Don't Throw Those Morphological Analyzers Away Just Yet: Neural Morphological Disambiguation for Arabic

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

Abstract

This paper presents a model for Arabic morphological disambiguation based on Recurrent Neural Networks (RNN). We train Long Short-Term Memory (LSTM) cells in several configurations and embedding levels to model the various morphological features. Our experiments show that these models outperform state-of-the-art systems without explicit use of feature engineering. However, adding learning features from a morphological analyzer to model the space of possible analyses provides additional improvement. We make use of the resulting morphological models for scoring and ranking the analyses of the morphological analyzer for morphological disambiguation. The results show significant gains in accuracy across several evaluation metrics. Our system results in 4.4% absolute increase over the state-of-the-art in full morphological analysis accuracy (30.6% relative error reduction), and 10.6% (31.5% relative error reduction) for out-of-vocabulary words.

1 Introduction

Recurrent Neural Networks (RNN) in general, and Long Short-Term Memory (LSTM) cells in particular, have been proven very successful for various Natural Language Processing (NLP) tasks, especially those involving sequential data tagging. RNN models can produce near or above state-of-the-art performance with minimal language-specific feature engineering. These models have the capacity of capturing syntactic and semantic features through the lexical word-level embeddings, and subword features through character-level embeddings.

Morphologically rich languages pose many challenges to NLP through their high degree of ambiguity and sparsity. These challenges are exacerbated for languages with limited resources. Morphological analyzers help reduce sparsity by providing several out-of-context morpheme-based analyses for words, but they usually introduce ambiguity by returning multiple analyses for the same surface form. Therefore, the model would require a further step of morphological disambiguation to choose the correct analysis in context.

Morphological modeling involves heavy use of sequential tagging, so using an LSTM-based model would be highly advantageous. LSTM models are also optimal for long-sequence tagging in particular, so such systems should be able to outperform other deep learning models with fixed window-based modeling. Morphological disambiguation is a well studied problem in the literature, but LSTM-based contributions are still relatively scarce. In this paper we use Bidirectional-LSTM (Bi-LSTM) models for morphological tagging and language modeling, and use the results of these models in ranking the analyses of the morphological analyzer. We incorporate various subword and morphological features at different linguistic depths in the tagger, along with both word-based and character-based embeddings.

Our results show significant accuracy gains for all the morphological features we study, and across several evaluation metrics. We compare our system against a strong baseline and a state-of-the-art-system. We achieve 4.4% absolute over the state-of-the-art in full morphological analysis accuracy (30.6% relative error reduction). When evaluated for the out-of-vocabulary (OOV) words alone, the system achieves 10.6% absolute increase (31.5% relative error reduction), and shows significant performance boost across all evaluation metrics.

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 713–722
Copenhagen, Denmark, September 7–11, 2017.

2 Linguistic Issues

What distinguishes morphologically rich languages (MRL), like Arabic, from other languages is that their words include more morphemes (such as prefixes and suffixes) representing a number of morphological features, e.g., gender, number, person, mood, as well as attachable clitics. Table 1 shows the set of 16 Arabic morphological features we model As a result, MRLs tend to have more fully inflected words (types) than their poor counterparts. For instance when comparing Modern Standard Arabic (an MRL) with English (not an MRL), the total number of Arabic words in a large corpus is 20% less than the English parallel version of the corpus; however the the total number of unique Arabic types is twice that of English (El Kholy and Habash, 2010).

Feature	Definition
diac	Diacratization
lex	Lemma
pos	Basic part-of-speech tags (34 tags)
gen	Gender
num	Number
cas	Case
stt	State
per	Person
asp	Aspect
mod	Mood
vox	Voice
prc0	Proclitic 0, article proclitic
prc1	Proclitic 1, preposition proclitic
prc2	Proclitic 2, conjunction proclitic
prc3	Proclitic 3, question proclitic
enc0	Enclitic

Table 1: The morphological features we use in the various models. The first two groups are lexical features; and the last two groups are inflectional and clitic features respectively, in addition to the part-of-speech tag.

