Proceedings Paper:
Alosaimy, A and Atwell, E orcid.org/0000-0001-9395-3764 (2018) Diacritization of a Highly Cited Text: A Classical Arabic Book as a Case. In: Proceedings of ASAR'2018 Arabic Script Analysis and Recognition. ASAR'2018 Arabic Script Analysis and Recognition, 12-14 Mar 2018, Alan Turing Institute, The British Library, London UK. IEEE , pp. 72-77.

Diacritization of a Highly Cited Text: A Classical Arabic Book as a Case

Abdulrahman Alosaimy
School of Computing
University of Leeds
Leeds, UK
scama@leeds.ac.uk

Eric Atwell
School of Computing
University of Leeds
Leeds, UK
e.s.atwell@leeds.ac.uk

***Abstract*— We present a robust and accurate diacritization method of highly cited texts by automatically "borrowing" diacritization from similar contexts. This method of diacritization has been tested on diacritizing one book: "Riyad As-Salheen", for the purpose of morphological annotation of the Sunnah Arabic Corpus. The original source of Riyad is about 48.66% diacritized, and after borrowing diacritization, the percentage jumps to 76.41% with low diacritic error rate (0.004), compared to 61.73% (DER=0.214) using MADAMIRA toolkit, and 67.68% (DER=0.006) using Farasa toolkit. More importantly, this method has reduced the word ambiguity from 4.83 diacritized form/word to 1.91.**

Keywords—diacritization; Arabic; NLP; Sunnah; Riyad As-Salheen

I. INTRODUCTION

In the Arabic language, a high amount of phonological information is missing such as *short vowels*, *Shaddah*, *tanween*, *Maddah*, and sometimes *hamzah*[1] as well. They (collectively called diacritics) are not usually written. As a result, the ambiguity at the word level is high in Arabic. There is an average of 11.5 diacritizations/word according to [1]. For example, a vowelized form of the word فهم (fhm) can be one of the following "non-comprehensive" list:

1. فَهِمَ (fahama) (v.) to understand
2. فَهَّمَ (fahhama) (v.) to teach
3. فَهُمْ (fa+humo) (conj. + pron.) and they
4. فَهَمَّ (fahamma) (conj. + v.) and (he) intend

Arabic diacritization is the computational process of recovering missing diacritics to the orthographic word. This process is known for improving readability (e.g. children books and educational textbooks), automatic speech recognition (ASR) [2], text to speech (TTS) [3], information retrieval (IR), and morphological annotation [4].

Words can be *fully* diacritized: diacritics for all letter are specified or *partially:* diacritics for part of the letters are specified. Texts are usually fully diacritized for children's educational purposes, or when the great precision of pronunciation is required e.g. the Quran. [5]. On the other hand, the text is mostly partly or unwritten, due to three reasons: to speed up the reading speed [5], not to strain the eyes and to speed up the typing by one third (required for typing diacritics).

A special type is the *minimal:* where some diacritics are specified in which these specifications are enough to avoid word's ambiguity. But ambiguity here is ambiguous, and the minimal level depends on the audience (e.g. reader's level of education) and target; for morphological annotation in Natural Language Processing (NLP), a minimal diacritization is the minimal partial diacritization that is sufficient to eliminate other possible diacritizations produced by a lexicon or morphological analyser.

[1] In cases where Hamza is considered a diacritic, only different shapes of Hamza on Alif is considered.

Diacritization is usually done fully, but this full diacritization is not necessary diacritizing each letter, due to the missing standard definition of the *fully-diacritized word.* There are some letters that are not diacritized even in lexicons, and by convention are *no-vowel* letters (i.e. has an intuitive vowel but not written). For example, using some diacritization standards, the letter that precedes a long vowel and the lam letter in definite AL article are two no-vowel letters. However, deciding whether *Waw*/*Yaa* letters are consonant or a vowel is ambiguous. Similarly, deciding whether the lam is part of a definite AL article is ambiguous too.

