Osama Hamed and Torsten Zesch

A Survey and Comparative Study of Arabic Diacritization Tools

Abstract

Modern Standard Arabic, as well as other languages based on the Arabic script, are usually written without diacritics, which complicates many language processing tasks. Although many different approaches for automatic diacritization of Arabic have been proposed, it is still unclear what performance level can be expected in a practical setting. For that purpose, we first survey the Arabic diacritization tools in the literature and group the results by the corpus used for testing. We then conduct a comparative study between the available tools for diacritization (Farasa and Madamira) as well as two baselines. We evaluate the error rates for these systems using a set of publicly available, fully-diacritized corpora in two different evaluation modes. With the help of human annotators, we conduct an additional experiment examining error categories. We find that Farasa is outperforming Madamira and the baselines in both modes.

1 Introduction

Automatic diacritization is the task of restoring missing diacritics in languages that are usually written without diacritics like Arabic; or in languages that have diacritically marked characters in their orthography like Dutch, German, Hungarian, Lithuanian, or Slovene (Acs and Halmi, 2016). The challenge is that many words have different meanings depending on their diacritization, which can only be resolved by the context and proper knowledge of the grammar (Rashwan et al., 2011).

Restoring diacritics is an important task, as diacritized texts are crucial for many Natural Language Processing (NLP) applications, including automatic speech recognition (Zitouni et al., 2006; Ananthakrishnan et al., 2005), statistical machine translation (Diab et al., 2007a), text-to-speech (Shaalan et al., 2009), text analysis, information retrieval (Azmi and Almajed, 2015), and the normalization and analysis of social media texts (Čibej et al., 2016). Diacritized text is also important at the early stages of language learning and for second language (L2) learners.

Although there is a large body of research on the topic, only very few tools are freely available, and it is still unclear what performance level can be expected in a practical setting. We aim at a fully reproducible comparison and will thus only include tools that are freely available and can be integrated into our comparison pipeline. To the best of our knowledge, there are currently only two tools that fulfill these requirements:

Madamira (Pasha et al., 2014) and *Farasa* (Darwish and Mubarak, 2016). There exist some additional tools like Mishkal[1], which is only available as a web service, and ArabicDiacritizer[2], which only works in a Windows environment. Additionally, both tools limit the size of the input text and cannot be easily integrated in our Java-based comparison framework. For the same reasons of ensuring reproducibility, we only use training and test data that is publicly available without license fees.

In this paper, we conduct a comparative study between the available tools for diacritization using a reasonable amount and variety of test data in two evaluation modes: strict and relaxed. While the strict mode expects the diacritics to be exactly the same as in gold standard text, the relaxed mode normalizes the texts (output and gold standard) to hold a specific (smaller) ratio of diacritics. Thus, the relaxed mode does not punish a tool that only provides partial diacritization. In order to put the results into perspective, we implement two strong baselines: a dictionary lookup system and one based on character-based sequence labeling. The first baseline labels each word using the diacritized form that appears most often in the training set. The second baseline treats diacritization as a sequence classification problem using conditional random fields (CRF). We report the error rates for the baselines and state-of-the-art systems using diacritized text from Classical Arabic (Quran and Tashkeela corpora) and contemporary writing (RDI corpus) in both evaluation modes.

2 Linguistic Background

Languages based on the Arabic script usually represent only consonants in their writing and do not mark the short vowels (Belinkov and Glass, 2015). The Arabic script (الخط العربي) is written from right to left and contains two classes of symbols for writing words: letters and diacritics (Habash and Rambow, 2007; Habash, 2010). Figure 1 shows the non-diacritized and the diacritized versions of the sentence "*The Arabic script*".

without diacritics	الخط العربي
with diacritics	اَلْخَطّ اَلْعَرَبِيُّ

Figure 1: Example of an Arabic sentence without and with diacritics (eng: *The Arabic script*).

[1] http://tahadz.com/mishkal
[2] https://sourceforge.net/projects/arabicdiacritizer/

Letters The Arabic alphabet has 29 letters, those include three long vowels (Alif (ا), Waw (و) and Yeh (ي)), 25 consonants, and the Hamza (glottal stop).

Diacritics The diacritics are optional. If present they appear as small strokes that are placed above or below the letter, such as ـَ ـُ ـِ.

Diab et al. (2007a) group these diacritical marks into three categories: vowel, nunation, and Shadda (gemination). The vowel diacritics refer to the three short vowels (Fatha (ـَ) /a/[3], Damma (ـُ) /u/, and Kasra (ـِ) /i/) and a diacritic indicating the absence of any vowel (Sukun) (Bouamor et al., 2015). Nunation diacritics indicate a short vowel followed by a non-written sound of the Arabic letter (ن) /n/. The Nunation diacritics look like a doubled version of their corresponding short vowels (Habash, 2010). They are named in Arabic as such: Fathatan, Dammatan, Kasratan.[4] For example, (دٌ) is pronounced /dun/ and transliterated as "duN".[5] The gemination mark (Shadda) is a consonant-doubling diacritical (دّ) "d∼". Shadda can be combined with diacritics from the other two categories, which results in a total of thirteen diacritical marks. For instance, (دُّ) "d∼u" and (دٌّ) "d∼uN".

There are general rules for diacritizating Arabic text. For example, Shaddah and Sukun cannot follow a word-initial letter, whereas Tanween appears only at word-final position (Elshafei et al., 2006). Table 1 exemplifies the shapes of diacritics in conjunction with the Arabic letter (د) /d/.

