Arabic and Multilingual Scripts Sorting and Analysis

Khalil Shihab
Department of Computer Science, Sultan Qaboos University, P.O.Box: 36, Al-Khod, Muscat 123, Sultanate of Oman

Abstract: - Arabic and multilingual scripts have two major problems: the encoding and the sorting of these scripts. The current character encoding systems including the Unicode standard is no different than any other alphabetic script. These systems are conceived for encoding raw text, not as glyph lists, i.e. ignoring the script graphical structure. Also, comparing strings can be an intricate operation that involves subtle issues, and determining the sorting order cannot be done by a simple table look-up or a comparison of character-code values. Furthermore, sorting rules vary not just among script systems but also among the individual languages. Therefore, this paper presents these problems along with proper treatments.

Keywords: Arabic Script, Multilingual Software, Sorting Algorithms

1. Introduction
In terms of encoding, Arabic is no different than any other alphabetic script, but care has to be taken to leave its graphical structure intact [2]. The Unicode standard is conceived for encoding raw text, not as a glyph list. The purpose of Unicode is to enable cultural diversity without imposing irrelevant constraints. What is required in Unicode is a variant character-glyph model to handle these issues on the level of the Operating System. But it is not just that. For the end-user it would be a major improvement if Operating Systems in general facilitated the use of the optimal script system for a given script. This means the ability to switch between different Font Rendering and Layout mechanisms. An example of such modularity is Apple's Open Font Architecture (OFA).

On the other issue, sorting is also the subject of a great deal of study since it is a common operation that is required for everyday text-manipulation tasks which can consume a lot of computer time. Current text utilities provide routines for comparing two strings for sorting purposes [1]. These routines have been used widely and they are based on the code page and not on the font page. However, these routines suffer from many drawbacks. For example, a few Arabic characters that have the same weight that should be treated identically in sorting unless they are the only difference between two words. Also, these routines do not give users the option to ignore diacritics. Furthermore, these routines do not efficiently handle Arabic documents mixed with other languages.

2. Background
In 1963, the American National Standards Institute (ANSI) announced the American Standard Code for Information (ASCII), which is based on 7-bit code. The ASCII character set defines a mapping of the letters, numerals, and specified punctuation and control characters to the numbers from zero to 127 [2]. However, this system left many positions, such as some of the control characters, unallocated. These control characters were later used to extend ASCII code into 8-bit codes. A similar modification to the Arabic character set, ASMO-449, had to be carried out to become the recognized standard ASMO-708, which is identical to ISO-8859/6 [3]. See Table 1.

In the real world there are many cases in which it is necessary to process multiple languages in a single document [4]. The most noteworthy of the systems that deal with these cases were Xerox Corporation's Star Workstation, which had a multilingual word processor called Viewpoint, and IBM Corporation's 5550 office computers. Both of these systems could process multiple Asian languages, in addition to multiple languages that use the Latin script, but they were never came into wide use because of their high cost, their inefficiency and for not sorting options [6].

The Unicode Standard specifies an algorithm for the presentation of text with bidirectional behaviour, for example, Arabic and English. Characters are stored in logical order. The Unicode Standard includes characters to specify changes in direction when scripts of different directionality are mixed. For all scripts Unicode text is in logical order within
the memory representation, corresponding to the order in which text is typed on the keyboard [7].

For assigning character codes, a single 16-bit number is assigned to each code element defined by the Unicode Standard. Each of these 16-bit numbers is called a code point and, when referred to in text, is listed in hexadecimal. For example, the code point U+0636 is the hexadecimal number 0636. It represents the character (ض) in the Unicode Standard [5]. Each character is also assigned a unique name that specifies it and no other. For example, U+0636 assigned the character name "ARABIC LETTER DHAD". These Unicode names are identical to the ISO/IEC 10646 names for the same characters.

3. Sorting Problems

3.1. Arabic Language Features

There are a number of differences and similarities between Arabic text and other texts [7]. These include the following main points:

- The characters are laid out in RTL (Right-TO-Left) order, with the exception of numbers, which are laid out LTR (Left-To-Right) as in English. Text is right aligned on the page, and written from top to bottom (like English).