Furthermore, MRLs have a tendency towards a higher degree of ambiguity, stemming from different interpretations of the same surface morphemes. In Modern Standard Arabic (MSA), this ambiguity is exacerbated by the language's diacritzation-optional orthography-leading a word to have about 12 analyses per word on average (Habash, 2010). These two issues, form richness and form ambiguity, are at the heart of why MRLs are challenging to NLP. Richness of form increases model sparsity, and ambiguity makes disambiguation harder. Table 2 shows an example of the various in-context and out-of-context morphological analyses of the word قيمتها *qymtha*[1] ('its value' among other readings).

A potential solution is to build a morphological analyzer, also known as morphological dictionary, that encodes all the word inflections in the language. A good morphological dictionary should cover all the inflected forms of a word lemma (richness); and return all the possible analyses of a surface word (ambiguity). Finally, both richness and ambiguity are more challenging when an MRL has limited data, and when the data is noisy.

3 Background and Related Work

Deep learning models have recently emerged as a viable approach for several morphological modeling tasks in general. Neural approaches are particularly appealing due to their generic modeling capabilities that can be scaled to multiple tasks, and for less reliance on specific feature engineering. Notable contributions include the work of Collobert et al. (2011), where they present a learning model that is applicable to several NLP tasks, like chunking, named entity recognition, and part-of-speech (POS) tagging, by deliberately avoiding task-specific feature engineering. They use a window-based deep neural network. The fixed window size, however, limits access to further parts of the sentence that might be relevant to the target word. Moreover, the analysis is applied on the surface word level only, without considering any subword features. Several other contributions utilize somewhat similar approaches, with various neural architectures (Wang et al., 2015; Huang et al., 2015). Dos Santos and Zadrozny (2014), on the other hand, argue that subword information is useful for certain NLP tasks, like POS tagging. They propose a character-based embedding along with the word embeddings, to be able to capture internal morphemic structures. Character embeddings, capturing subword features, are well studied in other contributions too (Labeau et al., 2015; Rei et al., 2016; Belinkov and Glass, 2015).

Morphological disambiguation, however, has

[1] All Arabic transliterations are provided in the Habash-Soudi-Buckwalter transliteration scheme (Habash et al., 2007).

واشارت الصحيفة الى ان تاريخ رسم اللّوحة ليس معروفا الا ان **قيمتها** تقدر ب ٢٥ مليون دولار.

wAšArt AlSHyfħ Alý An tAryx rsm AllwHħ lys mErwfA AlA An **qymthA** *tqdr b 25 mlywn dwlAr .*

The newspaper pointed out that the date of the painting is unknown, but **its value** is estimated at 25 million dollars.

diac	lex	gloss	pos	prc3	prc2	prc1	prc0	per	asp	vox	mod	gen	num	stt	cas	enc0
qay∼amatohA	qay∼am	evaluate;assess	verb	0	0	0	0	3	p	a	i	f	s	na	na	3fs:dobj
qay∼amotahA	qay∼am	evaluate;assess	verb	0	0	0	0	2	p	a	i	m	s	na	na	3fs:dobj
qay∼amotihA	qay∼am	evaluate;assess	verb	0	0	0	0	2	p	a	i	f	s	na	na	3fs:dobj
qay∼amotuhA	qay∼am	evaluate;assess	verb	0	0	0	0	1	p	a	i	m	s	na	na	3fs:dobj
qay∼imatahA	qay∼im	caretaker	noun	0	0	0	0	na	na	na	na	f	s	c	a	3fs:poss
qay∼imatihA	qay∼im	caretaker	noun	0	0	0	0	na	na	na	na	f	s	c	g	3fs:poss
qay∼imatuhA	qay∼im	caretaker	noun	0	0	0	0	na	na	na	na	f	s	c	n	3fs:poss
qaymatahA	**qaymaħ**	**value;worth**	**noun**	**0**	**0**	**0**	**0**	**na**	**na**	**na**	**na**	**f**	**s**	**c**	**a**	**3fs:poss**
qaymatihA	qaymaħ	value;worth	noun	0	0	0	0	na	na	na	na	f	s	c	g	3fs:poss
qaymatuhA	qaymaħ	value;worth	noun	0	0	0	0	na	na	na	na	f	s	c	n	3fs:poss
qymthA	qymthA	NOAN	noun_prop	0	0	0	0	na	na	na	na	m	s	i	u	0