Arabic diacritization has grabbed the attention of Arabic NLP researchers, and much work has been done. Previous approaches have focused on improving the quality of automatic diacritization to produce a fully diacritized version of the text, either using *rule-based* approach [6], *statistical* approaches using, for example, recurrent networks [7], n-gram model [8], or *hybrid* approaches which usually perform the best [9]–[11]. This work, however, focuses on diacritizing text for the purpose of manual annotation later. That is, the diacritization approach seeks a high accuracy in diacritization but is not necessary to diacritize the full text. This approach crosses some interests with [4] which exploits diacritizing to improve morphological annotation. Our methodology is unique as it exploits partial diacritized texts as a source for diacritization. We borrow partial diacritizations from similar contexts and merge them together, and hope it lowers the ambiguity level of that word as much as possible.

II. Motivation

This article is motivated by our project of developing semi-automatically annotated Sunnah Arabic Corpus. Since its text is not been fully diacritized, we needed to adopt a method for diacritizing. Since the corpus mostly consists of texts that are highly quoted in other diacritized texts, we had the idea of "borrowing" their diacritization.

In many Classical Arabic texts, it is common to diacritize the word at least minimally: to the amount that is enough to remove the ambiguity to the readers. However, this borderline is not clear enough, and words that seem clear to the writer might still be ambiguous to a reader. Therefore, we notice different diacritization of the same word in different positions within the book, or between different versions of the book. These differences are exploited for the sake of improving morphological annotation of our Sunnah Arabic Corpus (SAC) and reach the minimal diacritization for each word.

III. Data

We picked one book from our Sunnah Arabic Corpus: Riyāḍu Aṣṣāliḥīn[2] (aka The Meadows of the Righteous) which is a compilation of 1896 hadith narratives written by Al-Nawawi and published in 1334. The total number of words in Riyad is around ~144k (~17k word types), and 48.66% of its letters are diacritized. Riyad was chosen due to several reasons:

1. It compiles narrations reported in other Hadith books (e.g. Albukhari) which make them a good source for diacritization.
2. Its codex was validated and investigated by several scholars by a scientific palaeographical process; at least there are two digitally available validated versions of the same text.
3. Its narratives have been explained in 6 written books.

The currently available diacritized corpora are either annotated corpora (mainly news) and Tashkeela (religious texts) [12], a corpus of 6.15 million words, which we used as an initial source for diacritization. But since Tashkeela focuses on fully diacritized texts, we added several Hadith books downloaded from Shamela library. Shamela (http://shamela.ws) is a downloadable library that contains at least 5300 Arabic books in Islamic studies and becomes the standard library of Arabic classical books. It has been used to obtain Arabic classical text in building several corpora [12]–[14].

Although having a large collection should not lower the accuracy of our method, we limit the corpus size in our experiments for training time efficiency. We picked relevant books from Tashkeela and

[2] All experiments in this paper and their used data is available at: http://github.com/aosaimy/sac

Shamela, i.e. books that have a high likelihood of quoting texts from Riyad. This selection method is done manually. This selection can be done automatically as we developed a small companion tool that measures one book's contribution by computing the number of matching n-grams with additional diacritization. The final corpus is 7677814 words, where 58.31% of its letters are diacritized.

In Arabic examples for the rest of this article, we use Buckwalter transliteration[3] instead, as it is easier to examine the differences of the diacritics. Diacritics are small glyphs, and the differences might not easily visually noticeable. Please note that these diacritics does not represent the possible diacritization states of one letter as they can be combined (especially with Dhammah), and some letters only accept a subset of them. The maximum total number of states is 18.