Some of the diacritics vary depending on syntactic conditions (case-related), and some vary to indicate semantic differences. Functionally, diacritics fall into two types: lexical and inflectional diacritics (Diab et al., 2007a). The lexical diacritics distinguish between two lexemes; for example, "kAtib" (كَاتِب), meaning "writer," and "kAtab" (كَاتَب), meaning "to correspond". The inflectional diacritics distinguish different inflected forms of the same lexeme. For example, the final (last-letter) diacritic in "kitaAbu" (كِتَابُ), meaning "book," is *Damma* to indicate the nominative case (verb subject) and the final diacritic in "kitaAba" (كِتَابَ) is *Fatha* to indicate the accusative case (verb object) of the same word.

In unicode, the diacritics are presented as additional characters, so the diacritized word is longer than the non-diacritized word. For example, the diacritized word "Eal∼ama" (عَلَّمَ) has seven unicode characters, whereas the bare form "Elm" (علم) has only three.

[3] International Phonetic Alphabet (IPA)
[4] Dual feminine nouns that indicate two Fathas, two Dammas and two Kasras respectively.
[5] Buckwalter encoding (Buckwalter, 2004) is used exclusively in the paper.

Type	Diacritic Mark	Name	Transl.	IPA	Word Position
Short vowels	ـَ	Fatha	a	/a/	Any
	ـُ	Damma	u	/u/	Any
	ـِ	Kasra	i	/i/	Any
	ـْ	Sukun	o	Ø	Any
Nunation	ـًا	Tanween Fath	F	/an/	End
	ـٌ	Tanween Damm	N	/un/	End
	ـٍ	Tanween Kasr	K	/in/	End
Gemination	ـّ	Shadda	∼	:	Any

Table 1: Types of Arabic diacritics

2.1 Diacritization Levels

The level of diacritics refers to the number of diacritical marks presented on a word to avoid text ambiguity for human readers. Even in non-diacritized newswire text, 1.6% of all words have at least one diacritic indicated by their author to guide the reader with disambiguation (Habash, 2010). Ahmed and Elaraby (2000) grouped the diacritization levels into three levels (full, half, and partial):

Full All the letters are given appropriate diacritics. This applies to classical Arabic (CA), as in religion-related books, and at early stages of language learning, such as in children's books.

Half Only the morphological-independent letters are diacritized. In other words, all the letters of a word, except those that depend on the syntactic analysis of the word, are diacritized. For example, the word "wldh" (ولده), meaning "his son" consists of two clitics "wld+h" = (ه) + (ولد), i.e. the stem "wld" and the possessive pronoun "h" as suffix. With the half diacritization, it would be written like (وَلَده) instead of (وَلَدُهُ). This means that the diacritic was dropped from the pronoun "h" (ه) (morphology-dependent) and from the stem last letter "d" (د) (syntactic).

Partial Any other setting where one letter or a subset of letters is diacritized. While studying the impact of diacritization on statistical machine translation, Diab et al. (2007a) proposed to divide this level into four sub-levels for use with inflectional and lexical diacritics. A special case of partial diacritization is to drop the short

Type	Bare Form	Diacritized	Gloss / Transliteration
POS	علم	عِلم	Science / Eilom
		عَلَم	Flag / Ealam
		عَلِمَ	He knew / Ealima
		عُلِم	It was known / Eulim
		عَلَّمَ	He taught / Eal∼ama
Syntactic	مدير البنك الجديد	مُديرَ البَنكِ الجَديدِ	the manager of the new bank / mudyra Albanki Aljadydi
		مُديرَ البَنكِ الجَديدَ	the new bank manager / mudyra Albanki Aljadyda
Structure	ولي	وَلِي	and for me / waliy
		وَلِيّ	a pious person favored by God / waliy∼

Table 2: Types of ambiguity caused by missing diacritics

vowels and Sukun. For example, the short vowel is dropped from the letter that precedes a long vowel with similar sound like when *Fatha* is dropped from a letter if followed by an *Alef* (ا). Additionally, the Arabic definite article ال has only two diacritization possibilities depending on the preceding letter. The *Alef* is always diacritized with *Fatha*, and the *Lam* (ل) either has Sukun or has no diacritics.

2.2 Ambiguity

Writing Arabic without diacritics introduces three types of ambiguity (Azmi and Almajed, 2015). The first is part-of-speech (POS) tagging ambiguity (Maamouri et al., 2006). This is the case with the words that have the same spelling and POS tag but a different lexical sense, or words that have the same spelling but different POS tags and lexical senses (homograph ambiguity) (Farghaly and Shaalan, 2009). Second, there is ambiguity on the grammatical level (syntactic ambiguity). Sentences and phrases can be interpreted in more than one way, and diacritics are the only means to resolve ambiguity (Maamouri et al., 2006). The third is internal word structure ambiguity, such as when Arabic words are segmented in different ways. The agglutination property of Arabic might produce a problem that can only be resolved using diacritics. Table 2 summarizes the aforementioned types of ambiguity with excerpted examples from (Metwally et al., 2016; Farghaly and Shaalan, 2009).

Corpus	Description	Availability	# of tokens
Quran	Religious	Free	78 000
RDI	Religious/Modern	Free	20 000 000
Tashkeela	Religious	Free	60 000 000
ATB	News	Commercial	1 000 000
WikiNews	News	Free	18 300

Table 3: Overview of diacritized corpora.