Table 2: An example highlighting Arabic's rich morphology and ambiguous orthography. The word قيمتها *qaymatahA* 'its value' has a specific analysis in the context of the sentence shown at the top of the table; but it has many other analyses and diacritizations out of context. The correct analysis is bolded (4th from the bottom of the list).

relatively fewer deep learning contributions. Yildiz et al. (2016) presented a disambiguation model for Turkish based on Convolutional Neural Networks (CNN). Their model creates a representation for the surface form of a word from the root along with a set of morphemic features. Then they train a model to predict the optimal analysis of a word given the annotations within a context window. Shen et al. (2016), on the other hand, use a character-based Bi-LSTM model for morphological disambiguation of morphologically complex languages, without using a morphological analyzer. The LSTM cells have the advantage of capturing a longer sequence window than those of the fixed window and CNN approaches.

Arabic morphological analysis and disambiguation have seen a considerable amount of work, spanning both MSA (Habash and Rambow, 2005; Diab et al., 2004; Khalifa et al., 2016; Abdelali et al., 2016), and Dialectal Arabic (Duh and Kirchhoff, 2005; Al-Sabbagh and Girju, 2012; Habash et al., 2013). The current state-of-the-art system is MADAMIRA (Pasha et al., 2014); which uses SVMs to disambiguate among a target word's various morphological analyses provided by a morphological dictionary.

Neural-based contributions for Arabic, however, are also relatively scarce. Among the contributions that utilize morphological structures to enhance the neural models in different NLP tasks, we note Guzmán et al. (2016) for machine translation, and Abandah et al. (2015) for diacritization. Darwish et al. (2017) use Bi-LSTM models to train a POS tagger, and compare it against SVM-based models. The SVM models in their system outperform the neural model, even with incorporating pre-trained embeddings. Heigold et al. (2016) developed character-based neural models for morphological tagging for 14 different languages, including Arabic, using the UD treebank. Most related to our work though is by Shen et al. (2016), who applied their Bi-LSTM morphological disambiguation model on MSA, but did not present any improvements over the state-of-the-art.

Occurring in parallel to our work, Inoue et al. (2017) used multi-task learning to model fine-grained POS tags, using the individual morphosyntactic features. They also use dictionary information concatenated to the word embeddings, similar to the approach we use in this paper, and use the same dataset. Our approach provides slightly higher accuracy scores for the individual features, but the joint features score in their system is higher.

In this paper we study various architectures for neural based morphological tagging. We then use these architectures, along with neural language modeling systems, to train models for various Arabic morphological features. We utilize these models for morphological disambiguation of the optimal analysis for each given word in context.

4 Approach

The morphological disambiguation task involves choosing the correct morphological analysis from the set of potential analyses, obtained from the an-

alyzer. Towards that end, we train several models for the individual morphological features, and use their results to score and rank the different analyses and choose an optimal overall analysis. These features can be grouped into non-lexical features, where a tagger is used to obtain the relevant morphological tag, or **morphological feature tagging**, and lexical features that need a language model (Roth et al., 2008), or **neural language models**. Table 1 shows the set of morphological features we work with. The lexical features are handled with a language model, while the inflectional, clitic, and part-of-speech features are handled with a tagger.

We use Bi-LSTM-based taggers for the morphological feature tagging tasks, with various embedding levels and morphological features. We investigate the different architectures and design options in detail in Section 5. We then use the best design to build 14 different taggers, each specific to an individual feature. We also use LSTM-based neural language models for the lexical features. We discuss the neural language models in more detail in Section 6.

We then use the results for these various models to score the potential morphological analyses from the analyzer for each given word. These scores are used to rank the analyses and return the one with the highest result. The process of scoring is also tuned through tuning weights for the used features. The details of the ranking and disambiguation process are provided in Section 6.

Dataset: We use the Penn Arabic Treebank (PATB parts 1,2 and 3) (Maamouri et al., 2004) for all the experiments in this paper. We follow the data splits recommend by Diab et al. (2013) for training, dev, and testing sets. We use Alif/Ya and Hamza normalization, and we remove all diacritics. The pre-trained word embeddings are trained using the LDC's Gigaword corpus for MSA (Parker et al., 2011). Table 3 shows the overall data sizes.