- Letters change shape depending on context. Each letter has up to four forms: the initial form, here it is the first letter in a word, the final form, where it is the last letter in a word, the isolated form, and the medial form. Table 2 shows the forms of some the Arabic letters.

<table>
<thead>
<tr>
<th>Initial</th>
<th>Isolated</th>
<th>Medial</th>
<th>Final</th>
</tr>
</thead>
<tbody>
<tr>
<td>عرب</td>
<td>ع</td>
<td>مع</td>
<td>عرب</td>
</tr>
<tr>
<td>ق ا</td>
<td>فيم</td>
<td>ككتب</td>
<td>نص خيط</td>
</tr>
<tr>
<td>مالك</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 2: Contextual Analysis

- Arabic includes diacritics. For example, in sorting, the Arabic word حداد may come before حداد but it should be before because حداد include the diacritical marks and shadda.

- Arabic includes ligatures. This is the process whereby two letters printed together are replaced by a single new character. English uses the same process in typesetting (for example, ff, fi, fl, ffl, fi). Examples of Arabic are as follows:

<table>
<thead>
<tr>
<th>Hindi Digits</th>
<th>Arabic Digits</th>
</tr>
</thead>
<tbody>
<tr>
<td>٢٤٧٦٥٤٣٢١</td>
<td>٢٤٧٦٥٤٣٢١</td>
</tr>
</tbody>
</table>

Table 3: The Arabic and Hindi digits

- The physical appearance of a bilingual string can differ enormously from its internal representation. Non-contiguous cursor movement and selection and semantic re-ordering of sections of text must be taken into account by BiDi-enabled applications.

- Some characters are neglected in writing. For example, the word الزحم is wrongly sorted before the word الزحم. This is because the actual writing of the word الزحم.

- Distinctive connections.

Arabic writing early on manifested two types of letter connections: horizontal (right to left) and vertical (top to bottom), see Figures 1 and 2.

Fig. 1. Horizontal connection

Fig. 2. Vertical connection
Vowels began to be written in the seventh century. In modern Arabic script this method is still in use. However, the dots have been superseded by miniature versions of letters such as Alef (small stripe) and Waw (small comma). See Figures 3, 4, and 5.

Arabic Base Glyphs
The archigraphemic structure implies that Arabic graphemes share many structural elements [10]. This phenomenon was exploited here to the maximum. In the examples below ("BEH-class” graphemes), the skeletons – archigraphemes – are identical, see Figure 7; only the attachments – distinctive features – are different, see Figure 8.

3.2. Collation
Sorting, or collation, is something required by almost all software packages. This can show up in surprising places too. For example, the UNIX ls and DOS dir commands, which list files in a directory, both require a sophisticated sorting algorithm for them to work correctly.

Correct sorting is a rather complicated business. In most cases, sorting is performed to satisfy a user requirement of having a sorted list that is efficiently searchable, taking advantage of a conventional "alphabetical" order. To be of any use, the sort must meet the user's expectation.

As previously mentioned that Arabic and many other languages actually view two-character pairs as a single character (\(\check{\text{r}}\) = \(\text{ض} + \text{غ}\)). Although the Arabic language has no upper and no lower case characters, other languages do have [8]. Upper case letters almost always sort before lower case letters due to ASCII character ordering (the simple rule used in most sorting software is that if the ASCII representation of a letter is a smaller number than another, the letter is bibliographically lower, or earlier, in the alphabet.
### Table 6. Arabic script with primary and secondary sorting

<table>
<thead>
<tr>
<th>Primary sorting order</th>
<th>Secondary sorting order</th>
</tr>
</thead>
<tbody>
<tr>
<td>ببٍّ</td>
<td>ججٍّ</td>
</tr>
<tr>
<td>ببٌ</td>
<td>ججٌ</td>
</tr>
<tr>
<td>ببً</td>
<td>ججً</td>
</tr>
</tbody>
</table>

