Justification of Arabic Text

By Raniah Alghamdi
Outline

• Introduction
• Calligraphic rules of justification in Arabic
• Dynamic font
• Letter stretching:
  o Horizontal stretching
  o Horizontal and vertical stretching
• Letter substitution
• Conclusion
Introduction

• Justification in English typography is mainly based on: hyphenation and spacing.

• Justification of Arabic text is challenging due to [1]:
  o Hyphenation is not used.
  o The strict calligraphic rules of Arabic.
  o The characteristics of the language:
    
    1. Cursivity
      • It is mandatory in Arabic.
    
    2. Ligatures
      • Mandatory or aesthetic. 

    3. Allographs
      • There are different shapes for each letter based on its position in the word: isolated ٤, initial ٤, median ٤ or final ٤.
Introduction

• The curvilinear connection between Arabic letters is called “Kashida” or “Tamdid”.

• Kashida can be used for:
  o Letter stretching/intra-letter stretching
  o Connection stretching/ inter-letter stretching

• Some Arabic letters are composed of [2]:
  o Static part: unstretchable.
  o Dynamic part: stretchable.

• Not all letters are stretchable.

• Diacritic point is a measurement unit marked by the head of the nib.
Calligraphic rules of justification in Arabic

• Types of stretching: preferable, allowed, not recommended and prohibited [1].

• Type of stretching is determined by:
  o length of the word.
  o position of the word.
  o position of the letter to be stretched.

• The degree of extensibility is also determined by the nature of the letter and the position of letter.
Calligraphic rules of justification in Arabic

- Summarization of some stretching rules [1].

<table>
<thead>
<tr>
<th>Word length</th>
<th>Stretching type</th>
<th>Stretching position</th>
<th>Exceptions</th>
<th>Location in line</th>
</tr>
</thead>
<tbody>
<tr>
<td>Two</td>
<td>Strictly prohibited</td>
<td>--</td>
<td>• Words ends with (ي or ى) cannot be stretched.</td>
<td>• Only one word in a line could be stretched.</td>
</tr>
<tr>
<td>Three</td>
<td>Generally: allowed but not recommended.</td>
<td>Second letter</td>
<td>• Letter س should be followed by two metric points kashida.</td>
<td>• Stretching is preferable at the end, not recommended in the beginning, and prohibited in the middle.</td>
</tr>
<tr>
<td>Four</td>
<td>Allowed and preferable.</td>
<td>Second letter</td>
<td>• Stretching for words end with مـ is allowed and preferable, and kashida is placed before the last letter</td>
<td></td>
</tr>
<tr>
<td>More</td>
<td>It follows the rules of the extensibility table (next slide).</td>
<td>--</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Calligraphic rules of justification in Arabic

- Letter extensibility table based on the context [1].

| Precede | ạ | ِ | َ | ُ | ُ | ُ | ُ | ا | ل | م | ن | و | ُ | ج | د | ص | س | ر | ت | ع |
| 1 1 1 1 | +1 | -1 | -1 | -1 | -1 | -1 | -1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| -1 -1 -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 |
| -1 1 1 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 |

- +: preferable
- No sign: allowed
- -: not recommended
- Empty: prohibited
- [letter]: all letters with same shape.
- Number: length of stretching [number, 12]
Dynamic font

• Fortid system, for example, was using multiple fixed-sized baseline fillers (rectilinear segments) to stretch last connection in the word [4][7].
  o Letter stretching was not used.

• Methods of justification in Arabic:
  o Stretching using dynamic fonts: intra-letter or inter-letter.
  o Letter substitution.

• Font types [3]:
  o Static font:
    ▪ Characters are designed → digitized → printed.
    ▪ A font is defined as a whole which makes each letter has the same identical shape within the font.
  o Dynamic font:
    ▪ Characters are redefined each time they are printed.
    ▪ Each character could have a variation in its glyph shape.
Dynamic font

• Dynamic font could be divided into three categories based on the parameters exchanged [3]:
  1. Characters are self sufficient.
  2. Characters depend on limited parameters.
  3. Characters depend on large number of parameters.

• PostScript [3][4]:
  • It supports dynamic font.
  • It has a cache mechanism that may be disabled.
  • The bitmap of each character that is printed by using “show” command will be fully recomputed.
  • Postscript type 3 allows use of the full PostScript language.
Letter stretching

• Letter stretching requires dynamic font that accepts the amount of stretching as parameter.