F	Fatha Tanween	N	Dhammah Tanween
K	Kasrah Tanween	a	Fatha
u	Dhammah	i	Kasrah
~	Shaddah	o	Sokun
|	Maddah		

IV. Methodology

Since the text in Riyad is highly cited and quoted, we have increased its text diacritization level, by automatically "borrowing" diacritization from other books. We developed an open-source diacritizer[4] that matches undiacritized version of one word in Riyad with its equivalent in other books using their word n-gram concordance. Algorithm 1 describe formally the method which could be explained in more details as follows:

1. It converts target text into a list of word n-grams, with reference to its locations in text, diacritized and undiacritized versions of the centre word.
2. It reads documents in source corpora in parallel. For each n-gram that is on our list (after normalization), it builds a list of matching word-ngrams.
3. For matching n-grams, it extracts variant diacritizations of the centre word and counts the number of occurrences of that diacritization.
4. Once finished, variants are sorted by the number of occurrences to prevent infrequent diacritization from bubbling up to the surface diacritization in the next step.
5. Centre words variants are merged recursively: The merge procedure (Algorithm 2) is done letter by letter, and for every letter, only candidate diacritics that do not contradict with one existing are merged.
6. (*extended* version) uses morphological analyser (MA) to improve the results if

Algorithm 1. BorrowBasedDiacritize

DEFINE:

$W = \{w_1, w_2, ..\}$ is a series of words w.

$l(w)$ is a series of letters l_i of word w.

$v(w) = \{v_1, v_2, ..\}$ where v_i is a series of diacritics of letter l_i and $|v(w)| = |l(w)|$.

$nWGram(w_i, n) = \{w_{i-n}, w_{i-n+1}, \dots, w_i, \dots, w_{i+n-1}, w_{i+n}\}$

$MA(w)$ is a series of $v(w)$ from a morphological analyser.

INPUT: W_{train}, W_{target}, n

OUTPUT: $v`(w)$ for all $w \in W_{target}$ such that $|v_i| \leq |v`_i|$ for all i.

1. $G^{train} = nWGram(x, n)$ for $x \in W_{train}$
2. $M(w_i^{target}) \subset G^{train}$ where $g_k^{train} = nWGram(w_i^{target}, n)$
3. $D_i = \{\dots v(x) \dots\}, \forall\ nWGram(x, n) \in M(w_i)$
4. $D_i = sort(D_i)$
5. while $i < |D_i|$; do
   $v(w_i) = merge(v(w_i), v(d_i))$
   od
6. $v(w_i) = MA_0(w_i)\ iff\ |MA(w_i)| = 1$

end;

Algorithm 2. Merge

INPUT: $v(w_1), v(w_2)$ where $l(w_1) = l(w_2)$

OUTPUT: $v`(w_1)$ such that $\sum|v_i| \leq \sum|v`_i|$.

$v_i(w_1) := v_i(w_2)$ iff $v_i(w_1) \leq v_i(w_2)$

end;

[3] http://www.qamus.org/transliteration.htm

[4] Available freely at http://github.com/aosaimy/arabic-vowelizer

possible. Merged centre words are replaced by a more thorough diacritization (if exist) by consulting a morphological analyser if and only if it matches one candidate diacritization.

7. Centre word's locations in the text are replaced with the new diacritized version.

This methodology assumes the following:

1. The diacritization of the source corpora is done manually, i.e. not artificially,
2. Diacritization of both target and source is standard,
3. Word diacritization is only based on window of n,
4. Target text is quoted or reused in source corpora, and
5. There is no other diacritized form if morphological analyser says so (only applicable in extended version)

As stated before, our ultimate goal is to fully diacritize words in the SAC to increase the robustness of the morphological annotation of the corpus. In the next subsections, we show how these assumptions are valid for our case.

A. *Non-Artificial Diacritics in Source Corpora*

For the first assumption, we used Shamela, where we could not find a sign of automatic diacritization. Moreover, some diacritized corpora like [12] used some of its books.

B. *Diacritics Standardization*

To enforce the same standard in source and target, we perform diacritization normalization as illustrated in Table III. We use the notion of regular expressions, which is quite efficient for text substitutions. For example, Fatha Tanween should always be before Alif and Alif Maqsorah.

C. *Word diacritization is the same for n surrounding words*

Changing one final diacritic from a full sentence might change its meaning completely [20]. While this clearly contradicts with our assumption, we examine the quantity of these cases in the full corpus.

