Noise-Robust Morphological Disambiguation for Dialectal Arabic

Nasser Zalmout, Alexander Erdmann, Nizar Habash
Computational Approaches to Modeling Language Lab
New York University Abu Dhabi
United Arab Emirates
{nasser.zalmout,ae1541,nizar.habash}@nyu.edu

Abstract

User-generated text tends to be noisy with many lexical and orthographic inconsistencies, making natural language processing (NLP) tasks more challenging. The challenging nature of noisy text processing is exacerbated for dialectal content, where in addition to spelling and lexical differences, dialectal text is characterized with morpho-syntactic and phonetic variations. These issues increase sparsity in NLP models and reduce accuracy. We present a neural morphological tagging and disambiguation model for Egyptian Arabic, with various extensions to handle noisy and inconsistent content. Our models achieve about 5% relative error reduction (1.1% absolute improvement) for full morphological analysis, and around 22% relative error reduction (1.8% absolute improvement) for part-of-speech tagging, over a state-of-the-art baseline.

1 Introduction

There has been a growing interest in noise-robust NLP tools recently, motivated by the sheer magnitude of user-generated content in social media platforms. The noisy nature of user-generated content makes its processing very challenging for NLP tools. Noisy content is non-canonical in nature, with lexical, orthographic, and phonetic variations that increase the perplexity and sparsity of NLP models. Several contributions show considerable drop in performance for a number of tasks, where simply retraining existing models with social media data does not provide substantial improvement (Gimpel et al., 2011; Ritter et al., 2011; Habash et al., 2013a).

Morphological disambiguation for noisy content is further complicated for dialectal content, with additional morpho-syntactic variations. Morphological disambiguation is also more challenging for morphologically rich and ambiguous languages, like Arabic and Dialectal Arabic (DA). Arabic is morphologically rich, having more fully inflected words (types) than morphologically poorer languages. It is also ambiguous, with short vowels (diacritic marks) often dropped and disambiguated in context. These issues result in more morpho-syntactic variations for DA in written text compared to other dialectal content, and increase the number of potential analyses.

We present several morphological disambiguation models for Egyptian Arabic (EGY), based on previous models for EGY and Modern Standard Arabic (MSA). We use a bidirectional long short term memory (Bi-LSTM) architecture and various noise reduction techniques, including character embedding and embedding space mapping. We also experiment with the width of the embedding window in the pre-trained embeddings. Character embeddings allow access to subword units, while the embedding space mapping normalizes non-canonical forms to canonical neighbors. The narrow/wide embedding window in the pre-trained embeddings allows for more of syntactic/semantic modeling, respectively.

The goal of the various models is to achieve noise-robust analysis, rather than explicit noise normalization. We therefore use the normalization techniques on the vector-level only, instead of replacing the raw forms, which allows for less aggressive lexical normalization. The separation of raw forms and vector normalization also allows for independent word and character level normalization, eliminating any propagation of error.

Our system achieves a 5% relative error reduction (1.1% absolute accuracy boost) over a state-of-the-art baseline, using a strict metric. Our noise-robust system also matches the performance of a version of the system trained and tested on a manually orthography-normalized copy of the data. This indicates that the system performs as well as could be expected without orthographic inconsistency. We also present an error analysis of the system and identify areas of improvement.

The rest of the paper is structured as follows. We present common challenges to DA processing in Section 2. This is followed by background and related work in Section 3. We introduce the approach and various models in Section 4, and discuss the experimental setup and results in Section 5. We conclude and provide some directions for future work in Section 6.

2 Linguistic Issues

Dialectal Arabic, including EGY among other dialects, is the primarily spoken language used by native Arabic speakers in daily exchanges. The outbreak of social media platforms expanded the use of DA as a written language. The lack of a standard orthography (Habash et al., 2012a), combined with the fact that user-generated content in social media is prone to noise, increase sparsity and reduce performance.

EGY, similar to MSA, is also a morphologically complex language, having a number of morphological features, e.g., gender, number, person, mood, and attachable clitics. Moreover, the diacritization-optional orthography for Arabic (both DA and MSA) results in orthographic ambiguity, leading to several interpretations of the same surface forms. Richness of form increases model sparsity, and ambiguity makes disambiguation harder. One approach to model complexity, richness, and ambiguity uses *morphological analyzers*, also known as morphological dictionaries. Morphological analyzers are usually used to encode all potential word inflections in the language. A good morphological dictionary should return all the possible analyses of a surface word (ambiguity), and cover all the inflected forms of a word lemma (richness), covering all related features. The best analysis is then chosen through *morphological disambiguation*.

The set of morphological features that we model for EGY morphological disambiguation includes:

- Lexicalized features: lemma, diacritization.
- Non-lexicalized features: aspect, case, gender, person, part-of-speech (POS), number, mood, state, voice.
- Clitics: enclitics, like pronominal enclitics, negative particle enclitics; proclitics, like article proclitic, preposition proclitics, conjunction proclitics, question proclitics.

Despite the similarities, EGY and MSA have many differences that prevent MSA tools from being effectively utilized for EGY text. These include lexical, phonological, and morphological inconsistencies. Lexical differences can be numerous, beyond simple cognates, like the word ازاي *AzAy*[1] 'how' in EGY corresponds to the word كيف *kyf* in MSA. There are also many morphological differences, for example the MSA future proclitic /sa/+ (spelled +س s+) appears in EGY as either /ha/+ (+ه) or /Ha/+ (+ح). There are also many phonological variations between EGY and MSA that have direct implications on orthography as well. These include the consonant ث /θ/ in MSA, which can be mapped to either ت /t/ or س /s/ in EGY. These variations make the written EGY content more susceptible to noise and inconsistency. Table 1 shows an EGY sentence example, along with the set of potential analyses for a given word.

