Full Diacritization of the Arabic Text to Improve Screen Readers for the Visually Impaired

Batool Abuali and Mohamad-Bassam Kurdy

Master in Web Sciences, Syrian Virtual University, Damascus, Syria

Correspondence should be addressed to Batool Abuali; batoulabuali@gmail.com

Received 9 November 2021; Revised 14 January 2022; Accepted 17 May 2022; Published 16 July 2022

Academic Editor: Alessandra Agostini

Copyright © 2022 Batool Abuali and Mohamad-Bassam Kurdy. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

This paper aims to find the relationship between the full diacritization of the Arabic text and the quality of the speech synthesized in screen readers and presents a new methodology to develop screen readers for the visually impaired, focusing on preprocessing and diacritization of the text before converting it to audio. First, the actual need for our proposal was measured by conducting a MOS (Mean Opinion Score) questionnaire to evaluate the quality of the speech synthesized before and after full diacritization in the NVDA (https://www.nvda-ar.org/) screen reader. Then, an e-reader was built by integrating two models: the first one is for automatic Arabic diacritization (depending on Shakkala), and the second is a TTS model (depending on Tacotron). The quality of our proposed system was measured in terms of (1) pronunciation and (2) intelligibility, in which our system outperformed the commercial screen readers, NVDA and IBSAR (https://www.sakhr.com), as it recorded 60.67%, 17.67%, and 21.67% as correct, incorrect, and partially correct, respectively, for the isolated word test, and 84% correct results for the homograph test, and 78.50% and 93% correct results, respectively, for the DRT and DMRT tests.

1. Introduction

Text-to-Speech (TTS) systems are used to convert plain text included in digital documents into an audible format. It is a very essential technology in many fields and is used in multiple applications, especially those that help the visually impaired.

Even though Arabic is one of the most widely used languages worldwide, Arabic TTS systems are still in their infancy in comparison to other languages like English.

When we talk about Arabic natural language processing, the absence of diacritics poses a major dilemma for a group of software, especially speech synthesizers and screen readers, which help the visually impaired to overcome the barrier of using technology.

Arabic diacritics are divided into basic diacritics which play a grammatical role and give different meanings to words, for example, ( عَلَم) could mean flag or knowledge, and case endings or syntactic diacritics which determine words’ place in the context so it would be understood, for example, school ( مُدرسة), or an object ( مُدرسة), depending on the diacritization of the last letter.

Full diacritization improves the process of automatic reading of texts written in Arabic in screen reader applications. The basic Arabic alphabet contains 28 letters and 8 diacritics encoded in the standard hexadecimal notation under 0600–06FF as shown in Figure 1 [1].

TTS systems consist of two modules: (1) the NLP module and (2) the Digital Signal Processor module. In this research, we focus on the first part of a TTS system, suggest a change in the way that the input text is usually processed, and propose the possibility to improve screen readers used by the visually impaired by adding a full diacritization stage in the process of building an Arabic TTS system.

We measured the actual need for our proposal by conducting a 5-scale MOS questionnaire [2] (see Appendix), to evaluate the quality of speech synthesized before and after full diacritization in the NVDA screen reader in 7 categories. We selected our dataset to be balanced in the sense that it
contains a paragraph which is easy to pronounce, another paragraph which is slightly hard to pronounce, and a third one which is hard to pronounce. 152 native Arabic speakers aged 18–50 years participated in the survey.

The results shown in Figure 2 reflected a remarkable improvement, especially in the pronunciation part, where we recorded an increase of about 1 out of 5 overall degrees. This encouraged us to continue our work and be one of the first studies that focused on the importance and necessity of full diacritization of Arabic text in TTS systems, hoping to make the synthesized speech smoother and closer to the natural sound.

2. Literature Reviews

2.1. Automatic Arabic Diacritization. Regarding the Arabic language, the past approaches used in automatic diacritization are roughly classified as (1) rule-based, (2) statistical, and (3) hybrid [3]. We reviewed the approaches in the recently published literature on the diacritization problem as follows:

2.1.1. Rule-Based Approaches. These approaches used Arabic’s well-formed rules through methods like morphological analysis, syntactic analysis, and morph-phonological modules [3–5].