For example, here is a list of strings that have not been sorted yet:

بَيْتُ بَيْتَ حَمْدُ حَلَقُ دَمَّمُ رَاضِيُ سَانَ كَتَب

After primary sorting, the list appears as follows:

بَيْتُ بَيْتَ حَمْدُ حَلَقُ دَمَّمُ رَاضِيُ سَانَ كَتَب

After secondary sorting, the list appears as follows:

بَيْتُ بَيْتَ حَمْدُ حَلَقُ دَمَّمُ رَاضِيُ سَانَ كَتَب

### Table 7. Primary and secondary sorting

<table>
<thead>
<tr>
<th>Sorted Strings</th>
</tr>
</thead>
<tbody>
<tr>
<td>ءُبِّتْ</td>
</tr>
<tr>
<td>بَيْتُ</td>
</tr>
<tr>
<td>حَمْدُ</td>
</tr>
<tr>
<td>رَاضِيُ</td>
</tr>
<tr>
<td>سَانَ</td>
</tr>
<tr>
<td>كَتَبَ</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

Table 7. Primary and secondary sorting
Language sorting routines, see table 5, should also consist of two steps: primary sorting order and secondary sorting order. In the primary sorting order for many Arabic script languages diaritical marks and kashidas are ignored. In the secondary sorting order, diaritical marks and cultural issues should be considered. Table 6 shows some Arabic script with primary and secondary sorting.

In the primary sorting, each of the characters in the first row would be considered equivalent and sorted before characters in the second row. In secondary sorting, the order of the characters in each row would be taken into consideration. Table 7 illustrates the primary and the secondary sorting.

4. Soring Algorithms: Performance Comparison
Sorting Arabic and other languages using the available sorting algorithms either on UNIX system or on Microsoft Windows is based on the code pages and not the font pages. Diacritics and Kashidas are ignored in sorts, and these applications do not give users any option to consider any extension.

4.1. Windows NT and DOS Systems
The Windows sort program uses the collating-sequence table corresponding to the country code and code-page settings. Characters greater than ASCII code 127 are sorted based on information in the COUNTRY.SYS file or in an alternate file specified by the COUNTRY command in the CONFIG.NT file.

In addition to the Windows sort program, we used Perl for Windows NT because Perl is a highly recommended programming script for manipulating strings. Therefore, the following routine is used.

```perl
local(@keys);
foreach(@original_array){
push(@keys, key_expression);
}
sub sort_function {key_comparison_expression;}
@sorted_array=@original_array{sort sort_function
$[… $#original_array];

Where @keys hold each element's key value; @original_array is the array to be sorted; key_expression is the expression applied to $_ to compute a key value; sort_function is a subroutine that signifies sorting parameters; and key_comparison_expression and @sorted_array are the new arrays with the sorted elements. However, we gain no significant difference in the sorting order and speed.

4.2. Large Arabic and English Scripts
We prepared two text files each contains 15,000 lines of text. One contains only Arabic script and the other contains only English script. We tested the five widely used sorting methods: shell, selection, , bubble, merge and quick sort. From Figures 9 and 10, which are given below, we noticed that the performance of Shell Sort is much better than the rest.

![Graph](image_url)

Fig. 9. Shell, Merge and Quick Sort chart for Arabic Data
5. Conclusions
A variant character-glyph model is required in the Unicode encoding system to handle the features of the Arabic script on the level of the Operating Systems. But it is not just that. For the end-user it would be a major improvement if Operating Systems in general facilitated the use of the optimal script system for a given script.

Also, many problems can be faced in the process of developing multilingual sorting software that includes a bi-directional language such as Arabic. This paper reports these problems and provides reliable and efficient solutions. Also, we carried out an empirical study to compare the performance of five well known and recommended sorting algorithms. We have tested these algorithms on texts from English, Arabic and Urdu languages. As a result of this comparison study, we selected the shell and quick sort as the underlying techniques for our multilingual sorting software. This is because, these techniques performed far better than the others, especially for dealing with multilingual and bi-directional scripts such as the Arabic script.

6. References