Dataset	Size (words)
Train	503,015
Dev	63,137
Test	63,172
Gigaword corpus	2,154M

Table 3: Dataset statistics

Evaluation: We use accuracy as the evaluation metric for all experiments reported in the paper.

Baselines: We use the Maximum Likelihood Estimation (MLE) baseline, calculated by counting the frequency scores for each given word/tag out of context, with backoff to the most frequent tag for unknown words. We also use the MADAMIRA (release-2.1) scores as another baseline, designated as the state-of-the-art system. Unless otherwise specified, MADAMIRA was configured in the ADD_PROP backoff mode, which adds a proper noun analysis to all words. We use this configuration to match the analyzer format we used in training the deep learning system, and to match the models in previous contributions.

5 Neural Morphological Feature Tagging Architectures

The task of morphological tagging in general relies on the context for accurate analysis. Such tasks can be modeled as a sequential data tagging problem, with both word and subword embeddings. While word embeddings are used to convey syntactic and semantic features, subword embeddings convey morphological features.

We present our morphological tagging model in this section, and use the POS feature as a test case. We then generalize our findings for all the other features for the morphological disambiguation process in Section 6. The POS tag set we use is the MADAMIRA tag set presented at (Pasha et al., 2014), and covered in detail at the MADAMIRA manual (Pasha et al., 2013), comprised of 34 tags.

5.1 Deep Learning Model

Given a sentence consisting of N words $\{w_1, w_2, ..., w_N\}$, every word w_i is converted into a vector

$$v_i = [r^{wrd}; r^{morph}]$$

which is composed of the word (or character sequence) embedding vector r^{wrd}, and the morphological features embedding vector r^{morph}. The morphological features vector can be constructed through various constructs, representing morphological and/or subword units.

We then use two LSTM layers to model the relevant context for both directions of the target word, where the input is represented by the v_n vectors

mentioned above:

$$\overrightarrow{c}_i = g(v_i, \overrightarrow{c}_{i-1})$$

$$\overleftarrow{c}_i = g(v_i, \overleftarrow{c}_{i+1})$$

We join both sides, apply a non-linearity function, and softmax to get a probability distribution. We use two hidden layers of size 800. Each layer is composed of two LSTM layers for each direction, and a dropout wrapper with keep probability of 0.8, and peephole connections. We use Adam optimizer (Kingma and Ba, 2014) with a learning rate of 0.003, and cross-entropy cost function. We use Tensorflow as the development environment.

subword and Morphological Features Several features can be used to represent the r^{morph} vectors mentioned before. These features are utilized to convey morphological information that are not represented at the word-level embeddings. We use various features with various linguistic depth:

(a) Fixed-width affixes We represent the prefixes and suffixes through a fixed character length substring from the beginning and the end of every word. This requires no linguistic information. We use a subset of three characters on both ends.

(b) Language-specific affixes (*lightstemmer*) We use regular expressions to maximally match affix patterns at the word's beginning and end. This requires basic linguistic knowledge of the target language, but doesn't require any large-scale lexical resources or annotated corpora.

(c) Potential POS tags from a morphological dictionary We use a high coverage morphological dictionary to obtain all possible POS tags of the target word. This requires advanced resources/annotations of the language. We include the set of potential tags in a vector representation and concatenate it with the word embedding.

The vector representation of these features is made up of the sum of the one-hot vectors for each individual component.

5.2 Word and Character Embeddings

Using character-level embeddings has recently been proven proficient for various NLP problems. In this paper we also study the effect of using word-level vs character-level embeddings on the overall morphological tagging problem, especially in light of the various subword and morphological features that we utilize. For word-level embeddings, we pre-train the word vectors using Word2Vec (Mikolov et al., 2013) on the Gigaword corpus mentioned in Section 4 (and Table 3), and the text of the training dataset. The embedding dimension for the words is 250. For the character-level embeddings, we concatenate the word embeddings with the sequence of character embeddings, initialized with their one-hot representation.

5.3 Results

Model	Accuracy
MLE Baseline	92.5
MADAMIRA (no backoff)	95.9
MADAMIRA (with backoff)	97.0

Table 4: Maximum Likelihood Estimation (MLE) and MADAMIRA baselines for POS tagging.