3 State-of-the-Art Arabic Diacritization

In this section, we present the diacritized datasets usually used for evaluation and then give an overview of the results on different corpora that have so far been obtained using the standard evaluation metrics.

3.1 Datasets

Generally, the currently available diacritized corpora are limited to classical texts (usually religious or Arabic poetry), such as the Holy Quran, RDI, and Tashkeela on the one side, and newswire corpora, such as the Arabic Penn Treebank (ATB) from the Linguistic Data Consortium (LDC) on the other side, as shown in Table 3.

Quran The small diacritized Quranic corpus is part of Tanzil[6] project. It contains more than 78 thousand tokens that comes in a UTF-8 encoded text file. The file has no Arabic punctuation marks, and every Quranic verse appears in a separate line.

RDI The corpus was collected by the RDI[7] company for use in the field of automatic diacritization. It is composed of diacritized texts, which are mainly gathered from classical Arabic books with a small percentage from contemporary Arabic writing (modern books). Overall, it contains 20 million tokens. Our experiments are based on the subset of modern books, a collection of 12 books from the late 1990's.

Tashkeela The corpus contains more than 60 million diacritized tokens (Zerrouki and Balla, 2017). It is a collection of 84 Islamic religious heritage books. The books are provided in HTML format, encoded in CP1256 Windows Arabic. It can be downloaded under GPL license.[8]

[6] http://tanzil.net/download/
[7] http://www.rdi-eg.com/RDI/TrainingData/
[8] https://sourceforge.net/projects/tashkeela/

ATB Much of the previous work on diacritization relied on using the ATB. LDC's Arabic Penn Tree Bank (ATB) consists of distinct newswire stories collected from different news agencies and newspapers, including the Agence France-Presse (AFP), Al-Hayat, and An-Nahar newspapers (Maamouri et al., 2004, 2006, 2009). It contains about 1 million tokens. Though ATB is invaluable for many tasks, such as POS tagging and parsing, it is sub-optimal for diacritization (Darwish et al., 2017).

WikiNews Darwish et al. (2017) used a new test set composed of 70 WikiNews articles (the majority are from 2013 and 2014) that cover a variety of themes, namely: politics, economics, health, science and technology, sports, arts, and culture. The articles are evenly distributed among the different themes (10 per theme). The corpus contains 18,300 words.

3.2 Evaluation Metrics

In the literature, two standard evaluation metrics are used almost exclusively to measure systems performance (Rashwan et al., 2011; Said et al., 2013). It can either be expressed in terms of error rates on the character or on the word level. The smaller the error rate, the better the performance.

DER Diacritization Error Rate (DER) is the proportion of letters which are incorrectly labeled with diacritics. The following assumptions are made: (i) each letter or digit in a word is a potential host for a set of diacritics, and (ii) all diacritics on a single letter are counted as a single binary choice. The DER can be calculated as follows:

$$DER = (1 - \frac{|T_S|}{|T_G|}) \cdot 100 \quad (1)$$

where $|T_S|$ is the number of letters assigned correctly by the system, and T_G is the number of diacritized letters in the gold standard text.

WER Word Error Rate (WER) is the percentage of incorrectly diacritized white-space delimited words. In order to be counted as incorrect, at least one letter in a word must have a diacritization error. All words are counted, including numbers and punctuation.

While the diacritization techniques work relatively well on lexical diacritics (located on word stems), they are much less effective for inflectional diacritics (typically at

Diacritization Tool	Word Letters					
	A	**l**	**E**	**r**	**b**	**y**
Gold	a	o	a	a	i	∼u
Tool 1	-	o	a	a	i	∼u
Tool 2	-	o	a	a	-	∼u
Tool 3	-	-	-	a	-	∼u
Tool 4	-	-	a	a	-	∼u
in relaxed evaluation?	No	No	No	Yes	No	Yes

Table 4: The normalization of diacritics for comparison in relaxed evaluation mode.

word-final position) (Habash et al., 2007). In most cases, the last letter indicates the case ending. However, in some cases as with plural masculine nouns (جمع المذكر السالم) and dual masculine and feminine nouns (المثنى) the suffixes substitute the diacritics. The suffixes are added to the word to indicate case and number. For example, the suffixes (ون) or (ان) are added to the word to indicate plural masculines and dual masculine or feminine nouns in accusative case respectively. However, the suffix (ين) is added to the word to indicate plural masculines and dual masculine or feminine nouns in nominative and dative cases. Assigning the correct case can often only be decided using a wider context, thus diacritization tools usually perform worse on the last letter compared to the other positions in the word (Habash et al., 2007). It is thus usual to also report a variant of the above two mentioned metrics that ignore the last letter (assumed to have no syntactic diacritics), denoted as **DER-1** and **WER-1**.

3.3 Evaluation Modes

When comparing multiple tools, we distinguish two different evaluation modes:

Strict Mode Whenever a letter has a set of diacritics in the gold standard text, a diacritization tool is expected to predict this set exactly. This evaluation mode is most often used and gives an advantage to tools providing full diacritization.

Relaxed Mode This evaluation mode gives an advantage to tools that only output diacritics when being confident about the results. This might be useful for half or partial diacritization settings, e.g. the tools that drop the default diacritics. This is not so useful for other settings, e.g. full diacritization in children books.