3 Background and Related Work

Explicit handling of noisy content in NLP has recently gained momentum with the increasing use of social media outlets. Notable contributions for POS tagging include the ARK tagger (Owoputi et al., 2013), which is targeted for online conversational text. ARK tagger uses conditional random fields with word clusters as features, obtained via Brown clustering (Brown et al., 1992), along with various lexical features. Gimpel et al. (2011) also use conditional random fields for POS tagging, trained on annotated Twitter content. Derczynski et al. (2013) use manually curated lists to map low frequency and out of vocabulary terms to more frequent terms. Noisy content has also been addressed for named entity recognition (Liu et al., 2011; Ritter et al., 2011; Aguilar et al., 2017), and syntactic parsing (Foster et al., 2011; Petrov and McDonald, 2012).

Most relevant to our work is the paper by van der Goot et al. (2017), where they use Word2vec (Mikolov et al., 2013) to find potential normalization candidates for non-canonical words on the lexical level, and rank them using a classifier. They experiment with various normalization and embedding settings, and they find that both normalization and pre-trained embeddings are helpful for the task of POS tagging.

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

AntA fyn? fy AlmHl **brdk** *?*																انتا فين؟ في المحل **بردك** ؟
Where are you? Are you at the shop **still**?																
diacritization	**lemma**	**gloss**	**pos**	**prc3**	**prc2**	**prc1**	**prc0**	**per**	**asp**	**vox**	**mod**	**gen**	**num**	**stt**	**cas**	**enc0**
bi+rad∼+ak	rad∼	response	noun	0	0	bi_prep	0	na	na	na	na	m	s	c	u	2ms:poss
barDak	**barDak**	**still**	**adv**	**0**	**0**	**0**	**0**	**na**	**i**	**na**	**na**	**m**	**s**	**i**	**u**	**0**
bi+rad∼+ik	rad∼	response	noun	0	0	bi_prep	0	na	na	na	na	m	s	c	u	2fs:poss
b+Aarud∼+ak	rad∼	return	verb	0	0	0	bi_prog	1	i	a	i	m	s	na	na	2ms:dobj
bard+ak	bard	cold	noun	0	0	0	0	na	na	na	na	m	s	c	u	2ms:poss

Table 1: An example highlighting the effect of non-standard and ambiguous orthography, along with rich morphology, on EGY morphological disambiguation. The word برضك *barDak* 'still' is provided in the example with the non-standard (non-CODA compliant) orthography بردك *bardak*, which can lead to different morphological analyses than the one intended in context.

The issue of noisy text processing is exacerbated for dialectal content. Most contributions focus on spelling/lexical variations, whereas dialectal content is further characterized with morpho-syntactic and phonetic variations that make automatic processing more challenging (Jørgensen et al., 2015). In addition to the issues of morphological complexity, ambiguity, and lack of standard orthography for MSA and DA. There has been several contributions covering various NLP tasks including morphological analysis, disambiguation, POS tagging, tokenization, lemmatization and diacritization, addressing both MSA and DA (Al-Sabbagh and Girju, 2010; Mohamed et al., 2012; Habash et al., 2012b, 2013a; Abdelali et al., 2016; Khalifa et al., 2016b). Notable contributions for both MSA and EGY include MADAMIRA (Pasha et al., 2014), a morphological disambiguation tool that uses morphological analyzers to handle complexity and ambiguity. MADAMIRA can automatically correct common spelling errors as a side effect of disambiguation, but does not include explicit processing steps for noisy content. A neural version of MADAMIRA for MSA is presented by Zalmout and Habash (2017), who use Bi-LSTMs and morphological tag embeddings. Their system shows significant improvement over MADAMIRA, but does not use any explicit character embeddings nor noise reduction techniques.

To address the lack of standardized orthography for DA, Habash et al. (2012a) proposed CODA, a Conventional Orthography for Dialectal Arabic. CODA presents a detailed description of orthographic guidelines, mainly for the purpose of developing DA computational models, applied to EGY, and later extended to several other Arabic dialects (Zribi et al., 2014; Saadane and Habash, 2015; Turki et al., 2016; Khalifa et al., 2016a; Jarrar et al., 2016; Habash et al., 2018). CODA-treated DA content should be less sparse and less noisy. Eskander et al. (2013) presented a tool to normalize raw texts into a CODA compliant version using the K-nearest neighbor algorithm. Scaling this tool to other dialects, however, is challenging due to the lack of training data.

Our morphological tagging architecture is similar to the work of Inoue et al. (2017) and Zalmout and Habash (2017), but we further experiment with CNN-based character embeddings, and pre-train the word embeddings. The architecture is also similar to the work of Heigold et al. (2017) and Plank et al. (2016) in terms of the character embeddings, both LSTM and CNN-based systems. Our architecture, however, uses neural language models for modeling lemmas and diacritized forms, and utilizes the word-level embeddings in various configurations to combat noise, as explained throughout the rest of the paper.

4 Approach

We present a morphological disambiguation model for EGY. We use an LSTM-based architecture for morphological tagging and language modeling for the various morphological features in EGY. We also experiment with several embedding models for words and characters, and present several approaches for noise-robust modeling on the raw form and vector levels.

We present the overall tagging and disambiguation architecture, in addition to the character embedding model, in 4.1. We then present the noise handling approaches in 4.2 and 4.3.