Model	Embedding	
	Word	Char
No Morphology	96.4	96.7
Fixed Character Affixes	96.6	NA
Lightstemmer	96.7	96.8
Morphological Dictionary	97.5	97.5
+ Fixed Character Affixes	**97.6**	NA
+ Lightstemmer	**97.6**	**97.6**

Table 5: Results for word embeddings (Word) and character-level embeddings (Char) for POS tagging. We don't provide character-level embeddings results for the Fixed Character Affixes approach, because such features would be redundant with the character embeddings themselves.

Table 4 shows the baseline scores for the systems, including the results for MADAMIRA with and without backoff. Table 5 shows the results for all systems. The results show clear improvement for all systems over the baseline and state-of-the-art without using subword. In fact, our system with no morphology outperforms MADAMIRA without using backoff. While our best result outperforms both MADAMIRA systems. Affixes (fixed length or lightstemmer) in general increase the accuracy across all systems. We notice, however, that the performance doesn't vary much between the fixed-width and lightstemmer affixes. This proves that the Bi-LSTM model is powerful enough to identify relevant features

from a character-stream only, without the need for language-specific affixes. Using the morphological dictionary has the largest effect of improvement across all systems (among the three morphology features used): 0.8% over the next best approach (absolute score difference). We obtain the highest accuracy scores when incorporating both the morphological dictionary tags, and affixes, whether for the fixed-width or the lightstemmer approaches. This system shows 20.0% error reduction over MADAMIRA with backoff, and 41% error reduction without backoff.

The character-level embedding system shows a somewhat similar behavior in terms of relative performance. We do not provide the results for the Fixed Character Affixes approach here since the character embeddings would capture these fixed affixes within the overall embedding vector anyway. Hence, it will only provide redundant representation without any additional information.

We observe that the character-based system, without any additional features, outperforms the word-based system. This is expected, since the system has access to subword features, conveyed in the characters stream, that are not available for the word-based system. The same behavior persists with the lightstemmer, but the performance gap is smaller, since the word-based system is now provided with similar subword features that the character stream conveys. These subword features are somewhat redundant for the character-based system, so the performance is only slightly better.

Surprisingly, however, both systems perform exactly the same when using the morphological dictionary features. This indicates that the morphological features are powerful enough to convey and exceed the subword features that a character stream can convey.

5.4 POS Tagging Error Analysis

We analyzed the resulting tag predictions against the gold tags by transforming the POS tag set space into four categories: nominals, verbs, particles, and punctuation, and observed the resulting error patterns. We noticed that the errors' distribution across all developed systems is somewhat similar throughout the four different categories. Nominals dominate almost 80.0% of all errors, even though they constitute 61.5% only of the total tokens. When introducing the morphological dictionary tags as features, all four categories increase in accuracy (except for the punctuation, being tagged almost correctly at all systems). Verbs, however, have the highest accuracy increase, at 1.5%, relative to 1.0% for nominals and 0.8% for particles. This can be the result of verbs being the least common category in the dataset at 8.0% (vs 64.0%, 14.0%, and 12.0% for nominals, particles, and punctuation, respectively). The nominals set is also relatively bigger than the other categories, which makes it internally confusable with errors within the nominals' options, like noun_adj or noun_num/noun_quant, among others.

6 Morphological Disambiguation

In this section we apply the morphological feature tagging architecture we discussed earlier for POS tagging to the remaining morphological features. We use the results of these taggers, along with the language models for *diac* and *lex*, as the input to the scoring and ranking process.

6.1 Morphological Tagging Models

Section 5 shows that the best performing neural architecture for POS tagging, as an example of morphological tagging in general, is using the embeddings (either character-based or word-based) with the relevant morphological tags from the dictionary, along with fixed or lightstemmer affixes. The performance of both word and character embeddings in this architecture was similar, so we opt for the word embeddings due to the excessive computational overhead affiliated with training character-level embeddings.

We apply the same architecture for the 14 morphological, non-lexical, features we study in this paper. Table 6 shows the results for the different taggers, relative to the MLE and MADAMIRA baselines that we used in the previous section. All features show significant performance boost.

Notable features though include case and state, where good tagging requires a relatively wide analysis window surrounding the target word. These features have the biggest performance gap between the baselines and the Bi-LSTM approach among the various other features. This is mainly due to the fact that LSTM cells have the capability of maintaining a longer sequence memory than the other approaches, hence capturing more of the sentence structure when tagging, compared to traditional window-based approaches.