• PostScript type 3 is usually used for stretching.

• In intra-letter stretching, the dynamic part of the letter is a horizontal curvilinear stroke.

• Generally, outline strokes are presented as one (or more) four-point Bézier curve(s).

• Methods of stretching:
  o Horizontal stretching: by Berry [4].
  o Horizontal and vertical stretching: by Bayar and Sami [2] and CurExt [5].
Horizontal stretching

• In Berry’s method [4], to calculate the stretching amount:
  o effortid system divides the total amount of white spaces in a line by the number of the stretching places (connections or letters).

• For stretching, add the amount of stretching to the x value of the control points on the right of the Bézier curve.

Image adapted from [4]
Horizontal stretching

• Stretching rules: stretching should maintain:
  o Corners, the shape of the letters and the y value of the outline.

Original stroke: the two Bézier curves and the shared end point have the same tangents.

Process: Add A to the x values of the three rightmost points of the right-hand curve. Problem: a new corner is initiated which violates one of the stretching rules.

Image adapted from [4]
Horizontal stretching

• **A solution:**
  o Divide the amount of stretching to two equal amount and deals with each adjacent curves individually (next slide).

**Solution**: increase the y value as well as the x value of the control points on the right.

**Problem**: it changes the y values of the right part which violates one of the stretching rules.

Image adapted from [4]
Horizontal stretching

PostScript code of normal Baa letter

```
/baa_CP
{ 502 0 -15 -180 527 153 setcachedevice
  0 0 moveto
  243 -94 rmoveto
  42 -32 rlineto
  -34 -38 rlineto
  -40 28 rlineto
  closepath
  0 0 moveto
  260 -5 rmoveto
  -17 -1 -136 -10 -192 15 rcurveto
  -58 25 -18 106 -8 127 rcurveto
  0 -6 -11 -57 11 -69 rcurveto
  67 -39 163 -24 190 -23 rcurveto
  28 2 129 10 156 29 rcurveto
  17 11 27 44 28 48 rcurveto
  1 -14 3 -27 28 -51 rcurveto
  20 -19 27 -15 29 -15 rcurveto
  0 -56 rlineto
  -1 0 -21 -6 -48 26 rcurveto
  -9 11 -16 17 -22 34 rcurveto
  0 -1 1 -16 -28 -34 rcurveto
  -46 -28 -86 -25 -144 -31 rcurveto
  closepath 0 0 moveto fill } def
```

PostScript code after stretching

```
/baa_CP
{ /fact fact Arabic-NaskhFont /FontMatrix get 0 get div def
  502 fact add 0 setcharwidth
  /fact_2 fact 2 div def
  0 0 moveto
  243 fact_2 add -94 rmoveto
  42 -32 rlineto
  -34 -38 rlineto
  -40 28 rlineto
  closepath
  0 0 moveto
  260 fact_2 add -5 rmoveto
  -17 -1 -136 fact_2 sub -10 -192 fact_2 sub 15 rcurveto
  -58 25 -18 106 -8 127 rcurveto
  0 -6 -11 -57 11 -69 rcurveto
  67 -39 163 fact_2 add -24 190 fact_2 add -23 rcurveto
  28 2 129 fact_2 add 10 156 fact_2 add 29 rcurveto
  17 11 27 44 28 48 rcurveto
  1 -14 3 -27 28 -51 rcurveto
  20 -19 27 -15 29 -15 rcurveto
  0 -56 rlineto
  -1 0 -21 -6 -48 26 rcurveto
  -9 11 -16 17 -22 34 rcurveto
  0 -1 1 -16 -28 -34 rcurveto
  -46 -28 -86 -25 -144 -31 rcurveto
  closepath 0 0 moveto fill } def
```
Horizontal stretching

• Stretching Connection:
  ◦ Adding fillers.
  ◦ Stretching the connection part.

Image adapted from [4]
Horizontal stretching

Stretching with –scf command:
Connections of last words absorb all of the stretch.

Stretching with –slf command:
The final stretchable letter of each line absorbs all the stretch.

كِأختبار ذاتي للبرنامج،
هذه المقالة طَبَعَت
بواسطة البرنامج
الموصوف أعلاه، وتحتوي
على عدة أمثال لنَصْص
كَتَبَت باللغات العَربيَّة،
العَبريَّة، والفارسية.