4.1 Morphological Tagging and Disambiguation Architecture

We use a similar disambiguation approach as in previous contributions for MSA and EGY (Habash and Rambow, 2005; Habash et al., 2009, 2013b).

The morphological disambiguation task is intended to choose the correct morphological analysis from the set of potential analyses, obtained from the morphological analyzer. The analyzer provides a set of morphological features for each given word. These features can be grouped into non-lexical features, where a tagger is used to predict the relevant morphological tag, handled through *morphological feature tagging*, and lexical features that need a language model (Roth et al., 2008), handled through *lexicalized feature language models*. The inflectional, clitic, and part-of-speech features are handled with a tagger, while the lexical features are handled with a language model.

4.1.1 Morphological Feature Tagging

Overall Architecture We use Bi-LSTM-based taggers for the morphological feature tagging tasks. Given a sentence of length L words $\{w_1, w_2, ..., w_L\}$, every word w_i is converted into vector v_i:

$$v_i = [v_{w_i}; v_{c_i}; v_{t_i}]$$

composed of the word embedding vector v_{w_i}, word-level characters embedding vector v_{c_i}, and the candidate morphological tag embedding vector v_{t_i}. This separation of word and character embedding vectors enables further noise handling on the word embedding level alone, with the character embeddings learnt from the raw forms without any modification. We pre-train the word embeddings using Word2vec (Mikolov et al., 2013).

We use two LSTM layers to model the relevant context for both directions of the target word, where the input is represented by the v_i vectors mentioned above:

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

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

where h_i is the context vector from the LSTM for each direction. We join both sides, apply a non-linearity function, and softmax to get a probability distribution. Figure 1 shows the architecture.

Character Embedding We use convolutional neural networks (CNN) and LSTM-based architectures for the character embedding vectors v_{c_i}, both applied to the character sequence within each word separately. LSTM-based architectures have been shown to outperform CNN-based character

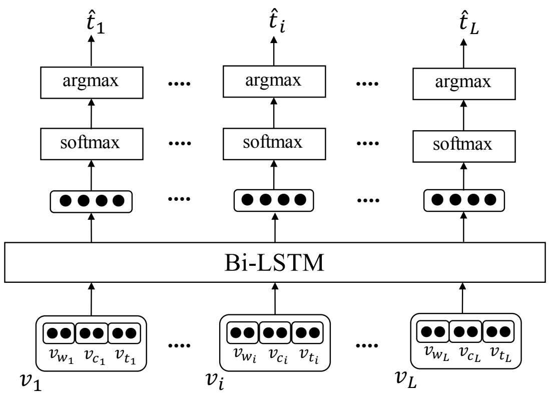

$\hat{t}_1$
$\hat{t}_i$
$\hat{t}_L$
argmax
softmax
Bi-LSTM
v_{w_1} v_{c_1} v_{t_1}
v_{w_i} v_{c_i} v_{t_i}
v_{w_L} v_{c_L} v_{t_L}
v_1
v_i
v_L

Figure 1: The overall tagging architecture, with the input vector as the concatenation of the word, characters, and candidate tag embeddings.

embedding in POS tagging (Heigold et al., 2017), but we experiment with both architectures to report their performance in noisy EGY content. We use various filter widths and max pooling for the CNN system, with the output fed to a dense connection layer. The resulting vector is used as the character embedding vector for the given word. For the LSTM-based architecture we use the last state vector as the embedding representation of the word's characters. Both architectures are outlined at figure 2.

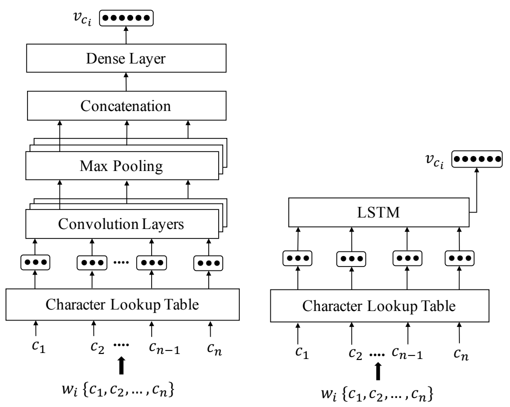

v_{c_i}
Dense Layer
Concatenation
Max Pooling
Convolution Layers
Character Lookup Table
c_1 c_2 .... c_{n-1} c_n
$w_i\ \{c_1, c_2, \dots, c_n\}$
v_{c_i}
LSTM
Character Lookup Table
c_1 c_2 .... c_{n-1} c_n
$w_i\ \{c_1, c_2, \dots, c_n\}$

Figure 2: CNN-based (left) and LSTM-based (right) architectures for word-level character embedding. Similar to the architecture by Heigold et al. (2017).

Morphological Tag Embedding The morphological features vector v_{t_i} embeds the candidate tags for each feature. The tags include the collection of morphological features. We use the morphological analyzer to obtain all possible tag values of the word to be analyzed. We use a lookup table to map the tags to their trainable vector representation, then sum all the resulting vectors to get v_{t_i}, since these tags are alternatives and do not constitute a sequence of any sort. Figure 3 outlines the tag embedding model.

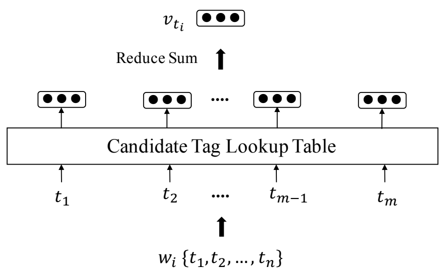


Figure 3: Candidate tag embedding, through summing the vectors of the individual tags.