2.1.2. Statistical Approaches. The used methods include using HMM (Hidden Markov Models) [3, 6], N-grams models [3, 7, 8], Dynamic Programming methods [8, 9], classical Machine Learning models like MaxEnt (Maximum Entropy) classifier [10], and DL (Deep Learning) methods [11–13].

2.1.3. Hybrid Approaches. These approaches mix rule-based methods and statistical ones, and this includes a combination of linguistic knowledge (well-formed rules and dictionary retrievals with morphological analysis) and other techniques, like N-grams, HMM, DL models, and Machine Learning methods [3, 11, 14–17].

However, the accessible tools for Arabic text diacritization are still limited as most of the systems from the literature are not available for free use.

2.1.4. Comparison. A previous study compared the three past approaches from different points of view, and the results showed that hybrid approaches record higher accuracy in comparison with the other two [3]. Another review of the existing diacritization systems was provided and showed that Shakkala, which is a DL approach, is quite better than the traditional rule-based approaches and other existing systems and tools [11].

2.2. TTS. We reviewed the existing systems and methods for the Arabic speech synthesizing problem as follows:

2.2.1. Screen Readers. In the current time, there are multiple commercially available Arabic TTS systems, like Sakhr TTS (2012), Acapela (2017), Natural Soft (2017), CSTR Festival (2017), and MBROLA (2017). But the visually impaired cannot use these systems because they are not available for free. NVDA (2012) is a free software [18], which makes it one of the most widely used screen readers by the visually impaired in the Arab world.

A TTS system for diacritic Arabic texts was designed based on unit selection using a bigrams model [19]. An Arabic TTS support was developed to be included and integrated into the eSpeak system [20].

![Figure 1: Basic Arabic (Unicode block): 28 letters and 8 diacritics.](image1)

![Figure 2: MOS questionnaire results comparing the quality of the output in NVDA before and after full diacritization.](image2)
2.2.2. Comparison of Speech Synthesis Methods. A review of the different speech synthesis methods, including HMM, RBM (Restrictive Boltzmann machine), DBN (Deep belief network), DMDN (Deep mixture density network), DBLSTM (Deep bidirectional long short term memory), WaveNet, Tacotron, and CNN (Convolutional neural network) was presented, discussing the advantages and disadvantages of each method, as shown in Figure 3, and drawing the conclusion that DL-based models achieve a higher quality of the synthesized speech than the traditional methods [21].

Also, Arabic speech synthesis using deep learning architectures was explored in another study where the main two models utilized in an end-to-end Arabic TTS system were compared to a concatenative TTS system as shown in Figure 4 [22].

2.2.3. Evaluation of Screen Readers. TTS systems can be measured against multiple criteria like pronunciation, intelligibility, and comprehensibility. We reviewed the recent studies on the evaluation of Arabic TTS systems. In their study [23], the authors evaluate six Arabic TTS systems using four intelligibility tests: (1) Diagnostic Rhyme, (2) Modified Rhyme, (3) Phonetically Confusable Sentences, and the fourth one is related to the prediction of the diacritics of the input text and was proposed by the authors themselves. Another two tests were performed: (5) Arabic Text with All Sounds and (6) Best/Worst Pleasant Voice, which were proposed by the authors to determine the voice pleasantness. The authors also conducted an objective evaluation using two types of measures: (1) signal-to-noise variation and (2) linear predictive measure. In [18], the authors evaluate two of the most popular screen readers used in the Arab community using two pronunciation tests: (1) the isolated word test and (2) homographs, and two intelligibility tests: (1) DRT and (2) DMRT.

In building our proposed system in this study, we used Shakkala as it achieved the best results in full diacritization in our opinion and it is available as an open source, and we used Tacotron, as it is one of the best options available for
speech synthesis, and here we would like to make a point, that Google TTS is not accessible to us as Syrians so we could not use it in the proposed system or the evaluation. For evaluation, we used the same methods in [18], as our focus is on improving the pronunciation and differentiation of words so that the output is clearer and more natural for the Arabic listener.