كِاختبار ذاتي للبرنامج،
هذه المقالة طَبَعَت
بواسطة البرنامج
الموصوف أعلاه، وتحتوي
على عدة أمثال لنَصْص
كَتَبَت باللغات العَربيَّة،
العَبريَّة، والفارسية.
Horizontal and vertical stretching

- Bayar and Sami’s method [2]:
  - They found that vertical stretching by 0 to half of a diacritic point is a characteristic of Arabic calligraphy.
  - The curves of characters are defined by the movement of the nib head.

Qaf letter without stretching of Berry’s font.

Qaf letter after stretching with Berry’s method; nib head was not considered; nib head exceeds the letter.

Qaf letter of Bayar and Sami’s method after horizontal and vertical stretching.

Nib head “Qalam” (rectangular shaped)

Width = l
Thickness = e = l/6

Images adapted from [1] and [2]
Horizontal and vertical stretching

• The movement of the nib is represented by the movement of one corner of the head which is defined as a Bézier curve.

• To design Raa (ڥ) letter:

**B1** a Bézier curve with four control points M10, M11, M12 and M13.

**B2** a Bézier curve with the four control points M20, M21, M22 and M23 is the translation of B1 by vector u.

Four different surfaces:
1. Sl1.
2. Sl2 (the translation of Sl1 by vector v).
4. Se2 (the translation of Se1 by vector u)

They define the surface razed by the nib head.
Horizontal and vertical stretching

• For stretching:
  o Bayar and Sami defined two types of curves: set B1 and set B2.
  o Kashida is located between two Bézier curves of type 1 and 2.

Image adapted from [2]
Horizontal and vertical stretching

- Two stretching functions $E_{be}$ and $E_{fa}$ have been defined to stretch curves in $B_1$ and $B_2$ respectively.

\[ E_{be}: B_1 \times [0, h_m] \times [0, v_m] \rightarrow B_1 \]
\[ (B, h, v) \rightarrow E_{be} (B, h, v) \]
\[ B_2 = E_{be} (B_1, h, v) \]

(h, v) $\in$ [0, $h_m$] $\times$ [0, $v_m$].

$h$: horizontal stretching amount

$B_2 = E_{fa} (B_1, h, v)$

$v$: vertical stretching amount

$E_{fa}: B_2 \times [0, h_m] \times [0, v_m] \rightarrow B_2$

(B, h, v) $\rightarrow$ $E_{fa} (B, h, v)$
Horizontal and vertical stretching

• The justification rule of Bayar and Sami’s method:
  • If the word ends with stretchable letter, then stretch it.
  • If not, stretch the last connection in the word.

Before justification

After justification

Image adapted from [2]
Horizontal and vertical stretching

• Stretching mathematical symbols:
  o Mathematical symbols could have variation of sizes:

  ![Symbols](image)

  ![CurExt](image)

  • CurExt [5]: It is considered the first application that produces variable-sized curved mathematical symbols through parameterized dynamic fonts.

Image adapted from [5]
Horizontal and vertical stretching

• CurExt is a package for Arabic mathematical expression that allows symbols stretching horizontally and vertically.

• Using CurExt with TeX:
  o The size of variable-sized curved symbols is calculated based on the length of the mathematical expression that is covered by the symbol.
  o METAFONT (font generator) is then used to create the bitmap font file for the calculated symbol size.

\[
\left(\begin{array}{ccc}
1 & 2 & 3 \\
4 & 5 & 6 \\
7 & 8 & 9 \\
0 & 1 & 2
\end{array}\right)
\]

Parenthesis in TeX

\[
\left\{\begin{array}{ccc}
1 & 2 & 3 \\
4 & 5 & 6 \\
7 & 8 & 9 \\
0 & 1 & 2
\end{array}\right\}
\]

Parenthesis in TeX under CurExt
Horizontal and vertical stretching

- The glyph parameters to determine the shape of the parenthesis symbol:
  - curvilinearity of the symbol.
  - the width (thickness) of the middle and the ends.
  - the shape of the ends.
  - the size of the covered expression.

\[
\begin{pmatrix}
1 & 2 & 3 \\
4 & 5 & 6 \\
7 & 8 & 9
\end{pmatrix}
\begin{pmatrix}
0 & 1 & 2 & 3 & 4 \\
5 & 6 & 7 & 8 & 9 \\
0 & 1 & 2 & 3 & 4
\end{pmatrix}
\begin{pmatrix}
0 & 1 & 2 & 3 & 4 \\
5 & 6 & 7 & 8 & 9 \\
0 & 1 & 2 & 3 & 4
\end{pmatrix}
\]
Horizontal and vertical stretching

- Sum symbol in RyDArab (an extension of TEX system):
  - Without CurExt: \( \sum \)
  - With CurExt: \( \texttt{csum} \)
  - Syntax: \( \texttt{\arabmath} \) 
    \[
    \{\texttt{csum}_{\{b=T-1}\}^{s}\}
    \]

Image adapted from [5]
Letter substitution

• Arabic letters may have various alternative shapes (glyphs).