In order to provide a fair comparison between multiple tools, the relaxed evaluation

Test Corpus	Size (10^3)	Approach	All Diacritics		Ignore Last	
			DER	WER	DER-1	WER-1
ATB (Parts 1–3)	144	(Nelken and Shieber, 2005)	12.8	23.6	6.5	7.3
	52	(Zitouni et al., 2006)	5.5	18.0	2.5	7.9
	52	(Habash and Rambow, 2007)	4.8	14.9	2.2	5.5
	613	(Schlippe et al., 2008)	4.3	19.9	1.7	6.8
	116	(Schlippe et al., 2008)	4.7	21.9	1.9	8.4
	16	(Alghamdi et al., 2010)	13.8	46.8	9.3	26.0
	52	(Rashwan et al., 2011)	3.8	12.5	1.2	3.1
	37	(Abandah et al., 2015)	2.7	9.1	1.4	4.3
	52	(Metwally et al., 2016)	-	13.7	-	-
Quran	1	(Elshafei et al., 2006)	4.1	-	-	-
	76	(Abandah et al., 2015)	3.0	8.7	2.0	5.8
Tashkeela	1902	(Hifny, 2012)	-	8.9	-	3.4
	272	(Abandah et al., 2015)	2.1	5.8	1.3	3.5
Tashkeela+RDI	199	(Bebah et al., 2014)	7.4	21.1	3.8	7.4
WikiNews	18	(Pasha et al., 2014)	5.4	19.0	1.9	6.7
	18	(Rashwan et al., 2015)	4.3	16.0	1.0	3.0
	18	(Belinkov and Glass, 2015)	7.9	30.5	3.9	14.9
	18	(Darwish et al., 2017)	3.5	12.8	1.1	3.3

Table 5: Performance of Arabic diacritization systems grouped by test corpus

mode only takes into account cases where all tools under consideration return a diacritic for a given letter. Table 4 gives an example.

3.4 Overview of Diacritization Results

The work on Arabic diacritization goes back quite a long time (El-Sadany and Hashish, 1989) and many different approaches have been proposed including hidden Markov model (Elshafei et al., 2006), n-gram language models (Hifny, 2012; Alghamdi et al., 2010), statistical machine translation (Schlippe et al., 2008), finite state transducers (Nelken and Shieber, 2005), maximum entropy (Zitouni et al., 2006), and deep learning (Rashwan et al., 2015; Abandah et al., 2015; Belinkov and Glass, 2015).

Additionally, many researchers have proposed to improve classification with morphological analysis (Habash and Rambow, 2005; Rashwan et al., 2011; Bebah et al., 2014; Metwally et al., 2016) and the standard n-gram language model. A recent approach by Darwish et al. (2017) employed a Viterbi decoder and SVM-rank to properly guess words diacritization.

ID	Corpus	# words (10^3)	∅ chars per word	Words / sentence
Q	Quran	78	4.25	12.6
T	Tashkeela	100	4.11	14.7
R	RDI	100	4.47	34.1

Table 6: Statistics of corpora sub-datasets used in this study.

Comparison Table 5 gives an overview of the reported results from the literature. The results are grouped by the corpus that was used for testing in order to allow for a fair comparison. There is a major drawback with these reported results: they do not follow a well-established framework for testing. For example, most numbers are still not directly comparable because they were obtained using different test sets. Moreover, some works used a fixed test set without performing any cross-validation, which further limits the weight that should be put on those numbers. The only exception to this is the last block of results, where Darwish et al. (2017) compared their system with other systems using the *WikiNews* test set. Under this controlled setting, their system outperforms all other systems regarding DER and WER. If we ignore the case-endings, the Rashwan et al. (2015) system performs best.

As most of the systems from the literature are not freely available, we have no way of directly comparing them. In this paper, we establish a comparative study that only includes the systems and corpora that are freely available in a controlled settings.

4 Experimental Setup

In this section, we present our experimental setup: used data, baselines, diacritization tools, and evaluation metrics.

The experiments were carried out using DKPro TC, the open-source UIMA-based framework for supervised text classification (Daxenberger et al., 2014). The baseline experiments were conducted as ten-fold cross-validation, reporting the average over the ten folds.

4.1 Datasets

Table 6 shows the statistics for the experimental sub-datasets (punctuation marks are not counted). All the experiments use a general setup for test sample-size: 78K, 100K, and 100K drawn from the Quran, RDI, and Tashkeela respectively.

Data Preprocessing The Quran text requires no special preprocessing. However, the files from Tashkeela and RDI contain Quranic symbols like the Dagger Alif (a small Alif quite common in Quranic Arabic (Dukes and Habash, 2010)) or English letters. In order to prepare those corpora for training and testing purposes, the following preprocessing steps are performed: (i) convert them from HTML to plain text files that have one sentence per line, (ii) clean the files by removing the Quranic symbols and words written in non-Arabic letters, and (iii) normalize the Arabic text by removing extra white spaces and Tatweel symbols.[9] For example, "qAl" (قـال), meaning "he said" has Tatweel, whereas قال has no Tatweel.

4.2 Baselines

We implemented two baselines: a simple dictionary lookup approach and a sequence labeling approach.

Dictionary Lookup This baseline labels each word with the diacritized form that appears most often in the training corpus. Words that are not found in the dictionary are not diacritized.