To validate prior assumptions (mainly the last), we extracted word quint-grams that has variant diacritization of its centre word. Then, we examine the top of the list (top 100), ranked based on the number of variants in descending order. Table IV lists a sample of top n-grams of first experiments for n=5.

All variants did not show a sign of artificial diacritic, nor show a non-standard diacritization. The centre word has no conflicting diacritization for 98% of the top 100 of the list. Conflicting diacritization is due to different pronunciation of proper nouns, misspelt diacritics, or improper last diacritic.

D. *Similarity between source and target corpora*

SAC is mostly a collection of religious text which is widely quoted. Its content has been explained by several authors. This increases the chance that its text has been quoted. The results of our experiment show that at least 84.34% of the corpus word n-grams has been found in the source corpora.

E. *There is no other diacritized form if morphological analyser says so*

We used four morphological analysers to increase the diacritization coverage for our corpus. By merging the output of analysing each word, we built a list of possible diacritization of each word. After close examination of the results, their level of diacritization is different. The diacritized format is not usually full. Table II showed the diacritization coverage for each analyser. While merging analysers' results increases the coverage, similar words do not merge as their level of diacritization is different; which results in having more than one form of diacritization when in fact there should be one. This explains the jump in the number of possible diacritization from 10.38 (at maximum) to 17.42.

Using SAWAREF toolkit [15], we run four morphological analysers, namely Elixir Functional Morphology (EX) [16], ALMORGEANA (included in MADA toolkit) (AL) [17], AraMorph (BP) [18], and AlKhalil (KH) [19], on the lexicon of Riyadh Asslaheen (17600 distinct words). The average number of possible diacritized forms is shown in Table I.

TABLE I. POSSIBLE DIACRITIZATION STATISTICS PER MORPHOLOGICAL ANALYSER.

MA	Max	Mean	Median	Coverage
EX	124	8.46	6	67.46%
KH	96	10.38	7	80.64%
BP	20	2.38	2	47.67%
AL	23	3.69	3	42.65%

We only use MA diacritization if it matches only one form. Using a random sample (of 100 words) that matches this criterion, we could not spot a single error in the enhanced diacritization. This suggests that it is safe to assume there is no other diacritized form if morphological analyser says so.

V. EVALUATION

Our evaluation uses two metrics for accuracy, and coverage, both in terms of character level. Accuracy is measured by Diacritic Error Rate (DER), i.e. a letter is marked correctly if it has all diacritics in the original text. Coverage is measured by the percentage of letters that has at least one diacritic.

In addition, we introduce ambiguity measure defined as the practical average of the possible number of diacritizations per word. In theory, if a word of three letters has no diacritics, there are at least eight possible diacritization for each letter (final letter can have more). But we report the practical number of diacritizations only, extracted from a lexicon (or in our case morphological analysers). In case a partially diacritized word, the morphological analyser will only return the subset of possible diacritizations that has match given diacritization.

$$DER = \frac{\sum f(w_i)}{\sum |l(w_i)|} \quad (1)$$

$$f(w) = |s|, s \subset v(w) \; that \; is \; incorrect$$

$$Coverage = \frac{\sum v_i, v_i \in v(w_i) \; and \; |v_i|>0}{\sum |l(w_i)|} \quad (2)$$

$$Ambiguity = \frac{\sum ambig(w_i)}{|W|} \quad (3)$$

$$ambig(w) = |MA(v(w))|$$

We test on the part of the text that is already diacritized. In other words, we used our models to diacritize a completely undiacritized version of Riyad, and later test the accuracy and coverage of our assumption on the diacritized version. However, since this method does not diacritize the full text, we only evaluate the letters with a diacritic.

In Table II, we compare the accuracy (in terms of DER), coverage, and the word ambiguity after diacritization of six models of diacritization. We can see that accuracy improves when word's context is larger, but on the other hand, the coverage drops. Word ambiguity does not change after using MA, as MA's diacritization is not used unless word diacritization only matches one candidate. The accuracy increased very slightly (about 0.0001) when using MA; however, the coverage increased by ~0.2.

TABLE II. EVALUATION OF NGRAM MODELS.