It is important to emphasize that the purpose of our study is not to compete with other Arabic TTS systems, but to point out the importance of the full diacritization, and suggest the possibility of improving existing e-readers or building new ones that handle the initial text more efficiently.

3. Methodology

TTS systems consist of two modules: (1) the NLP module and (2) the Digital Signal Processor module. As our study focuses on the impact of full diacritization, as shown in Figure 5, our proposed system consists of two main units:

(i) Automatic diacritizer unit.
(ii) Text-to-speech unit.

3.1. Work Stages

First: add input; an input text is presented to the proposed system by either writing it directly in the GUI textbox shown in Figure 6, or by adding a whole file via the button named Full Process.

Second: clean input; the input text is processed and previous diacritics (if any) are removed.

Third: full diacritization; the processed text is fully diacritized (basic and case-ending diacritics).

Fourth: TTS; the fully diacritized text is converted to audio.

Fifth: save and play; a wave file is built and played, as shown in Figure 7. Check the file titled “output.wav” in the Supplementary Material and listen to the speech synthesized from the same input example.

3.2. More Details on the Proposed Model.

To build our proposed model, we used two pretrained models, the first one is Shakkala model [24], and the second is an open-

<table>
<thead>
<tr>
<th>Package</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bleach</td>
<td>1.5.0</td>
</tr>
<tr>
<td>Click</td>
<td>7.1.2</td>
</tr>
<tr>
<td>enum34</td>
<td>1.1.10</td>
</tr>
<tr>
<td>h5py</td>
<td>2.10.0</td>
</tr>
<tr>
<td>Keras</td>
<td>2.1.3</td>
</tr>
<tr>
<td>Markdown</td>
<td>3.3.3</td>
</tr>
<tr>
<td>Nltk</td>
<td>3.5</td>
</tr>
<tr>
<td>Numpy</td>
<td>1.15</td>
</tr>
<tr>
<td>Protobuf</td>
<td>3.14</td>
</tr>
<tr>
<td>PyYAML</td>
<td>5.3.1</td>
</tr>
<tr>
<td>SciPy</td>
<td>1.5.4</td>
</tr>
<tr>
<td>TensorFlow</td>
<td>1.4.0</td>
</tr>
<tr>
<td>Six</td>
<td>1.15</td>
</tr>
<tr>
<td>Tensorflow-tensorboard</td>
<td>0.4.0</td>
</tr>
<tr>
<td>Werkzeug</td>
<td>1.0.1</td>
</tr>
<tr>
<td>Falcon</td>
<td>1.2.0</td>
</tr>
<tr>
<td>Librosa</td>
<td>0.5.1</td>
</tr>
<tr>
<td>Matplotlib</td>
<td>2.0.2</td>
</tr>
<tr>
<td>Tqdm</td>
<td>4.11.2</td>
</tr>
<tr>
<td>Unidecode</td>
<td>0.4.20</td>
</tr>
<tr>
<td>Arabic-pronounce</td>
<td>0.2.5</td>
</tr>
</tbody>
</table>

Table 1: Libraries used.
A virtual environment was created using Visual Studio Code (https://code.visualstudio.com/) and the Python language to integrate both models. Several packages were tested for stability on those that make the two models work without errors, as in Table 1.

Also, some modifications were made to improve our proposed system, like adding a function to delete the pre-existing diacritics in the first unit of the system and editing the way that Tacotron pronounces specific parts of the Arabic words by modifying the Arabic-pronounce package.

It is important to emphasize that we did not write a single Python code from scratch, but most of the Python codes used in this study are from previously mentioned studies, which have been cited.

### 4. Evaluation

Based on the evaluation methods used in a previous comparative study [18], we compared the output of our proposed system with the previous results of both IBSAR and NVDA programs in two tests: (1) pronunciation and (2) intelligibility.

#### 4.1. Pronunciation Test

Two different pronunciation tests were conducted. The first one is called the “isolated word test,” where we pronounce each word without being in any surrounding context, and the second is the homograph test, where we present each homograph in a single sentence context.

In Arabic, homographs are words composed of the exact same letters but have different pronunciations and meanings, e.g. كَابَطُ (Kataba, means wrote) and كَابِتُ (Kotiba, means written).