Sequence Labeling We treat diacritization as a sequence labeling problem and propose a baseline solution using conditional random fields (Lafferty et al., 2001). Given a sentence (set of non-diacritized words) separated using white-space delimiters, each word in the sentence is a sequence of characters, and we want to label each letter with its corresponding labels from the diacritics set $D = (d_1, ..., d_N)$. We represent each word as an input sequence $X = (x_1, ..., x_N)$ where we need to label each consonant in X with the diacritics that follow this consonant. Note that an Arabic letter has a maximum of two diacritics, and if it has two, then one of them is always Shadda. Shadda might accompany all diacritics except Sukun, so in total we have 14 labeling possibilities (including the 'no diacritic' option). Thus, in order to diacritize sequence X, we must find its labeling sequence Y (usually of word length) derived from D.
A word might have more than one valid labeling. The word "ktAb" (كتاب) represented as (k, t, A, b), can be labeled with $Y_1 = (i, a, o, u)$ or $Y_2 = (i, a, o, a)$ resulting in the diacritized words "kitaAobu" and "kitaAoba" respectively.

Our features are character n-grams language models (LMs) in sequence labeling approach. The features extractor selects the character-level features relevant to diacritics

[9] Tatweel are used to stretch words to indicate prominence or simply to force vertical justification (Habash, 2010).

from annotated corpora. It collects the diacritics on previous, current and following character and up to the 6th character.

Note that the out-of-vocabulary (OoV) rate of this approach is zero as it is able to provide a sequence of diacritics for arbitrary unknown words.

4.3 Diacritization Tools

To the best of our knowledge, the only tools that can be tested on large corpora and are easily integrated with Java frameworks are *Madamira* and *Farasa*.

Madamira Madamira (Pasha et al., 2014) improves upon its two ancestors MADA (Habash et al., 2009) and AMIRA (Diab et al., 2007b) with a Java implementation that is more robust, portable, extensible, and faster. Arabic processing with Madamira includes automatic diacritization, lemmatization, morphological analysis and disambiguation, part-of-speech tagging, stemming, glossing, tokenization, base-phrase chunking, and named-entity recognition. Madamira makes use of fast, linear SVMs implemented using *Liblinear* (Fan et al., 2008).

Madamira was trained on the training portion of ATB (parts 1, 2 and 3). There are two varieties of Madamira. The first integrates the public version of Arabic morphological analyzer (AraMorph).[10] The second integrates the Standard Arabic Morphological Analyzer (SAMA) and its recommended database (Graff et al., 2009).[11]

Our experiments are carried out using the SAMA enabled version of Madamira *v2.1*. Madamira was used to diacritize the test sequences from the three corpora. As the resulting diacritized text is encoded using Buckwalter transliteration, it is necessary to decode it into Arabic text. We compare the mapped Arabic text with a gold standard sequence and then calculate the different metrics.

Farasa Farasa (Darwish and Mubarak, 2016) is an open-source tool, written entirely in native Java. Farasa consists of a segmentation/tokenization module, POS-tagger, Arabic text diacritizer, and dependency parser. Its approach is based on SVM-ranking using linear kernels. Farasa matches or outperforms state-of-the-art Arabic segmenters (Darwish and Mubarak, 2016) and diacritizers.

Corpus	Approach	OoV	All Diacritics		Ignore Last	
		rate	DER	WER	DER-1	WER-1
Quran	Dict. Lookup	11.8	19.7	27.5	16.1	16.8
	Sequence Labeling	0.0	21.4	28.3	9.0	19.9
	Madamira	3.4	21.1	36.7	15.4	20.9
	Farasa	0.3	**12.2**	**19.0**	**8.9**	**9.5**
RDI	Dict. Lookup	13.3	26.6	31.8	19.7	22.5
	Sequence Labeling	0.0	24.9	37.0	15.4	22.4
	Madamira	2.1	17.8	28.4	13.1	14.2
	Farasa	0.1	**10.5**	**15.7**	**6.7**	**7.6**
Tashkeela	Dict. Lookup	13.4	26.9	32.2	19.9	22.7
	Sequence Labeling	0.0	24.9	37.0	15.7	22.3
	Madamira	2.2	17.9	28.6	13.1	14.2
	Farasa	0.1	**10.6**	**15.9**	**6.8**	**7.7**

Table 7: Error rates in strict evaluation mode. The "OoV" rate refers to the ratio of tokens that were not diacritized by the system.

5 Results

We now report the results of our diacritization experiments using first 'strict' and then 'relaxed' evaluation.

5.1 Strict Evaluation

Table 7 gives an overview of our evaluation results in *strict* mode. The results are grouped by the corpus that was used for testing. Note that the OoV column refers to the ratio of tokens that got "No Analysis" and thus no diacritization by the system.

In general, the error rates are rather high. With keeping in mind that the reported results are non-comparable, none of the methods (including the two well-known state-of-the-art systems) comes even close to the numbers in Table 5. It is likely that many approaches do not use strict evaluation mode when reporting results, even if it is the most comparable setup. When indirectly competing with other published results, the numbers obtained in that way are just not competitive.

Looking at individual results, Farasa outperforms all other methods under all metrics. For the remaining three approaches, there is no clear trend, but it should be noted that the baselines perform surprisingly well even if they make no real attempt at resolving ambiguity. Sequence labeling doesn't take context into account and the dictionary

[10] `http://www.nongnu.org/aramorph/`
[11] Catalog number LDC2009E73

	Diacritics per letter			Diacritized letters per word		
Approach	**Quran**	**RDI**	**Tashkeela**	**Quran**	**RDI**	**Tashkeela**
Gold	.84	.83	.83	.78	.77	.77
Dict. Lookup BL	.84	.84	.82	.78	.77	.77
Seq. Labeling	.82	.78	.78	.77	.74	.74
Madamira	.55	.59	.61	.51	.54	.56
Farasa	.58	.58	.61	.55	.54	.58

Table 8: Average number of diacritics per letter and average number of diacritized letters per word

lookup makes a majority class decision for each ambiguous token. We suspect that many tokens within a domain are not ambiguous and the repetitious nature of the religious texts increases the effect.