System	pos	cas	num	gen	vox	mod	stt	asp	per	enc0	prc0	prc1	prc2	prc3
MLE	92.5	80.5	98.3	97.5	97.7	97.4	90.2	97.9	97.9	98.3	97.9	98.5	97.9	99.6
MADAMIRA	97.0	91.1	99.5	99.4	99.1	99.1	97.0	99.3	99.2	99.6	99.6	99.6	99.6	**99.9**
Bi-LSTM	97.6	94.5	99.6	99.5	99.2	99.4	97.9	99.4	99.4	99.7	99.7	99.8	**99.7**	**99.9**
Disambiguated Bi-LSTM	**97.9**	**94.8**	**99.7**	**99.7**	**99.4**	**99.6**	**98.3**	**99.6**	**99.6**	**99.8**	**99.8**	**99.9**	**99.7**	**99.9**
Absolute Increase	0.9	3.7	0.2	0.3	0.3	0.5	1.3	0.3	0.4	0.2	0.2	0.3	0.1	0.0
Error Reduction	30.0	42.0	40.0	50.0	33.0	56.0	43.0	43.0	50.0	50.0	50.0	75.0	25.0	0.0

Table 6: Morphological tagging results. The absolute increase and error reduction are of the disambiguated Bi-LSTM against MADAMIRA.

6.2 Neural Language Models

In addition to the morphological taggers for the non-lexical features, we use neural language models for the lemmatization and diacritization features. Lemmas and diacratized forms are lexical in nature, and cannot be modeled directly using a classifier. We use an LSTM-based neural language model (Enarvi and Kurimo, 2016), with class-based input rather than words. Using a class-based approach speeds convergence drastically and improves the overall perplexity, especially for the *diac* (diacritization) language model, which has a relatively large type count.

We use the MKCLS tool (Och, 1999), through GIZA++ (Och and Ney, 2003), to train the word classes. We use two hidden layers of size 500 and input layer of size 300, and use Nesterov Momentum as the optimization algorithm.

We encode the testing set in the HTK Standard Lattice Format (SLF), with a word mesh representation for the various options of each word.

Table 7 shows the accuracy results of the language models for *lex* and *diac* for both MADAMIRA (which uses SRILM (Stolcke, 2002) for language modeling), and the LSTM model we use here. All models are trained on the same ATB training dataset used in the paper. The LSTM results outperform MADAMIRA's vastly, proving the superiority of neural language models.

Feature	*lex*	*diac*
3-gram model	76.7	68.2
3-gram model disambiguated	96.2	87.7
Our system (LSTM)	89.6	73.5
Our system disambiguated	**96.9**	**91.7**

Table 7: The language model accuracy scores for both MADAMIRA and the LSTM models, for the *lex* and *diac* features.

6.3 Disambiguation

We use a similar morphological disambiguation approach to the model proposed by Habash and Rambow (2005) and Roth et al. (2008), where the resulting morphological features are matched and scored against the morphological analyzer options, as a way to rank the different analyses, and tuned using feature weights. If the analysis and the predicted morphological tag for a feature of a given word match, the analysis score for that analysis is incremented by the weight corresponding to that feature.

The morphological analysis with the highest score is chosen as the disambiguated option. Any tie-breaking after the disambiguation is handled through random selection among the reduced options[2]. For feature weight tuning we use the approach presented by Roth et al. (2008), using the Downhill Simplex Method (Nelder and Mead, 1965). A tuning dataset of almost 2K lines (∼63K words) is randomly selected from the training dataset. We retrain all the systems using the remaining training dataset, to be used in the tuning process. We finally use the resulting optimal weights in the original systems, trained on the full training dataset.

6.4 Evaluation

We use the following accuracy metrics to evaluate the disambiguation model, which Pasha et al. (2014) also use in their evaluation:

- EVALFULL: The percentage of correctly analyzed words across all morphological features. This is the strictest possible metric.
- EVALDIAC: The percentage of words where the chosen analysis has the correct fully diacritized form.

[2]This results in %0.02 variation range only in the EVALFULL end result.

