Arabic Diacritization with Recurrent Neural Networks

Yonatan Belinkov, James Glass
Massachusetts Institute of Technology
Computer Science and Artificial Intelligence Laboratory
{belinkov, glass}@mit.edu

Abstract

Hebrew and Arabic are typically written without diacritics, leading to ambiguity and posing a major challenge for core tasks like speech recognition. Previous approaches to automatic diacritization used a variety of machine learning techniques. However, they typically rely on existing tools like morphological analyzers and therefore cannot be easily extended to new genres and languages. We develop a recurrent neural network with long short-term memory layers that is trained solely from diacritized text yet rivals state-of-the-art methods that have access to external resources.

1 Introduction

Hebrew, Arabic, and other languages based on Arabic script usually represent only consonants in writing and do not mark vowels. In such writing systems, diacritics (Hebrew Nikkud, Arabic Harakat) are used for short vowels, gemination, and other phonetic units. In practice, diacritics are usually restricted to language teaching or to religious texts. Faced with a non-diacritized word, readers infer missing diacritics based on their prior knowledge and the context of the word in order to resolve ambiguities. For example, Maamouri et al. (2006) mention several types of ambiguity for the Arabic string علم Elm, and a morphological analyzer produces at least 13 different diacritized forms (a subset is shown in Table 1).[1]

Word	Gloss
Ealima	he knew
Eulima	it was known
Eal~ama	he taught
Eilomu	knowledge (def.nom)
...	...
EalamK	flag (indef.gen)

Table 1: Possible diacritized forms for علم Elm.

The ambiguity in Arabic orthography presents a problem for many language processing tasks, including acoustic modeling for speech recognition, language modeling, and morphological analysis. Automatic methods for diacritization aim to restore diacritics in a non-diacritized text. While earlier work used rule-based methods, more recent studies attempted to learn a diacritization model from diacritized text. A variety of methods have been used, including hidden Markov models, finite-state transducers, and maximum entropy – see the review in (Zitouni and Sarikaya, 2009) – and more recently, deep neural networks (Al Sallab et al., 2014). In addition to learning from diacritized text, these methods typically rely on external resources such as part-of-speech taggers and morphological analyzers (Habash and Rambow, 2007). However, building such resources is labor-intensive and cannot be easily extended to new languages, dialects, and domains.

In this work, we propose a diacritization method based solely on diacritized text. We treat the problem as a sequence classification task, where each character has a corresponding diacritic label. The sequence is modeled with a recurrent neural network whose input is a sequence of characters and whose output is a probability distribution over the diacritics. Any RNN architecture can be used in this framework; here we focus on long short-term memory (LSTM) networks, which had recent success in a number of tasks. We experiment with several architectures and show that we can approach the state-of-the-art, without relying on external resources.

2 Approach

We define the following sequence classification task, similarly to (Zitouni and Sarikaya, 2009).

[1] Arabic transliteration follows the Buckwalter scheme: http://www.qamus.org/transliteration.htm.

Let $\mathbf{c} = (c_1, ..., c_k)$ denote a sequence of characters, where each character c_i is associated with a label l_i. A label may represent 0, 1 or more diacritics, depending on the language. Assume further that each character c in the alphabet is represented as a real-valued vector x_c. This letter embedding may be learned during training or fixed.

Our neural network has the following structure:

- Input layer: mapping the letter sequence $\mathbf{c}$ to a vector sequence $\mathbf{x}$.
- Hidden layer(s): mapping the vector sequence $\mathbf{x}$ to a hidden sequence $\mathbf{h}$.
- Output layer: mapping each hidden state vector h_i to a probability distribution over labels.

During training, each sequence is fed into this network to create a prediction for each character. As errors are back-propagated down the network, the weights at each layer are updated. During testing, the learned weights are used in a feed-forward step to create a prediction over the labels. We always take the best predicted label for evaluation.

2.1 Implementation details

The input layer maps the character sequence to a sequence of letter vectors (initialized randomly). It also stacks previous and future letter vectors, so the model can learn contextual information. We experiment with several types of hidden layers, from one feed-forward to multiple bidirectional LSTM layers. We also add a linear projection layer after the input layer, which has the effect of learning a new representation for the letter embeddings. The output layer is a Softmax over labels.

Training is done with stochastic gradient descent with momentum, optimizing the cross-entropy objective function. Our implementation is based on Currennt (Weninger et al., 2015).

3 Experiments

We test our approach on Arabic data extracted from the Arabic treebank, following the train/dev/test split in (Zitouni and Sarikaya, 2009).[2] As Table 2 shows, LSTM models preform better than traditional feed-forward networks, even when the latter have a similar number of parameters. A 2-layer LSTM achieves the best results. On the test set (Table 3), this model beats the lexical variant of Zitouni and Sarikaya (2009) and approaches the performance of their best model, which used also a segmenter and part-of-speech tagger. This shows that our model can effectively learn to diacritize without relying on any resources other than diacritized text.

	DER	# parameters
Feed-forward	11.76	63K
Feed-forward (large)	11.55	908K
LSTM	6.98	838K
B-LSTM	6.16	518K
2-layer B-LSTM	5.77	916K

Table 2: Results (DER) on the Arabic dev set.

MaxEnt (only lexical)	8.1
MaxEnt (full)	5.1
2-layer B-LSTM	5.61

Table 3: Results (DER) on the Arabic test set. MaxEnt results from (Zitouni and Sarikaya, 2009)

Finally, we are currently incorporating our diacritization system in a speech recognizer. Recent work has shown improvements in Arabic speech recognition by diacritizing with MADA (Al Hanai and Glass, 2014). Since creating such tools is a labor-intensive task, we expect our diacritization approach to promote the development of speech recognizers for other languages and dialects.

[2]Other papers report work on a train/test split, without a dedicated dev set (Al Sallab et al., 2014); we will test our model in this setting in future experiments.

References

Tuka Al Hanai and James Glass. 2014. Lexical Modeling for Arabic ASR: A Systematic Approach. In *Proceedings of INTERSPEECH*.

Ahmad Al Sallab, Mohsen Rashwan, Hazem M. Raafat, and Ahmed Rafea. 2014. Automatic Arabic diacritics restoration based on deep nets. In *Proceedings of the EMNLP 2014 Workshop on Arabic Natural Language Processing (ANLP)*.

Nizar Habash and Owen Rambow. 2007. Arabic Diacritization through Full Morphological Tagging. In *Proceedings of HLT-NAACL*.

Mohamed Maamouri, Ann Bies, and Seth Kulick. 2006. Diacritization: A challenge to arabic treebank annotation and parsing. In *Proceedings of the British Computer Society Arabic NLP/MT Conference*.

Felix Weninger, Johannes Bergmann, and Björn Schuller. 2015. Introducing currennt: The munich open-source cuda recurrent neural network toolkit. *JMLR*, 16:547–551.

Imed Zitouni and Ruhi Sarikaya. 2009. Arabic Diacritic Restoration Approach Based on Maximum Entropy Models. *Comput. Speech Lang.*, 23(3).