Embedding the morphological tags using the analyzer does not constitute a hard constraint in the system, and the v_{t_i} vector can be discarded or substituted with less resource-demanding options for other languages or dialects.

4.1.2 Lexicalized Feature Language Models

We use LSTM-based neural language models (Enarvi and Kurimo, 2016) for the lexical features (lemma and diacritization). Lemmas and diacritized forms are lexical and cannot be modeled directly using a classifier (Habash and Rambow, 2007), since the target space is big (around 13K for lemmas, and 33K for the diacritized forms, in Train). We therefore use a language model to choose among the candidate lemmas and diacritized forms obtained from the analyzer. We encode the runtime dataset in the HTK Standard Lattice Format (SLF), with a word mesh representation for the various options of each word.

4.2 Embedding Window Width

Several contributions show that the window size (i.e. amount of context) in word embeddings affects the type of linguistic information that gets modeled. Goldberg (2016) and Trask et al. (2015) explain that larger windows tend to create more semantic and topical embeddings, whereas smaller windows capture syntactic similarities. Tu et al. (2017) also find that a window of one (one word before the target word and one word after) is optimal for syntactic tasks.

We experiment with both wide and narrow window embeddings, and evaluate their effects on tagging accuracy. These experiments show the role of topical or semantic vs syntactic embeddings in the morphological disambiguation model. We then experiment with embedding vector extension, by combining both wide and narrow embeddings through concatenation. This technique is expected to handle noisy and unstandardized spellings, since spelling variants are not just semantically related, but must share the same syntactic valency.

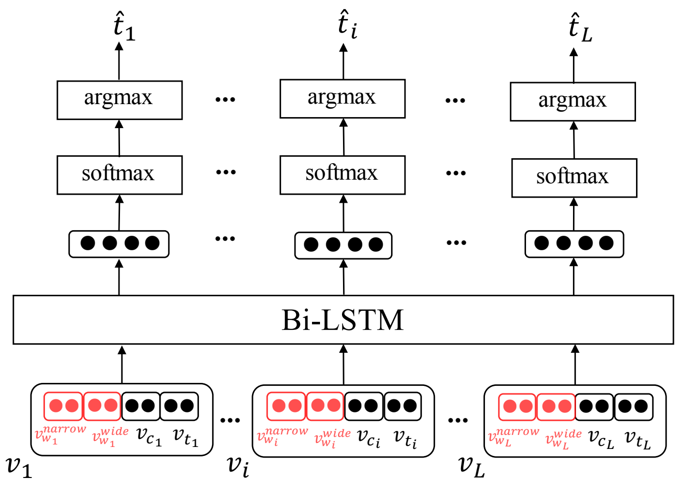
Figure 4 shows the updated architecture, with the narrow window embedding $v_{w_i}^{narrow}$ concatenated to the v_i vector, along with the existing wide window embedding $v_{w_i}^{wide}$.

4.3 Embedding Space Mapping

The embedding space mapping approach is based on the hypothesis that non-standard words are likely to have similar contexts as their canonical equivalents. We define the canonical equivalent here as the most frequent semantically and syntactically equivalent word to the target word. We use this definition since the operation is unsupervised, and for the lack of a standard canonical forms. Variants of this approach have been used in several spelling error correction tasks (Sridhar, 2015). Dasigi and Diab (2011) also use a similar approach to identify variants in DA. We use the Word2vec framework (Mikolov et al., 2013) in the Gensim implementation (Řehůřek and Sojka, 2010) to generate the embedding spaces. We use these embeddings to learn and score normalization candidates based on their cosine distance as a semantic score, and edit distance as a lexical score. In this scope, we first learn a weighted distance function for the individual insertion, deletion, and substitution operations, then use these weights to score the candidates.

Edit Distance Weights The spelling variants are first identified based on narrow window and wide window embeddings, to capture both semantic and syntactic based relationships. For each word in each embedding space we get the nearest *N* neighbors, and intersect them with the *N* nearest neighbors of the word in the other embedding space. We get these neighbors to obtain the weights first, and then use them again for the actual normalization in the next step. We discard candidates that have an edit distance above two, and obtain the individual edit operation weights through their normalized frequencies in the remaining candidates.

Word Mapping We use the learnt edit distance weights to score the normalization candidates mentioned above from the wide and nar-



$\hat{t}_1$
$\hat{t}_i$
$\hat{t}_L$
argmax
softmax
Bi-LSTM
$v_{w_1}^{narrow}$ $v_{w_1}^{wide}$ v_{c_1} v_{t_1}
$v_{w_i}^{narrow}$ $v_{w_i}^{wide}$ v_{c_i} v_{t_i}
$v_{w_L}^{narrow}$ $v_{w_L}^{wide}$ v_{c_L} v_{t_L}
v_1
v_i
v_L

Figure 4: The tagging architecture with the extended (wide and narrow) window embeddings.

row window embedding spaces, and further prune them based on their weighted edit distance. We select the candidate with the highest frequency in the text as the canonical equivalent.

Low Frequency Words Word2vec has a minimum count threshold for the words to be embedded. This value is tunable based on the used corpus. For the words below this threshold Word2vec does not guarantee a good vector representation, and discards them in the embedding model, so we can not use this normalization approach in this case. Instead, we use the weighted edit distances to score and map these words to more frequent cognates, on the character level only.[2]

Normalized Embeddings The pipeline so far results in a more consistent version of the text, which we use to learn the final embeddings upon. These embeddings are used as the pre-trained embeddings in the tagging architecture. This results in normalization at the embedding space level only, where the raw forms are still unmodified. The raw forms can be used for character-level noise reduction later in the tagging pipeline.