For the isolated word test, we selected a list consisting of 30 words from a database of Arabic phonemes [26].

We selected the dataset to contain 10 words that are easy to pronounce, 10 words that are slightly hard to pronounce, and 10 words that are hard to pronounce. The words (written in Arabic and their transliteration in English) are shown in Table 2.

Similarly, we selected a list of 10 homographs and embedded each one of them in a one-sentence context. The homographs (written in Arabic and their transliteration in English) are shown in Table 3.

<table>
<thead>
<tr>
<th>No.</th>
<th>Word</th>
<th>No.</th>
<th>Word</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(bakhs)</td>
<td>1</td>
<td>(Asara)</td>
</tr>
<tr>
<td>2</td>
<td>(aghnaa)</td>
<td>2</td>
<td>(Usira)</td>
</tr>
<tr>
<td>3</td>
<td>(andhar)</td>
<td>3</td>
<td>(Sallama)</td>
</tr>
<tr>
<td>4</td>
<td>(badr)</td>
<td>4</td>
<td>(Sollima)</td>
</tr>
<tr>
<td>5</td>
<td>(badhk)</td>
<td>5</td>
<td>(Jamaa)</td>
</tr>
<tr>
<td>6</td>
<td>(fwz)</td>
<td>6</td>
<td>(Jammaj)</td>
</tr>
<tr>
<td>7</td>
<td>(hadhwah)</td>
<td>7</td>
<td>(Rabbaa)</td>
</tr>
<tr>
<td>8</td>
<td>(hadid)</td>
<td>8</td>
<td>(Robbiaa)</td>
</tr>
<tr>
<td>9</td>
<td>(kafat)</td>
<td>9</td>
<td>(Qatar)</td>
</tr>
<tr>
<td>10</td>
<td>(makhudh)</td>
<td>10</td>
<td>(Qurrr)</td>
</tr>
<tr>
<td>11</td>
<td>(addad)</td>
<td>11</td>
<td>(Kataba)</td>
</tr>
<tr>
<td>12</td>
<td>(ashkhas)</td>
<td>12</td>
<td>(Kotiba)</td>
</tr>
<tr>
<td>13</td>
<td>(iithm)</td>
<td>13</td>
<td>(Zahab)</td>
</tr>
<tr>
<td>14</td>
<td>(bikhashish)</td>
<td>14</td>
<td>(Zahaba)</td>
</tr>
<tr>
<td>15</td>
<td>(ejz)</td>
<td>15</td>
<td>(Aabra)</td>
</tr>
<tr>
<td>16</td>
<td>(fahm)</td>
<td>16</td>
<td>(Ebra)</td>
</tr>
<tr>
<td>17</td>
<td>(eatshan)</td>
<td>17</td>
<td>(Wagada)</td>
</tr>
<tr>
<td>18</td>
<td>(ladagh)</td>
<td>18</td>
<td>(Wogeda)</td>
</tr>
<tr>
<td>19</td>
<td>(lahazt)</td>
<td>19</td>
<td>(Aeen)</td>
</tr>
<tr>
<td>20</td>
<td>(maduba)</td>
<td>20</td>
<td>(Oean)</td>
</tr>
<tr>
<td>21</td>
<td>(maeswl)</td>
<td>21</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>(madah)</td>
<td>22</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>(marym)</td>
<td>23</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>(mishkat)</td>
<td>24</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>(waqf)</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>(qutna)</td>
<td>26</td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>(rasi)</td>
<td>27</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>(sabuun)</td>
<td>28</td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>(yastafi)</td>
<td>29</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>(raquhm)</td>
<td>30</td>
<td></td>
</tr>
</tbody>
</table>
4. Intelligibility Test. We determine intelligibility by whether the human user can understand the output of the TTS system or not. To achieve that, we used the most common tests: DRT (Diagnostic Rhyme Test) and DMRT (Diagnostic Medial Consonant Test).

Ten native Arabic speakers who are university students with vision impairment participated in the experiments, and were provided with instructions as follows:

Each time you hear a word, you will read two words written in braille. Please choose the word you heard.