Evalustion Metric	All Words			Out-Of-Vocabulary Words		
	MADAMIRA	Our System	Error Reduction	MADAMIRA	Our System	Error Reduction
EVALFULL	85.6	**90.0**	30.6	66.3	**76.9**	31.5
EVALDIAC	87.7	**91.7**	32.5	70.2	**79.8**	32.8
EVALLEX	96.2	**96.8**	15.8	82.9	**87.8**	28.7
EVALPOS	97.0	**97.9**	30.0	89.9	**96.0**	60.4
EVALATBTOK	99.4	**99.6**	33.3	94.2	**97.8**	60.2

Table 8: Accuracy results of the disambiguation system, evaluated using different metrics, for all words and out-of-vacbulary (OOV) words alone. OOV percentage of all words is 7.9%.

- EVALLEX: The percentage of words where the chosen analysis has the correct lemma.
- EVALPOS: The percentage of words where the chosen analysis has the correct part-of-speech.
- EVALATBTOK: The percentage of words that have a correct ATB tokenization.[3]

Deep learning models, through word embeddings, provide an advantage in terms of the analysis of unseen words. So, in addition to calculating the metrics for all the words in the testing set, we also calculate these metrics for the out-of-vocabulary (OOV) words alone.

Table 8 shows the accuracy scores for MADAMIRA and our system. All evaluation metrics indicate the performance boost of our system relative to MADAMIRA, with significant relative error reduction. The same trend stands for the OOV words, with even higher absolute and relative error reduction scores, especially for EVALLEX, EVALPOS, and EVALATBTOK. This increase in OOV analysis accuracy is the result of modeling the data on a semantic level, with the embeddings and neural networks, instead of pure lexical approach.

6.5 Discussion

We conducted additional data analysis over the test set comparing the performance of our system to MADAMIRA.

Comparative Error Patterns When considering full analyses, we observe that our system still makes some errors in words where MADAMIRA is correct. However, the number of times our system is correct and MADAMIRA is not is over twice as the reverse (MADAMIRA is correct and our system is not). From a manual analysis of a sample of 500 words, we observe the majority of the instances where MADAMIRA was correct and our system failed involved the case features. This is not surprising since case is one the features our system still struggles with although we have made major improvements beyond MADAMIRA. Shahrour et al. (2015) used syntax as an additional model to improve the analysis of case. Our model still improves the accuracy beyond theirs, but this highlights the value of using syntax in future work.

Minority Feature-Value Pairs While we show a lot of improvements across the board above in terms of accuracy, we also observe very large improvements in the performance on some minority feature-value pairs. For example, among the values of the *case* feature, the nominative (cas:n) and accusative (cas:a) appear about 7.0% and 11.0% of all words, respectively, of all the values of case. We improve the F-1 score from 70.4% in MADAMIRA to 84.1% in our system for (cas:n); and we similarly improve the F-1 score from 76.7% in MADAMIRA to 85.5% in our system for (cas:a). We also observe similar improvement in the *mood* feature, with the F-1 score for subjunctive mood (occurring 0.55% of all words) increasing from 76.9% in MADAMIRA to 89.5%.

This great increase was not observed across all features. The F1-score of the passive voice feature-value pair (vox:p) occurring 0.6% of all words (and 7.0% of all verbs) only increased from 70.1% in MADAMIRA to 73.4% in our system. Voice in Arabic is harder to model than mood and case since some verbal constructions can be rather ambiguous even for human readers; for example, the noun phrase الكاتبة التي نشرت مقالتها *AlkAtbħ Alty nšrt mqAlthA* has two readings 'the writer who published her article' (active voice) or 'the writer whose article was published' (passive voice). Case and mood are more likely to be de-

[3] ATB scheme tokenizes all clitics except the +ال *Al* 'the' determiner.

terminable from the context using long and short-distance syntactic clues. In the example above the case of the noun مقالتها *mqAlthA* 'her article' is dependent on the voice reading of the verb, which determines if the noun is the subject or object of the verb. For another example, the F1-score of the 2nd person feature-value pair (per:2) occurring 0.05% of all words (and 0.5% of all verbs) only increased from 29.7% in MADAMIRA to 31.7% in our system. The very low performance in the 2nd person makes sense, since the corpus we used is a news corpus where the 2nd person is hardly ever used. We would expect more training data to help such feature-value pairs.