Model	Coverage	DER	Ambiguity
Undiacritized	0	N/A	17.42
Baseline	48.66%	N/A	4.83
3-gram	80.32%	0.007	1.56
3-gram+MA	81.26%	0.007	1.56
5-gram	76.41%	0.004	1.91
5-gram+MA	77.70%	0.004	1.91
7-gram	73.97%	0.003	2.13
7-gram+MA	75.59%	0.003	2.13

Additionally, we compare our results to two major available diacritizers: MADAMIRA [10] and FARASA [9]. Diacritization is normalized for both toolkits. Our 5-gram model slightly surpasses both tools, and FARASA scored an error rate is 0.006 while MADAMIRA was not performing well: 0.214, which is due to the fact that MADAMIRA removes original diacritics before processing the text. For a fair comparison, we re-compute the error rate given undiacritized version; FARASA error rate jumped to 0.263, and the DER of our 5-gram model increased slightly to 0.008.

TABLE III. COMPARISON WITH MAJOR OFF-THE-SHELF DIACRITIZERS.

Tool	Coverage	DER	Input Text
MADAMIRA	N/A	N/A	Diacritized
	61.73%	0.214	Undiacritied
FARASA	67.68%	0.006	Diacritized
	65.36%	0.263	Undiacritied
5-gram	76.41%	0.004	Diacritized
	71.81%	0.008	Undiacritied

While the two tools are expected to diacritize the text thoroughly, we found that MADAMIRA only diacritized 61.73% of letters, and FARASA only diacritized 65.36%, 67.68% for undiacritized, and diacritized input text respectively. Using our method, the 5-gram model diacritized 71.81% of letters. This is due to diacritization standards of final letter, article AL and long vowels in addition to the fact that our measure does not tolerate letters with obvious diacritics (such as Alif Madd (آ), Alif (إ) and Lower Hamza (إ)). Even the Quran text (extracted from

Tanzil Project), which is known to have a full diacritized form, covers only 77.83% of letters. Table III summarizes these findings.

VI. Conclusion

We presented and evaluated a methodology for diacritizing highly quoted texts by borrowing diacritization from its citations. This method exploits and reuses manual diacritization from other works. To fully exploit diacritized text from different origins, we had to deal with diverse diacritization standardization. This method is unique in reusing partially diacritized text as a source for diacritization.

By matching the undiacritized version of one word in target text with its equivalent standardized version in other books using their word n-gram concordance, the percentage of diacritized words in Riyad As-Salheen rose with high accuracy. We compared different models of our method intrinsically, and extrinsically with available external diacritizers.

We urge linguists and researchers to develop standards way of diacritization. We plan to extend this work to build a fully diacritized corpus of highly quoted texts. We plan to incorporate this method with our morphological annotation tool (Wasim)[21], as a helper for diacritization annotation layer. Additionally, merging diacritized forms from morphological analysers could help to build a more robust ensemble tagger. Finally, we plan to test the possibility of applying same method to recover missing Hamzah.

A. References

[1] F. Debili and H. Achour, "Voyellation automatique de l'arabe," in *Proceedings of the Workshop on Computational Approaches to Semitic Languages*, 1998, pp. 42–49.

[2] D. Vergyri and K. Kirchhoff, "Automatic Diacritization of Arabic for Acoustic Modeling in Speech Recognition," in *Proceedings of the Workshop on Computational Approaches to Arabic Script-based Languages*, 2004, pp. 66–73.

[3] C. Ungurean, D. Burileanu, V. Popescu, C. Negrescu, and A. Dervis, "Automatic diacritic restoration for a TTS-based e-mail reader application," *UPB Sci. Bull. Ser. C*, vol. 70, no. 4, pp. 3–12, 2008.

[4] N. Habash, A. Shahrour, and M. Al-Khalil, "Exploiting Arabic Diacritization for High Quality Automatic Annotation," in *Proceedings of Language Resources and Evaluation Conference*, 2016, pp. 4298–4304.