5 Experiments

5.1 Data

We use the "ARZ" (Maamouri et al., 2012) manually annotated EGY Arabic corpus, from the Linguistic Data Consortium (LDC), parts 1, 2, 3, 4 and 5. The corpus is based on the POS guidelines used by the LDC for Egyptian Arabic, and consists of about 160K words (excluding numbers and punctuations, 175K overall). The set of analyses for a given raw word includes the correct CODA orthography, in addition to the full morphological and POS annotations.

We use the splits suggested by Diab et al. (2013), comprised of a training set (Train) of about 134K words, a development set (Dev) of 20K words, and a blind testing set (Blind Test) of 21K words. The Dev set is used during the system development to assess design choices. The Blind Test set is used at the end to present the results.

The morphological analyzer we use in this paper is similar to the one used by Habash et al. (2013b). It is based on the SAMA (Graff et al., 2009), CALIMA (Habash et al., 2012b), and ADAM (Salloum and Habash, 2014) databases. EGY content, as in DA in general, contains many MSA cognates. The decision therefore to use all three analyzers was to maximize the recall of the overall analyzer.

We also use an in-house EGY monolingual corpus of about 410 million words, collected from online commentaries of blogs and social media platforms, to pre-train the word embeddings.

To better assess the notions of noise and ambiguity in the EGY dataset, we compare it to the Penn Arabic Treebank (PATB parts 1, 2 and 3) (Maamouri et al., 2004), which is commonly used for morphological disambiguation systems in MSA. MSA is also morphologically rich with high ambiguity levels, so it should provide a suitable reference for EGY. We sample an MSA data of size similar to the EGY dataset size, to be able to

[2] Instead of searching through the entire word space for each word to be normalized, which is computationally expensive, we pruned the search space by only looking at words sharing at least two consonants (in the same order) with it.

draw comparable comparison. Table 2 provides some statistics regarding both datasets. The average number of unique types per lemma (different types mapped to the same lemma encountered in the corpus) is relatively higher for the raw EGY content compared to MSA, at 2.7 vs 2.4. The average for the CODA-based EGY, however, is similar to MSA. This indicates that the normalized version of EGY has a similar sparsity as that for MSA, which is inherently less noisy. The difference in the ratio between raw and CODA EGY is a good indicator of the noise and inconsistency in the EGY dataset.

	EGY Raw	**EGY CODA**	**MSA**
Tokens	133,751	133,751	133,763
Inflected types	32,927	30,272	22,022
Lemmas	13,242	13,242	9,522
Avg types/lemma	2.7	2.4	2.4

Table 2: Dataset statistics showing tokens, types, and lemmas count in EGY and an MSA subset. Both from the Train set. The average inflections per lemma is calculated by counting the average unique types that map to the same lemma.

Regarding ambiguity, we calculated the average number of different analyses from the morphological analyzer for a given word in EGY at about 24 analyses per word (about 15 MSA, 6.5 DA, and 2.5 "no-analysis" analyses[3]), whereas for MSA it is around 12. This reflects the severe ambiguity of the EGY dataset compared with MSA in this context. Both noise and ambiguity issues make morphological tagging and disambiguation systems for EGY a very challenging task.

5.2 Experimental Setup

For the Bi-LSTM tagging architecture 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 0.002, and cross-entropy cost function. We use Tensorflow as the development environment.

The LSTM character embedding architecture uses two LSTM layers of size 100, and embedding size 50. The CNN architecture also uses embedding size 50, with filter widths ranging from one to six and max pooling strides of 50.

[3] The morphological analyzer has a backoff mode of "no-analysis" that provides a "proper noun" analysis to all word. The "proper noun" analysis can sometimes be cliticized, so some words might have multiple backoff analyses.

As for the neural language models for lemmatization and diacritization, we use two hidden layers of size 400 for lemmatization, and 600 for diacritization. We also use an input layer of size 300. We use Adam optimizer (Kingma and Ba, 2014) as the optimization algorithm, with learning rate of 0.002. We use TheanoLM (Enarvi and Kurimo, 2016) to develop the models.

The pre-trained word embeddings are of size 250, for both narrow and wide window embeddings. The wide window is set to five, whereas the narrow window is set to two (we experimented with a window of one but it performed slightly lower than a window of two). The number of nearest neighbors in the embedding space mapping experiment is 10 neighbors.

Metrics We use the following evaluation metrics for all systems:

- POS Accuracy (POS): The accuracy over the POS tag set comprised of 36 tags (Habash et al., 2013b).
- Morph Tags Accuracy (Morph Tags): The analysis and disambiguation accuracy over the 14 morphological features we work with, excluding lemmas and diacritized forms.
- Lemmatization Accuracy (Lemma): The accuracy of the lemma form of the words.
- Diacritization Accuracy (Diac): The accuracy of the diacritized form of the words.
- Full Analysis Accuracy (Full): The evaluation accuracy over the entire analysis, including the morphological features, lemma, and diacritized form.

5.3 Results

Table 3 shows the results of all systems for Dev, and Table 4 shows the results for the Blind Test set. We use the MADAMIRA results as the baseline.

Narrow embeddings seem to consistently outperform wide embeddings across all experiments. Regarding character embeddings, using both CNN and LSTM-based character embeddings improve the overall performance for both wide and narrow word embeddings, but LSTMs show consistent improvement over CNNs, which is in line with the conclusions of Heigold et al. (2017).