7 Conclusion and Future Work

In this paper we presented an LSTM-based morphological disambiguation system for Arabic. The system significantly outperforms a state-of-the-art system. Our experiments showed that enriching the input word embedding with additional morphological features increases the morphological tagging accuracy drastically, beyond the capabilities of even character-level embeddings. We also showed that using an LSTM based system provides a significant performance boost for syntax based features, which often require wide context window for accurate tagging.

Future directions include exploring additional deep learning architectures for morphological modeling and disambiguation, especially joint and sequence-to-sequence models. We also intend to further investigate the role of syntax features in morphological disambiguation, and explore additional techniques for more accurate tagging. Finally, we aim at applying our models to Arabic dialects and other languages. We expect that character-level embeddings will have a bigger role in scenarios with noisy input, such as non-standard spontaneous orthography used in social media.

Acknowledgment The first author was supported by the New York University Abu Dhabi Global PhD Student Fellowship program.

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.

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 (EMNLP)*, pages 2281–2285.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(Aug):2493–2537.

Kareem Darwish, Hamdy Mubarak, Ahmed Abdelali, and Mohamed Eldesouki. 2017. Arabic POS tagging: Don't abandon feature engineering just yet. In *Proceedings of the Third Arabic Natural Language Processing Workshop*, pages 130–137, Valencia, Spain.

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

Cícero Nogueira Dos Santos and Bianca Zadrozny. 2014. Learning character-level representations for part-of-speech tagging. In *ICML*, pages 1818–1826.

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. 2010. Techniques for Arabic Morphological Detokenization and Orthographic Denormalization. In *Proceedings of the seventh International Conference on Language Resources and Evaluation (LREC)*, Valletta, Malta.

Seppo Enarvi and Mikko Kurimo. 2016. Theanolm - an extensible toolkit for neural network language modeling. *CoRR*, abs/1605.00942.

Francisco Guzmán, Houda Bouamor, Ramy Baly, and Nizar Habash. 2016. Machine translation evaluation for Arabic using morphologically-enriched embeddings. In *Proceedings of COLING 2016, the International Conference on Computational Linguistics*, pages 1398–1408, Osaka, Japan.

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, 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.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF models for sequence tagging. *arXiv preprint arXiv:1508.01991*.

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.

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, Osaka, Japan.

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

Matthieu Labeau, Kevin Löser, Alexandre Allauzen, and Rue John von Neumann. 2015. Non-lexical neural architecture for fine-grained pos tagging. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 232–237.

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.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.

J.A. Nelder and R. Mead. 1965. A simplex method for function minimization. *The Computer Journal*, 7:308–313.

Franz Josef Och. 1999. An efficient method for determining bilingual word classes. In *Proceedings of the Ninth Conference on European Chapter of the Association for Computational Linguistics*, EACL '99, pages 71–76, Stroudsburg, PA, USA.

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*.

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 *Proceedings of LREC*, Reykjavik, Iceland.

Arfath Pasha, Mohammad Al-Badrashiny, Mona Diab, Nizar Habash, Manoj Pooleery, and Owen Rambow. 2013. *MADAMIRA v1.0 User Manual*. http://www.nizarhabash.com/publications/MADAMIRA-UserManual.pdf.

Marek Rei, Gamal KO Crichton, and Sampo Pyysalo. 2016. Attending to characters in neural sequence labeling models. *arXiv preprint arXiv:1611.04361*.

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*, Columbus, Ohio.

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, Lisbon, Portugal.

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.

Andreas Stolcke. 2002. SRILM - an Extensible Language Modeling Toolkit. In *Proceedings of the International Conference on Spoken Language Processing (ICSLP)*, volume 2, pages 901–904, Denver, CO.

Peilu Wang, Yao Qian, Frank K. Soong, Lei He, and Hai Zhao. 2015. Part-of-speech tagging with bidirectional long short-term memory recurrent neural network. *CoRR*, abs/1510.06168.

Eray Yildiz, Caglar Tirkaz, H. Bahadir Sahin, Mustafa Tolga Eren, and Ozan Sonmez. 2016. A morphology-aware network for morphological disambiguation. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, pages 2863–2869. AAAI Press.