Table 7 also shows the out-of-vocabulary rate for each approach. As expected, the dictionary lookup baseline has a rather high rate and sequence labeling has no out-of-vocabulary tokens at all, because it always returns one of the possible diacritization patterns. For all corpora, Farasa has a lower OoV rate than Madamira.

When looking into individual OoV examples, we find that in some cases the tools do not return any analysis. However, in some cases they change the input token instead of just adding diacritics. For example in one case in Madamira, the verb "rawaAhu" (رواه), meaning "narrated by" is changed into "ruwaAp" (رواة), meaning "narrators". Another example is the passive verb "yusotavonaY" (يسْتثنى), meaning "to be excluded" that is changed into the present tense verb "yasotavoniy" (يسْتثني), meaning "excludes". In both examples, the last letter is changed into a very similar, but different form. We see a similar behavior in Farasa, where in some examples a word containing two adjacent *Lam* (ل) letters (with Shadda on the second *Lam*), where the first *Lam* is a preposition. In this case, there is an additional Alif letter introduced between the two Lam letters. For example, the word (للّه) "lil∼ah" (l + Allah) is transformed into (لاله) "liAlhi" – i.e. (l + Alh).

In Table 8, we show the average number of diacritics per letter as well for the gold standard and all systems used in our experiments. It shows that Madamira and Farasa both assign about the same amount of diacritics on average, but substantially fewer than the gold standard. This means that both tools are especially punished by the strict evaluation. These findings motivate us to repeat the evaluation using the *relaxed mode*.

Corpus	Approach	All Diacritics		Ignore Last	
		DER	WER	DER-1	WER-1
Quran	Dict. Lookup	**7.3**	24.0	**3.2**	15.6
	Seq. Labeling	15.1	22.0	7.6	13.5
	Madamira	14.5	26.4	10.2	15.6
	Farasa	7.8	**14.0**	5.0	**6.8**
RDI	Dict. Lookup	10.1	27.9	**3.4**	16.7
	Seq. Labeling	16.7	28.0	12.0	13.6
	Madamira	12.5	20.4	8.6	10.2
	Farasa	**8.3**	**13.8**	5.0	**5.1**
Tashkeela	Dict. Lookup	10.1	28.1	**3.3**	16.7
	Seq. Labeling	24.0	35.4	15.0	22.0
	Madamira	12.4	20.3	8.5	10.1
	Farasa	**8.3**	**13.9**	5.0	**5.1**

Table 9: Error rates in relaxed evaluation mode

5.2 Relaxed Evaluation

Table 9 shows the results in relaxed mode, where we only take into account cases where all tools under consideration return a diacritic for a given letter. As expected, the error rates drop substantially, but not evenly for all approaches. In order to better show the improvement (decrease in error rates) obtained by switching from strict to relaxed evaluation mode, we report the relative change between both modes in Table 10. It can be clearly seen that this switching improves the tools performance in general. Sometimes, a tool is making a dramatical change, such as the dictionary lookup baseline under the DER and DER-1 metrics.

Looking again at the error rates in Table 9, relaxed evaluation mode reveals that Farasa is still performing better than Madamira in all cases, but for the DER and DER-1 metrics the dictionary lookup baseline is close or even better. The big difference between DER and WER performance for the dictionary lookup approach is most likely explained by errors in the inflectional diacritics that are impossible to resolve without looking at the context. However, that such a simple approach performs so well is surprising and shows that there is still a lot room for improvement in the area of automatic diacritization.

Corpus	Approach	All Diacritics		Ignore Last	
		DER	WER	DER-1	WER-1
Quran	Dict. Lookup	63	13	80	7
	Seq. Labeling	29	22	16	32
	Madamira	31	28	34	25
	Farasa	36	26	44	28
RDI	Dict. Lookup	62	12	83	26
	Seq. Labeling	33	24	22	39
	Madamira	30	28	34	28
	Farasa	21	12	25	33
Tashkeela	Dict. Lookup	62	13	83	26
	Seq. Labeling	4	4	4	1
	Madamira	31	29	35	29
	Farasa	22	13	26	34

Table 10: The relative change (in %) between the strict and relaxed evaluation modes

6 Qualitative Analysis

As most of the systems from the literature are not freely available, we have no way of directly comparing our results with those approaches unless they have the same settings. There is still a gap between our experimental results in relaxed mode and some of the reported published results in Table 5. Part of the gap can certainly be attributed to differences in the corpora. To see how the systems are performing, we also conducted a small diacritization experiment that only involves the best baseline (dictionary lookup), Madamira, and Farasa. We conduct a simple experiment using a blind MSA test set, a sample with 94 non-diacritized words (crawled from the internet). It was then diacritized using dictionary lookup (which was trained with RDI), Madamira (SAMA-enabled), and Farasa. We gave the resulting diacritized text to two Arabic teachers with appropriate experience to conduct the evaluation.

To look at the kinds of errors we were getting, the annotators were asked to identify the incorrectly diacritized words using word error rates (WERs) metrics because it is easy to manage for the volunteer teachers. Additionally, they were asked to state the reason if a diacritization produced by Madamira or Farasa was incorrect. For that purpose, we are using a error classification scheme developed for Arabic learner corpora (Abuhakema et al., 2008).