Embedding extension, through combining the wide and narrow window word embeddings, with

Model	Lemma	Diac	POS	Morph Tags	Full
MADAMIRA EGY (Baseline)	86.4	82.4	91.7	86.7	76.2
Bi-LSTM wide window embeddings	87.3	82.6	92.2	88.0	76.5
+ CNN character embeddings	87.3	82.5	92.6	88.2	76.6
+ LSTM character embeddings	87.4	82.5	92.6	88.3	76.7
+ Embedding space mapping	87.5	82.8	92.6	88.6	76.9
Bi-LSTM narrow window embeddings	87.5	82.9	92.3	88.0	76.7
+ CNN character embeddings	87.5	82.9	92.6	88.6	76.9
+ LSTM character embeddings	87.6	82.9	**92.9**	88.8	77.0
+ Embedding space mapping	87.4	82.8	92.7	88.7	76.9
Bi-LSTM wide+narrow embeddings and LSTM character embeddings	87.6	83.0	92.8	88.8	77.1
+ Embedding space mapping (Best System)	**87.7**	**83.2**	**92.9**	**88.9**	**77.4**
Relative error reduction of best result compared to baseline	9.6%	4.5%	14.5%	16.5 %	5.0%

Table 3: Results of the various systems over the Dev dataset, with MADAMIRA EGY (Pasha et al., 2014) as a state-of-the-art baseline.

Model	Lemma	Diac	POS	Morph Tags	Full
MADAMIRA EGY (Baseline)	87.3	83.3	91.8	86.9	77.3
Bi-LSTM wide window embeddings	87.5	83.1	92.6	87.9	77.4
+ CNN character embeddings	87.7	83.3	92.9	88.1	77.5
+ LSTM character embeddings	87.8	83.3	93.1	88.2	77.6
+ Embedding space mapping	87.8	83.5	93.4	88.9	78.0
Bi-LSTM narrow window embeddings	87.6	83.4	92.7	88.2	77.6
+ CNN character embeddings	87.8	83.6	93.3	88.8	78.0
+ LSTM character embeddings	88.0	83.6	93.5	89.1	78.2
+ Embedding space mapping	87.8	83.6	93.2	88.8	78.1
Bi-LSTM wide+narrow embeddings and LSTM character embeddings	87.9	83.5	93.1	88.6	78.0
+ Embedding space mapping (Best System)	**88.1**	**83.8**	**93.6**	**89.2**	**78.4**
Relative error reduction of best result compared to baseline	6.3%	3.0%	21.9%	17.6 %	4.9%

Table 4: Results of the various systems over the Blind Test dataset.

the LSTM-based character embeddings, significantly enhances the performance beyond the character embeddings alone for the wide embeddings. This is not the case though for narrow window embeddings. This highlights the significance of narrow embeddings for syntactic and morphological modeling, since the extension approach merely adds narrow window embedding capability to the wide window embeddings.

We observe the same pattern for the embedding space mapping approach for noise reduction against the narrow window embeddings. However, combining the extension with the embedding space mapping methods, along with the LSTM-based character embeddings, results in the best performing system. Both approaches seem to complement each other, as the accuracy exceeds any of the methods alone.

The result of the narrow window embeddings is particularly interesting, as it shows that to achieve a relatively good noise-robust morphological disambiguation accuracy, using narrow window embeddings should go a long way. Using more sophisticated, and computationally expensive, noise handling approaches, like embedding extension with embedding space mapping, should achieve even better results.

5.4 System Analysis

Oracle Conventional Orthography Experiment The availability of the manually annotated CODA equivalent of the EGY dataset allows for a deeper analysis of the noise effects on morphological disambiguation. We trained and tested the system using the CODA version of the data, as an oracle experiment of noise-reduced content. CODA-based content is not guaranteed to be noise-free, or be optimal for such syntactic and morphological tasks, but it should provide a good reference in terms of orthography-normalized content.

We train the model on the CODA-EGY training, and test it with the CODA-EGY Dev set. We use the same word pre-training dataset as before. We use LSTM-based character embeddings, and experiment with both wide and narrow embedding window. Table 5 shows the results for the CODA based modeling for Dev. The results are very similar to the best performing model in our earlier ex-

Model	Lemma	Diac	POS	Morph Tags	Full
Bi-LSTM wide window embeddings	87.4	82.5	92.6	88.3	76.7
Bi-LSTM narrow window embeddings	87.6	82.9	**92.9**	88.8	77.0
Bi-LSTM wide+narrow window embeddings+embeddings space mapping	**87.7**	**83.2**	**92.9**	**88.9**	**77.4**
(Oracle Experiment) CODA narrow window embeddings	87.9	83.3	93.0	89.1	77.4
(Oracle Experiment) CODA wide window embeddings	87.7	83.1	92.8	88.8	77.2

Table 5: Results of training and testing the system using the CODA-based Dev data, compared to the results of our system (taken from Table 3). All systems use LSTM-based character embeddings.

periments. These results indicate that our model is very close to the upper performance limit in terms of noise and inconsistency, and achieves noise-robust tagging and disambiguation.

The results for wide and narrow window contexts are also consistent with our earlier experiments, with narrow windowed contexts performing better across all evaluation metrics.