• These alternatives can be used to optimize text justification.

• Arabic is rich of ligatures that could be used to increase/decrease word length.

• OpenType [6]:
  o It uses this language feature and provides “jalt” (justification alternative) feature.
  o In rendering process, OpenType can provide alternative glyph forms for letters based on their context.
  o A “jalt” table shows some glyph variants:
    ▪ ligatures, allographs, or stretched glyphs with kashida.
Letter substitution

• Optimum-fit algorithm [7]:
  o Main idea: find the optimum breaking point in the line to improve the overall justification of the paragraph.
  o Paragraph consists of a list of horizontal nodes: glyph, ligature, breakpoint... etc.

• Elyaakoubi and Lazrek extended the optimum-fit algorithm to cover Arabic text justification [7].
  o It scans through the nodes and for each glyph node the algorithm checks the jalt table for another alternative glyph.
  o It calculates and store the difference between the width of the original glyph and the alternative one.
  o It calculates the badness of the lines twice to determine the breakpoint feasibility.
  o **Note:** badness is the difference between the width of the current line and the maximum width of a line.
Letter substitution

- Azmi and Alsaiari designed an algorithm to use ligature substitution effectively [8].
- They increase the number of ligatures in Arabic Typesetting Font (ATF) font.
- Two levels of justification:
  1. substitution level
  2. kashida level

<table>
<thead>
<tr>
<th>Isolated</th>
<th>Initial</th>
<th>Middle</th>
<th>Final</th>
<th>Variations</th>
</tr>
</thead>
<tbody>
<tr>
<td>كا</td>
<td>كا</td>
<td>لا</td>
<td>لا</td>
<td></td>
</tr>
<tr>
<td>بم</td>
<td>بم</td>
<td>لا</td>
<td>لا</td>
<td></td>
</tr>
<tr>
<td>كم</td>
<td>كم</td>
<td>لا</td>
<td>لا</td>
<td></td>
</tr>
<tr>
<td>كل</td>
<td>كل</td>
<td>لا</td>
<td>لا</td>
<td></td>
</tr>
<tr>
<td>لا</td>
<td>لا</td>
<td>لا</td>
<td>لا</td>
<td></td>
</tr>
<tr>
<td>لم</td>
<td>لم</td>
<td>لم</td>
<td>لم</td>
<td></td>
</tr>
<tr>
<td>بح</td>
<td>بح</td>
<td>بح</td>
<td>بح</td>
<td></td>
</tr>
<tr>
<td>صح</td>
<td>صح</td>
<td>صح</td>
<td>صح</td>
<td></td>
</tr>
<tr>
<td>سم</td>
<td>سم</td>
<td>سم</td>
<td>سم</td>
<td></td>
</tr>
<tr>
<td>صم</td>
<td>صم</td>
<td>صم</td>
<td>صم</td>
<td></td>
</tr>
</tbody>
</table>
Letter substitution

1. Substitution level
   • Compute the badness of each line.
   • If badness > zero, then find alternative glyphs and re-compute the badness value until the badness becomes zero.

2. Kashida level
   • Check the priority levels of words for holding the kashida based on words’ length.
   • Stretch the word with highest priority.
   • Use the extensibility table and the badness value to decide the stretching amount.

Note:
• Priority levels of words for holding the kashida: 4 letters > 5 letters > 6 letters > more letters.
Letter substitution

Justification of standard MS Word

Incorrect stretching places

Justification using Azmi and Alsaiari’s algorithm

Image adapted from [8]
Conclusion

• Dynamic fonts have been used to stretch letters, letter connections and mathematical symbols in Arabic.

• Stretching in Arabic requires horizontal and vertical stretching.

• Justification rules of Arabic calligraphy are considered partially in most Arabic justification algorithms.

• Using letter substitution method for text justification helps to benefit from ligature feature to extend or reduce words’ length.
References


Thank you for listening,,,,