We describe \texttt{dadh}, a package allowing simple typesetting in Arabic script, intended for mixed Latin–Arabic script usage, in situations where heavy-duty solutions are discouraged. The \texttt{dadh} system operates with both Unicode and transliterated input, allowing the user to choose the most appropriate approach.

1 Introduction

As with many \LaTeX{} projects, this one was started to accommodate \LaTeX{} and with the Write\LaTeX{} platform [3], of which the author is an enthusiastic user; it should allow user-friendly and robust input of Arabic text, including when placed inside \LaTeX{} command arguments; it should typeset in an optimal way all combinations of letters and diacritics that may appear in scholarly text; it should provide some extra features: being able to easily change the letter form, as well as to colorize specific letters without breaking contextual analysis; the font should be easily readable in a context of mixed Latin-Arabic script. In the following discussion we will see why existing systems did not fulfill the requirements, and how the author solved the problem.

1.1 Existing systems and their pitfalls

As we live in the 21st century, an obvious choice for typesetting Arabic in \LaTeX{} is \texttt{XeLaTeX} [12], a \LaTeX{} avatar (in)famous for typesetting in non-Latin scripts by taking advantage of operating system resources.

\textbf{Requirement 1} \texttt{XeLaTeX} indeed is provided on the Write\LaTeX{} platform. But the \LaTeX{} overhead for typesetting Arabic in \texttt{XeLaTeX} is quite heavy (packages \texttt{fontspec, xunicode, arabxetex}, etc.) and hence, not surprisingly, \texttt{XeLaTeX} is incompatible with the IEEE style.

\textbf{Requirement 2} \texttt{XeLaTeX} is unable to use a transliteration system and hence requires the Arabic text to be input in Arabic script and \textit{only} in Arabic script. Using a transliteration to input Arabic may seem terribly old-fashioned to the reader, but there are cases where it is the best solution. One of these cases is the context of this paper: a mixture of Latin script, Arabic script, and \LaTeX{} commands.

Indeed, at the GUI level, there are — at least — two drawbacks in combining Arabic and Latin script in the same paragraph:

1. the use of the cursor and of left and right arrow keys is very cumbersome: when you select a location with the cursor you don’t know whether you are in right-to-left or left-to-right mode and hence you don’t know in which direction to advance, or how to select a given character string;

2. the situation is made even worse by the fact that some punctuation marks (period, exclamation mark, dashes, parentheses, braces, brackets, etc.) are common to the two scripts and hence the — quite sophisticated — bidirectional algorithm is used to determine whether a punctuation mark is to be placed on the left or on the right of an Arabic word. The bidirectional algorithm (or ‘bidi’ for the insiders) is both a blessing and a curse. It is a blessing because it puts some order in the rendering of mixed right-to-left and left-to-right texts (cf. [6, p. 133–146]) — but this works well only if \texttt{rle} and \texttt{lre} are used to indicate the embedding level. Otherwise, in everyday use, bidi is a curse. Fig. 1 shows how the \LaTeX{} code for writing the word \texttt{كـ} with the middle letter colorized in red is displayed by various programs under various operating systems; not a single one of them really makes sense.

\textbf{Requirement 4} Sophisticated OpenType fonts [6, \S\ D.9.4] handle relatively well the many letter + diacritic combinations, and most systems (including \texttt{XeLaTeX}) can colorize word parts without breaking contextual analysis through the use of the zero-width joiner character [6, p. 104]. But not a single system is able to colorize single letters inside \texttt{y}, since this ligature has always been considered as a single glyph by font designers.

\textbf{Requirement 5} The best way to match Latin and Arabic script is to choose an Arabic font with relatively small differences in height between letters. A quite common choice is the font \textit{Geezah} by Diwan Software Ltd (developed for Apple WorldScript in the early nineties, and still included in Mac OS X, through today). \textit{Geezah} is a nice font but its diacritics are placed rather suboptimally, and modifying their positions requires a high amount of competence in fiddling around with OpenType features.

A simple Arabic typesetting system for mixed Latin/Arabic documents: \textit{dadh}
1.2 The long and winding road towards a solution

Obviously, Requirement 1 and Write\LaTeX compatibility ruled out Ω \[8, 9, 10\], despite its powerful machinery for defining transliterations and applying contextual analysis. \texttt{XeLaTeX} was ruled out since it satisfies none of the requirements (1), (2), (4) or (5). The two remaining choices are pdf(e)\TeX\ [2] (with bidirectional typesetting support) and Lua\TeX\ [11].