Manual Error Analysis

POS analysis We first analyze the overall error distribution in the POS tagging results. The most common POS error type is mistagging a nominal tag (Noun, Adjective, etc) with a different nominal tag, at 74% of the errors. Nominals include many very frequent tags, such as nouns and adjectives. The next most common error category is mistagging particles with other particles, at around 15%. Mistagging nominals with verbs is at around 4%. Several other low frequency errors cover the remaining 7%. To better understand the nature of the errors we manually checked a sample of 100 POS tagging errors. Almost 48% of them are gold errors, out of which our system gets 74% correct.

Lemma analysis We also manually checked a sample of 100 lemmatization errors. We observe that 30% of them are gold errors, 23% are the result of a wrong POS tag, 15% are acceptable MSA lemmas, 12% are due to minor and normally acceptable spelling issues, mainly the Hamza letter (glottal stops), and 6% are due to inconsistent diacritization. The MSA-related errors are due to the many MSA cognates in DA content. So providing an MSA-based analysis instead of an equivalent DA analysis can be acceptable for the purpose of this analysis. Hamza spelling variations, especially at the beginning of the word, are common in both DA and MSA written content.

Diacritization analysis We checked a sample of 100 diacritization errors. We observed more errors attributed to error propagation, as wrong POS tags and lemmas lead to many diacritization errors. The percentage of gold errors is only 17%, whereas MSA-cognate related errors are about 32%, POS related errors cover 13%, Hamza errors 11%, lemmatization errors include 7%, and the rest are mostly due to wrong case, gender, person tags, and other unidentified issues.

6 Conclusion and Future Work

We presented several neural morphological disambiguation models for EGY, and used several approaches for noise-robust processing. Our system outperforms a state-of-the-art system for EGY. We observed that character embeddings, combined with pre-trained word embeddings, provide a significant performance boost over the baseline. We showed that LSTM-based character embeddings outperform CNN-based models for EGY. We also showed that narrow window embeddings significantly outperform wide window embeddings for tagging. We also experimented with a normalization model on the word-level vectors, mapping non-canonical words to canonical neighbors through embedding space mapping. The results showed an additional improvement over the narrow window embeddings.

Future directions include exploring additional deep learning architectures for morphological modeling and disambiguation, especially joint and multitasking architectures. We also plan to explore knowledge transfer and adaptation models for more dialects with limited resources.

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 also gratefully acknowledged.

References

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*. San Diego, California, pages 11–16. http://www.aclweb.org/anthology/N16-3003.

Gustavo Aguilar, Suraj Maharjan, Adrian Pastor López Monroy, and Thamar Solorio. 2017. A multi-task approach for named entity recognition in social media data. In *Proceedings of the 3rd Workshop on Noisy User-generated Text*. pages 148–153.

Rania Al-Sabbagh and Roxana Girju. 2010. Mining the Web for the Induction of a Dialectical Arabic Lexicon. In *LREC*. Valetta, Malta.

Peter F Brown, Peter V Desouza, Robert L Mercer, Vincent J Della Pietra, and Jenifer C Lai. 1992. Class-based n-gram models of natural language. *Computational linguistics* 18(4):467–479.

Pradeep Dasigi and Mona Diab. 2011. Codact: Towards identifying orthographic variants in dialectal arabic. In *Proceedings of the International Joint Conference on Natural Language Processing*. Chiang Mai, Thailand.

Leon Derczynski, Alan Ritter, Sam Clark, and Kalina Bontcheva. 2013. Twitter part-of-speech tagging for all: Overcoming sparse and noisy data. In *Proceedings of the International Conference Recent Advances in Natural Language Processing RANLP 2013*. INCOMA Ltd. Shoumen, BULGARIA, pages 198–206. http://www.aclweb.org/anthology/R13-1026.

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

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

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.

Jennifer Foster, Özlem Çetinoğlu, Joachim Wagner, Joseph Le Roux, Stephen Hogan, Joakim Nivre, Deirdre Hogan, and Josef Van Genabith. 2011. #hardtoparse: Pos tagging and parsing the twitterverse. In *Proceedings of the 5th AAAI Conference on Analyzing Microtext*. AAAI Press, AAAIWS'11-05, pages 20–25. http://dl.acm.org/citation.cfm?id=2908630.2908634.

Kevin Gimpel, Nathan Schneider, Brendan O'Connor, Dipanjan Das, Daniel Mills, Jacob Eisenstein, Michael Heilman, Dani Yogatama, Jeffrey Flanigan, and Noah A. Smith. 2011. Part-of-speech tagging for twitter: Annotation, features, and experiments. In *ACL-HLT '11: Short Papers - Volume 2*. Association for Computational Linguistics, Stroudsburg, PA, USA, HLT '11, pages 42–47. http://dl.acm.org/citation.cfm?id=2002736.2002747.

Yoav Goldberg. 2016. A primer on neural network models for natural language processing. *J. Artif. Intell. Res.(JAIR)* 57:345–420.

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 International Conference on Language Resources and Evaluation (LREC)*.

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*. Montréal, Canada, pages 1–9.

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*. Ann Arbor, Michigan, pages 573–580.

Nizar Habash and Owen Rambow. 2007. Arabic Diacritization through Full Morphological Tagging. In *Proceedings of the 8th Meeting of the North American Chapter of the Association for Computational Linguistics/Human Language Technologies Conference (HLT-NAACL07)*.

Nizar Habash, Owen Rambow, and Ryan Roth. 2009. MADA+TOKAN: A toolkit for Arabic tokenization, diacritization, morphological disambiguation, POS tagging, stemming and lemmatization. In Khalid Choukri and Bente Maegaard, editors, *Proceedings of the Second International Conference on Arabic Language Resources and Tools*. The MEDAR Consortium.