[5] E. Hermena, D. Drieghe, S. Hellmuth, and S. P. Liversedge, "Processing of Arabic Diacritical Marks: Phonological-Syntactic Disambiguation of Homographic Verbs and Visual Crowding Effects," *J. Exp. Psychol. Hum. Percept. Perform.*, pp. 494–507, 2015.

[6] Y. A. El-Imam, "Phonetization of Arabic: rules and algorithms," *Comput. Speech Lang.*, vol. 18, no. 4, pp. 339–373, 2004.

[7] G. A. Abandah, A. Graves, B. Al-Shagoor, A. Arabiyat, F. Jamour, and M. Al-Taee, "Automatic diacritization of Arabic text using recurrent neural networks," *Int. J. Doc. Anal. Recognit.*, vol. 18, no. 2, pp. 183–197, 2015.

[8] Y. Hifny, "Higher Order n-gram Language Models for Arabic Diacritics Restoration," in *The Twelfth Conference on Language Engineering (ESOLEC'12)*, 2012, pp. 1–5.

[9] K. Darwish, H. Mubarak, and A. Abdelali, "Arabic Diacritization: Stats, Rules, and Hacks," in *Proceedings of The Third Arabic Natural Language Processing Workshop*, 2017, pp. 9–17.

[10] A. Pasha, M. Al-Badrashiny, M. Diab, A. El Kholy, R. Eskander, N. Habash, M. Pooleery, O. Rambow, and R. M. Roth, "Madamira: A fast, comprehensive tool for morphological analysis and disambiguation of arabic," in *Proceedings of the Language Resources and Evaluation Conference (LREC), Reykjavik, Iceland*, 2014.

[11] M. Rashwan, M. Al-Badrashiny, M. Attia, and S. M. Abdou, "A Hybrid System for Automatic Arabic Diacritization," *2nd Int. Conf. Arab. Lang. Resour. Tools*, no. June 2014, pp. 54–60, 2009.

[12] T. Zerrouki and A. Balla, "Tashkeela: Novel corpus of Arabic vocalized texts, data for auto-diacritization systems," *Data Br.*, vol. 11, pp. 147–151, 2017.

[13] M. Alrabiah, A. Al-Salman, E. S. Atwell, and N. Alhelewh, "KSUCCA: a key to exploring Arabic historical linguistics," *Int. J. Comput. Linguist.*, vol. 5, no. 2, pp. 27–36, Jun. 2014.

[14] Y. Belinkov, A. Magidow, M. Romanov, A. Shmidman, and M. Koppel, "Shamela: A Large-Scale Historical Arabic Corpus," *arXiv Prepr. arXiv1612.08989*, 2016.

[15] A. Alosaimy and E. Atwell, "Ensemble Morphosyntactic Analyser for Classical Arabic," in *2nd International Conference on Arabic Computational Linguistics*, 2016.

[16] O. Smrz, "Functional Arabic Morphology. Formal System and Implementation," ufal.mff.cuni.cz, 2007.

[17] N. Habash, O. Rambow, and R. Roth, "MADA+TOKAN: A Toolkit for Arabic Tokenization, Diacritization, Morphological Disambiguation, POS Tagging, Stemming and Lemmatization," in *Proceedings of the Second International Conference on Arabic Language Resources and Tools*, 2009, pp. 102–109.

[18] T. Buckwalter, "Arabic Morphological Analyzer (AraMorph)." 2002.

[19] M. Boudchiche, A. Mazroui, M. O. A. O. Bebah, A. Lakhouaja, and A. Boudlal, "AlKhalil Morpho Sys 2: A robust Arabic morpho-syntactic analyzer," *J. King Saud Univ. Inf. Sci.*, 2016.

[20] A. M. Azmi and R. S. Almajed, "A survey of automatic Arabic diacritization techniques," *Nat. Lang. Eng.*, vol. 21, no. 3, pp. 477–495, 2015.

[21] A. Alosaimy and E. Atwell, "Web-based Annotation Tool for Inflectional Language Resources Major features," in *11th Edition of its Language Resources and Evaluation Conference*, 2018.