morphologically rich languages with brnns. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4921–4928.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*.

Mohamed Maamouri, Ann Bies, Tim Buckwalter, and Wigdan Mekki. 2004. The Penn Arabic Treebank: Building a Large-Scale Annotated Arabic Corpus. In *NEMLAR Conference on Arabic Language Resources and Tools*, pages 102–109, Cairo, Egypt.

Mohamed Maamouri, Sondos Krouna, Dalila Tabessi, Nadia Hamrouni, and Nizar Habash. 2012. Egyptian Arabic Morphological Annotation Guidelines.

Chaitanya Malaviya, Shijie Wu, and Ryan Cotterell. 2019. A simple joint model for improved contextual neural lemmatization. *CoRR*, abs/1904.02306.

Guido Minnen, John Carroll, and Darren Pearce. 2001. Applied morphological processing of English. *Natural Language Engineering*, 7(3):207–223.

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)*, pages 2390–2395, Minneapolis, Minnesota. Association for Computational Linguistics.

Thomas Müller, Ryan Cotterell, Alexander Fraser, and Hinrich Schütze. 2015. Joint lemmatization and morphological tagging with lemming. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2268–2274.

Rani Nelken and Stuart M. Shieber. 2005. Arabic Diacritization Using Finite-State Transducers. In *Proceedings of the ACL Workshop on Computational Approaches to Semitic Languages*, pages 79–86, Ann Arbor.

Robert Parker, David Graff, Ke Chen, Junbo Kong, and Kazuaki Maeda. 2011. Arabic Gigaword Fifth Edition. LDC catalog number No. LDC2011T11, ISBN 1-58563-595-2.

Arfath Pasha, Mohamed Al-Badrashiny, Ahmed El Kholy, Ramy Eskander, Mona Diab, Nizar Habash, Manoj Pooleery, Owen Rambow, and Ryan Roth. 2014. MADAMIRA: A Fast, Comprehensive Tool for Morphological Analysis and Disambiguation of Arabic. In *In Proceedings of LREC*, Reykjavik, Iceland.

Tobias Pütz, Daniël De Kok, Sebastian Pütz, and Erhard Hinrichs. 2018. Seq2seq or perceptrons for robust lemmatization. an empirical examination. In *Proceedings of the 17th International Workshop on Treebanks and Linguistic Theories (TLT 2018), December 13–14, 2018, Oslo University, Norway*, 155, pages 193–207. Linköping University Electronic Press.

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 *ACL 2008: The Conference of the Association for Computational Linguistics; Companion Volume, Short Papers*, Columbus, Ohio. Association for Computational Linguistics.

Wael Salloum and Nizar Habash. 2014. ADAM: Analyzer for Dialectal Arabic Morphology. *Journal of King Saud University-Computer and Information Sciences*, 26(4):372–378.

Helmut Schmid, Arne Fitschen, and Ulrich Heid. 2004. Smor: A german computational morphology covering derivation, composition and inflection. In *LREC*, pages 1–263. Lisbon.

Qinlan Shen, Daniel Clothiaux, Emily Tagtow, Patrick Littell, and Chris Dyer. 2016. The role of context in neural morphological disambiguation. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 181–191, Osaka, Japan. The COLING 2016 Organizing Committee.

Jennifer Tracey, Haejoong Lee, Stephanie Strassel, and Safa Ismael. 2018. BOLT Arabic Discussion Forum Source Data. LDC catalog number LDC2018T10.

Daniel Watson, Nasser Zalmout, and Nizar Habash. 2018. Utilizing character and word embeddings for text normalization with sequence-to-sequence models. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 837–843.

Nasser Zalmout, Alexander Erdmann, and Nizar Habash. 2018. Noise-robust morphological disambiguation for dialectal Arabic. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 953–964, New Orleans, Louisiana. Association for Computational Linguistics.

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, Copenhagen, Denmark. Association for Computational Linguistics.

Nasser Zalmout and Nizar Habash. 2019. Adversarial multitask learning for joint multi-feature and multi-dialect morphological modeling. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, ACL '19, Stroudsburg, PA, USA. Association for Computational Linguistics.

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 44th Annual Meeting of the Association for Computational Linguistics*, pages 577–584, Sydney, Australia. Association for Computational Linguistics.