Nizar Habash, Ryan Roth, Owen Rambow, Ramy Eskander, and Nadi Tomeh. 2013a. Morphological Analysis and Disambiguation for Dialectal Arabic. In *Proceedings of the 2013 Conference of NAACL-HLT*. Atlanta, GA.

Nizar Habash, Ryan Roth, Owen Rambow, Ramy Eskander, and Nadi Tomeh. 2013b. Morphological

Analysis and Disambiguation for Dialectal Arabic. In *Proceedings of NAACL-HLT*. Atlanta, Georgia, pages 426–432. http://www.aclweb.org/anthology/N13-1044.

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.

Georg Heigold, Guenter Neumann, and Josef van Genabith. 2017. An extensive empirical evaluation of character-based morphological tagging for 14 languages. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*. Association for Computational Linguistics, pages 505–513. http://www.aclweb.org/anthology/E17-1048.

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 Conference on Computational Natural Language Learning (CoNLL 2017)*. pages 421–431.

Mustafa Jarrar, Nizar Habash, Faeq Alrimawi, Diyam Akra, and Nasser Zalmout. 2016. Curras: an annotated corpus for the Palestinian Arabic dialect. *Language Resources and Evaluation* pages 1–31. https://doi.org/10.1007/s10579-016-9370-7.

Anna Jørgensen, Dirk Hovy, and Anders Søgaard. 2015. Challenges of studying and processing dialects in social media. In *Proceedings of the Workshop on Noisy User-generated Text*. pages 9–18.

Salam Khalifa, Nizar Habash, Dana Abdulrahim, and Sara Hassan. 2016a. A Large Scale Corpus of Gulf Arabic. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC)*. Portorož, Slovenia.

Salam Khalifa, Nasser Zalmout, and Nizar Habash. 2016b. Yamama: Yet another multi-dialect arabic morphological analyzer. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: System Demonstrations*. pages 223–227.

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

Xiaohua Liu, Shaodian Zhang, Furu Wei, and Ming Zhou. 2011. Recognizing named entities in tweets. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*. Association for Computational Linguistics, Stroudsburg, PA, USA, HLT '11, pages 359–367. http://dl.acm.org/citation.cfm?id=2002472.2002519.

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*. Cairo, Egypt, pages 102–109.

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

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, Curran Associates, Inc., pages 3111–3119.

Emad Mohamed, Behrang Mohit, and Kemal Oflazer. 2012. Annotating and Learning Morphological Segmentation of Egyptian Colloquial Arabic. In *Proceedings of the Language Resources and Evaluation Conference (LREC)*. Istanbul.

Olutobi Owoputi, Brendan O'Connor, Chris Dyer, Kevin Gimpel, Nathan Schneider, and Noah A. Smith. 2013. Improved part-of-speech tagging for online conversational text with word clusters. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, pages 380–390. http://www.aclweb.org/anthology/N13-1039.

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.

Slav Petrov and Ryan McDonald. 2012. Overview of the 2012 shared task on parsing the web. In *Notes of the First Workshop on Syntactic Analysis of Non-Canonical Language (SANCL)*. volume 59.

Barbara Plank, Anders Søgaard, and Yoav Goldberg. 2016. Multilingual part-of-speech tagging with bidirectional long short-term memory models and auxiliary loss. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, pages 412–418. https://doi.org/10.18653/v1/P16-2067.

Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*. ELRA, Valletta, Malta, pages 45–50. http://is.muni.cz/publication/884893/en.

Alan Ritter, Sam Clark, Mausam, and Oren Etzioni. 2011. Named entity recognition in tweets: An experimental study. In *Proceedings of the Conference on Empirical Methods in Natural*

Language Processing. Association for Computational Linguistics, Stroudsburg, PA, USA, EMNLP '11, pages 1524–1534. http://dl.acm.org/citation.cfm?id=2145432.2145595.

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*. Association for Computational Linguistics, Columbus, Ohio.

Houda Saadane and Nizar Habash. 2015. A Conventional Orthography for Algerian Arabic. In *ANLP Workshop 2015*. page 69.

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.

Vivek Kumar Rangarajan Sridhar. 2015. Unsupervised text normalization using distributed representations of words and phrases. In *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*. pages 8–16.

Andrew Trask, David Gilmore, and Matthew Russell. 2015. Modeling order in neural word embeddings at scale. *arXiv preprint arXiv:1506.02338* .

Lifu Tu, Kevin Gimpel, and Karen Livescu. 2017. Learning to embed words in context for syntactic tasks. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*. pages 265–275.

Houcemeddine Turki, Emad Adel, Tariq Daouda, and Nassim Regragui. 2016. A Conventional Orthography for Maghrebi Arabic. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC)*. Portorož, Slovenia.

Rob van der Goot, Barbara Plank, and Malvina Nissim. 2017. To normalize, or not to normalize: The impact of normalization on part-of-speech tagging. In *Proceedings of the 3rd Workshop on Noisy User-generated Text*. Association for Computational Linguistics, Copenhagen, Denmark, pages 31–39. http://www.aclweb.org/anthology/W17-4404.

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*. Association for Computational Linguistics, Copenhagen, Denmark, pages 704–713. https://www.aclweb.org/anthology/D17-1073.

I. Zribi, R. Boujelbane, A. Masmoudi, M. Ellouze Khmekhem, L. Hadrich Belguith, and N. Habash. 2014. A Conventional Orthography for Tunisian Arabic. In *In Proceedings of LREC*. Reykjavik, Iceland.