Error Category	Error Subcategoy	Annotator 1		Annotator 2	
		Madamira	Farasa	Madamira	Farasa
Form/Spelling	Shadda	2	3	2	3
	Tanween	6	6	6	6
Morphology	Partial-Inflection	1	1	0	1
	Full-Inflection	2	0	2	0
Grammar	Active-Passive Voice	2	2	2	2
Diacritization	Missing Short Vowel	6	0	5	0
	Confused Short Vowel	1	5	1	4
Overall		20	17	18	16

Table 11: The annotated WERs subcategories.

Form/Spelling Errors caused by Shadda (consonant doubling), or Tanween (nunation).

Morphology Correct lexical item, but wrong case ending, e.g. Kasra instead of Fatha.

Grammar Errors caused by changes in grammatical role, e.g. active or passive voice (المبني للمعلوم و المجهول).

Diacritization Errors caused by incorrect, missing or redundant short vowels (i.e. lexical diacritics).

Table 11 shows the distribution of error categories as reported by the annotators. The inter-evaluator agreement for the annotated WER (using Cohen's kappa) is almost perfect with values of .93 and .96 for Madamira and Farasa respectively. The majority of the mistakes are due to form/spelling and diacritization errors. In the form/spelling category, both tools make a lot of Tanween errors. This is to be expected, as it has been reported that the diacritization tools work relatively well on lexical diacritics, but that they are much less effective for case-ending diacritics (Habash et al., 2007). In the 'Diacritization' category, we observe a quite different behavior. Madamira has more missing vowels, i.e. it seems to rather not return a diacritic than to get it wrong. Farasa is on the opposite side of the trade-off with no missing short vowels, but almost as many confused short vowels.

7 Conclusion

The performance numbers reported in the literature on automatic diacritization are inconclusive, as the experimental settings are not comparable in most cases. In this

paper we establish a framework to compare the state-of-the-art publicly available Arabic diacritizers. The test data was drawn from the Quran, Tashkeela, and RDI corpora. Under controlled settings, we compared two strong baselines and two well-known systems: Madamira and Farasa. The error rates are reported in strict and relaxed evaluation modes to ensure fair comparison. We find that Farasa is outperforming Madamira in both evaluation modes, but that in relaxed mode the simple dictionary lookup baseline is surprisingly strong. In general, our error rates are much higher than the ones reported in the literature and we currently have no satisfying explanation for the difference. We are making our evaluation framework publicly available in order to foster additional research in this area and to allow for more approaches to be tested under reproducible conditions.

References

Abandah, G., Graves, A., Al-Shagoor, B., Arabiyat, A., Jamour, F., and Al-Taee, M. (2015). Automatic diacritization of arabic text using recurrent neural networks. *International Journal on Document Analysis and Recognition (IJDAR)*, 18(2):183–197.

Abuhakema, G., Faraj, R., Feldman, A., and Fitzpatrick, E. (2008). Annotating an arabic learner corpus for error. In *LREC*.

Acs, J. and Halmi, J. (2016). Hunaccent: Small footprint diacritic restoration for social media. In *Normalisation and Analysis of Social Media Texts (NormSoMe) Workshop Programme*, page 1.

Ahmed, A. and Elaraby, M. (2000). *A large-scale computational processor of the arabic morphology, and applications*. PhD thesis, Faculty of Engineering, Cairo University Giza, Egypt.

Alghamdi, M., Muzaffar, Z., and Alhakami, H. (2010). Automatic restoration of arabic diacritics: a simple, purely statistical approach. *Arabian Journal for Science and Engineering*, 35(2):125.

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

Azmi, A. and Almajed, R. (2015). A survey of automatic arabic diacritization techniques. *Natural Language Engineering*, 21(03):477–495.

Bebah, M., Amine, C., Azzeddine, M., and Abdelhak, L. (2014). Hybrid approaches for automatic vowelization of arabic texts. *arXiv preprint arXiv:1410.2646*.

Belinkov, Y. and Glass, J. (2015). Arabic diacritization with recurrent neural networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2281–2285.

Bouamor, H., Zaghouani, W., Diab, M., Obeid, O., Oflazer, K., Ghoneim, M., and Hawwari, A. (2015). A pilot study on arabic multi-genre corpus diacritization annotation. In *ANLP Workshop 2015*, page 80.

Buckwalter, T. (2004). Buckwalter arabic morphological analyzer version 2.0. linguistic data consortium, university of pennsylvania, 2002. ldc cat alog no.: Ldc2004l02. Technical report, ISBN 1-58563-324-0.

Čibej, J., Fišer, D., and Erjavec, T. (2016). Normalisation, tokenisation and sentence segmentation of slovene tweets. In *Normalisation and Analysis of Social Media Texts (NormSoMe) Workshop Programme*, page 5.

Darwish, K. and Mubarak, H. (2016). Farasa: A new fast and accurate arabic word segmenter. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Paris, France. European Language Resources Association (ELRA).

Darwish, K., Mubarak, H., and Abdelali, A. (2017). Arabic diacritization: Stats, rules, and hacks. In *Proceedings of the Third Arabic Natural Language Processing Workshop*, pages 9–17.

Daxenberger, J., Ferschke, O., Gurevych, I., Zesch, T., et al. (2014). DKPro TC: A Java-based Framework for Supervised Learning Experiments on Textual Data. In *ACL (System Demonstrations)*, pages 61–66.

Diab, M., Ghoneim, M., and Habash, N. (2007a). Arabic diacritization in the context of statistical machine translation. In *Proceedings of MT-Summit*.