In both DRT and DMRT tests, we selected 20 rhyming word pairs which differ in their initial consonant, e.g. جِلِّ (written in braille) and لَنْ (silent) in DRT, and in the intervocalic consonant, e.g. تَمْرَ (written in Arabic and their transliteration in English) are shown in Table 4.

5. Results and Discussion

5.1 Punctuation Test. Answers were listed as correct, incorrect, or partially correct for each word. The results revealed that for the first test (isolated word test): 60.67% of the answers were correct, 17.67% were incorrect, and 21.67% were partially correct, as shown in Figure 8, approximating the performance of NVDA and outperforming the performance of the IBSAR program, as stated in the previous study which was mentioned above.

Similarly, for the homograph test, the results revealed that 84% of the answers were correct, and 16% were incorrect, as shown in Figure 9, approximating the performance of NVDA and outperforming the performance of IBSAR.

5.2 Intelligibility Test. Answers were listed as correct or incorrect for each test. The results of DRT revealed that 79%
of our system answers were correct, and 21% were incorrect, as shown in Figure 10, outperforming the performance of NVDA and approximating IBSAR results, as stated in the previous study that was mentioned above.

Similarly, for the DMRT test, the answers were also listed as correct or incorrect for each test. The results revealed that 93% of our system’s answers were correct, and 7% were incorrect, as shown in Figure 11, clearly outperforming both of the other programs.

6. Conclusion

In this work, a new approach has been proposed to improve Arabic screen readers by focusing on the full diacritization of the text. Our proposed model was built by integrating two models: the first one is for automatic Arabic diacritization (depending on Shakkala), and the second is a TTS model (depending on Tacotron).

We evaluated our proposed system in terms of (1) pronunciation and (2) intelligibility, and compared it to the commercial screen readers: NVDA and IBSAR.

The results showed that the overall quality of our proposed system is better than the other two.

In the future, we will work to improve our proposed system by:

(i) Training the system (both models) on a larger set of data.
(ii) Adding rules related to reading known numbers, dates, and names.
(iii) Working on the prosody factor that affects the quality, naturalness, and intelligibility of synthesized speech.
(iv) Building an interactive website that can contribute to evaluating the system.

Appendix

MOS questionnaire

Q1: Global impression

Please rate the sound quality of the voice you heard.

Bad
Poor
Fair
Good
Excellent

Q2: Listening effort

Please rate the degree of effort you had to make to understand the message.

Message not understood with any feasible effort
Major effort required
Effort required
Slight effort required
No Effort Required

Q3: Comprehension problems

Were single words hard to understand?

Every word
Many
Some
Few
None

Q4: Speech sound articulation

Were the speech sounds clearly distinguishable?

Not at all
Not very clear
Fairly clear
Clearly enough
Q5: Pronunciation
Did you notice any anomalies in the naturalness of sentence pronunciation?
Yes, very
Yes, annoying
Yes, slightly annoying
Yes, but not annoying
No

Q6: Speaking rate
Was the speed of delivery of the message appropriate?
No, too fast
No, too slow
Yes, but faster than preferred
Yes, but slower than preferred
Yes

Q7: Voice Pleasantness
Was the voice you heard pleasant to listen to?
Very Unpleasant
Unpleasant
Fair
Pleasant
Very Pleasant

Data Availability
Most Python codes used in this study are from previously reported studies, which have been cited. Any further data or content used in the experiments are available from author Batool Abuali upon request via: batoulabuali@gmail.com and batool_110906@svuonline.org.

Conflicts of Interest
The authors declare that they have no conflicts of interest.

Authors’ Contributions
Batool Abuali designed the research plan, reviewed the previous studies, wrote the literature review, collected the data, organized and ran the experiments, presented, analyzed, and interpreted the results, and prepared and wrote the draft manuscript. Prof. Mohamad-Bassam Kurdy supervised the study and aided in revising the manuscript.

Acknowledgments
The authors extend their appreciation to the visually impaired students and the management team at Light Initiative for helping run the experiments.

Supplementary Materials
Audio file titled “output.wav” is an example of the speech synthesized by our proposed system. (Supplementary Materials)

References
Application of Natural Language to Information Systems, Salford, UK, June 2013.


[25] Y. Sharief, *End to End Arabic TTS System Based on Tacotron*.