Diab, M., Hacioglu, K., and Jurafsky, D. (2007b). Automated methods for processing arabic text: From tokenization to base phrase chunking. *Arabic Computational Morphology: Knowledge-based and Empirical Methods. Kluwer/Springer*.

Dukes, K. and Habash, N. (2010). Morphological annotation of quranic arabic. In *LREC*.

El-Sadany, T. and Hashish, M. (1989). An arabic morphological system. *IBM Systems Journal*, 28(4):600–612.

Elshafei, M., Al-Muhtaseb, H., and Alghamdi, M. (2006). Statistical methods for automatic diacritization of arabic text. In *The Saudi 18th National Computer Conference. Riyadh*, volume 18, pages 301–306.

Fan, R.-E., Chang, K.-W., Hsieh, C.-J., Wang, X.-R., and Lin, C.-J. (2008). Liblinear: A library for large linear classification. *Journal of machine learning research*, 9(Aug):1871–1874.

Farghaly, A. and Shaalan, K. (2009). Arabic natural language processing: Challenges and solutions. *ACM Transactions on Asian Language Information Processing (TALIP)*, 8(4):14.

Graff, D., Maamouri, M., Bouziri, B., Krouna, S., Kulick, S., and Buckwalter, T. (2009). Standard arabic morphological analyzer (sama) version 3.1. *Linguistic Data Consortium LDC2009E73*.

Habash, N. (2010). Introduction to arabic natural language processing. *Synthesis Lectures on Human Language Technologies*, 3(1):1–187.

Habash, N., Gabbard, R., Rambow, O., Kulick, S., and Marcus, M. P. (2007). Determining case in arabic: Learning complex linguistic behavior requires complex linguistic features. In *EMNLP-CoNLL*, pages 1084–1092.

Habash, N. and Rambow, O. (2005). Arabic tokenization, part-of-speech tagging and morphological disambiguation in one fell swoop. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 573–580. Association for Computational Linguistics.

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

Habash, N., Rambow, O., and Roth, R. (2009). MADA + TOKAN: A toolkit for Arabic tokenization, diacritization, morphological disambiguation, POS tagging, stemming and lemmatization. In *Proceedings of the 2nd international conference on Arabic language resources and tools (MEDAR), Cairo, Egypt*, pages 102–109.

Hifny, Y. (2012). Smoothing techniques for arabic diacritics restoration. In *12th Conference on Language Engineering*, pages 6–12.

Lafferty, J., McCallum, A., and Pereira, F. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data.

Maamouri, M., Bies, A., Buckwalter, T., and Mekki, W. (2004). The penn arabic treebank: Building a large-scale annotated arabic corpus. In *NEMLAR conference on Arabic language resources and tools*, volume 27, pages 466–467.

Maamouri, M., Bies, A., and Kulick, S. (2006). Diacritization: A challenge to arabic treebank annotation and parsing. In *Proceedings of the Conference of the Machine Translation SIG of the British Computer Society*. Citeseer.

Maamouri, M., Bies, A., and Kulick, S. (2009). Creating a methodology for large-scale correction of treebank annotation: The case of the arabic treebank. In *Proceedings of MEDAR International Conference on Arabic Language Resources and Tools, Cairo, Egypt*.

Metwally, A. S., Rashwan, M. A., and Atiya, A. F. (2016). A multi-layered approach for arabic text diacritization. In *Cloud Computing and Big Data Analysis (ICCCBDA), 2016 IEEE International Conference on*, pages 389–393. IEEE.

Nelken, R. and Shieber, S. (2005). Arabic diacritization using weighted finite-state transducers. In *Proceedings of the ACL Workshop on Computational Approaches to Semitic Languages*, pages 79–86. Association for Computational Linguistics.

Pasha, A., Al-Badrashiny, M., Diab, M., El Kholy, A., Eskander, R., Habash, N., Pooleery, M., Rambow, O., and Roth, R. (2014). Madamira: A fast, comprehensive tool for morphological analysis and disambiguation of arabic. In *LREC*, pages 1094–1101.

Rashwan, M., Al-Badrashiny, M., Attia, M., Abdou, S., and Rafea, A. (2011). A stochastic arabic diacritizer based on a hybrid of factorized and unfactorized textual features. *Audio, Speech, and Language Processing, IEEE Transactions on*, 19(1):166–175.

Rashwan, M. A., Al Sallab, A. A., Raafat, H. M., and Rafea, A. (2015). Deep learning framework with confused sub-set resolution architecture for automatic arabic diacritization. *IEEE Transactions on Audio, Speech, and Language Processing*, 23(3):505–516.

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

Schlippe, T., Nguyen, T., and Vogel, S. (2008). Diacritization as a machine translation problem and as a sequence labeling problem. In *8th AMTA conference, Hawaii*, pages 21–25.

Shaalan, K., Abo Bakr, H., and Ziedan, I. (2009). A hybrid approach for building arabic diacritizer. In *Proceedings of the EACL 2009 workshop on computational approaches to semitic languages*, pages 27–35. Association for Computational Linguistics.

Zerrouki, T. and Balla, A. (2017). Tashkeela: Novel corpus of Arabic vocalized texts, data for auto-diacritization systems. *Data in Brief*, 11:147–151.

Zitouni, I., Sorensen, J., and Sarikaya, R. (2006). Maximum entropy based restoration of arabic diacritics. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 577–584. Association for Computational Linguistics.