Let us make a two-decade jump back in time. On April 12, 1993, the author organized a workshop entitled “\TeX\ and the Arabic Script”, at the National Institute of Oriental Languages and Civilizations (INALCO) in Paris. One of his contributions to this workshop was a public domain Arabic \TeX\ font in Geezah design, in which the contextual analysis was entirely done via \TeX\’s smart ligatures and boundary characters \[13\]. This is indeed the simplest approach: no extra technology was required other than \TeX–XET bidirectional typesetting and \TeX\ 3 smart ligatures. Nevertheless that font had a serious pitfall: because the number of glyphs was limited to 256, it was impossible to handle automatically quadriform letters followed by a vowel. In that case, it was necessary to introduce a vertical bar between the letter and the vowel, so as to obtain its final (or isolated) form (as for example, in \textasciicircum{\textcolor{red}{\textbullet}} = \textasciicircum{ktAbuN} which had to be written \textasciicircum{ktAb\mid uN}). The proceedings of this workshop were planned to be published in the \textit{Cahiers GUTenberg}, but this never happened — which is a pity since among the speakers was also the legendary creator of the Moroccan simplified Arabic writing system \[15, 17\], Ahmed Lakhdar-Ghazal (†2008).

As building a font based entirely on smart ligatures was an interesting approach, the author tried to investigate ways of succeeding now where he had failed in 1993, by the use of modern technologies.

A first idea was to create ligatures between letters and digits 0–3 to obtain contextual forms of the letters, and to have some other mechanism insert the digits. Indeed, Lua\TeX\ provides callbacks for both the token and the node list, which would be natural choices for such a mechanism. Traversal of the node list by Lua\TeX\ is quite efficient, but unfortunately unsuitable for this particular project since at that stage, ligatures have already been applied, so it is too late to insert digits among the nodes. After some attempts, the token list callback was abandoned since it is still in a very rudimentary state. Also, unfortunately, Lua\TeX\ has not yet implemented Lua patterns, which are a kind of regular expressions, and would make programming the contextual analysis much easier.

A second idea was to use large virtual fonts to encode all letter + diacritic combinations, so that the 1993 approach would be applied not to letters alone, but to letter + diacritic combinations. But the attempt to generalize the 1993 approach to large virtual fonts (OVF) has failed as well, because OVF\s do not provide support for Knuth’s boundary character (indeed, in Ω, \texttt{ΩTPs} provide a much more elegant solution to the word boundary problem, and therefore the boundary character was left out of the Ω system, but again Lua\TeX\ does not provide ΩTPs). In this approach, the default form of a letter would be the medial one, and boundary characters would turn a letter into initial, final or isolated form.

The third attempt proved successful: instead of using boundary characters, the author used smart...
ligatures between characters. In this case, the default letter form is the isolated one. The presence of a second letter changes the form of the first from isolated to initial, and the one of the second from isolated to final. A third letter will turn the second letter into middle form and the third letter into final (provided, of course, the second letter is quadriform), and so on. More details are given in §4.

2 The name

The next question was what to name the package.

Thanks to the Internet, search engines, social media, and the like, people are becoming more and more aware of other languages and writing systems. Why not give this package an Arabic name, be it a

FD
There are also rudimentary

TFM
OVF
change kerning and diacritic placement.

D

Finally adjustment files, in case the user wants to

UTF

D

3.1 Rationale of the transliteration

Here are the rules of the proposed transliteration (see Table 1):

1. pharyngeal ح = H, emphatic ص = S, ض = D, ط = T, ب = Z and velar ی = R are uppercased — do not confuse them with glottal ُ = h, non-emphatic س = a, د = d, و = т, ز = z, and alveolar ز = r;

2. long vowels (1 = k, ی = Y) and ālif maqṣūra (۷ = I) are also uppercased;

3. some consonants are modified by adding a character h (ب = dh, ُ = th, ش = sh);

4. the stand-alone hamza is obtained by a vertical bar | and letter ُayn by a grave accent (which, in legacy \TeX produces an inverted curly apostrophe, which is sometimes used to transliterate this letter);

5. to avoid confusion between pairs of letters and letters obtained by digraphs, one has to use a dash to separate characters: compare ١ = a-h and ش = sh, or ١ = t-h and ش = th:

6. more generally, the dash plays the role of zero-width joiner:1 when writing ب = -b, the letter ب will be in final form; ُ = -r and ُ = -b- will produce initial and middle letters, provided of course the letter is quadriform (as is letter ب in this example). This is very useful when describing grammar rules, to signify that a letter (or letter group) is an affix:

7. the dash can also be used to reestablish contextual forms when combined with \TeX commands, for example, to colorize letters as in Fig. 1. There is only one special case: when we want to colorize a letter of an isolated ligature، we add a digit 4 in front of the dash. For the final ligature، it will be a digit 5. Example: to colorize the lāms of ٢، write

\textcolor{red}{
\arabtt{l-	extcolor{red}{-}A4%}
\textcolor{red}{\{14-\}-A4}}

1 Except for the case of letter ُ which is biform and hence is not connected with the following letter. By writing ُ = d-h one obtains letters ُd and ُh, but the ُh is not in medial form, as it would be in any other case when preceded by a dash.

3 How to use ﺭ

The package provides three PostScript Type 1 fonts (plain, bold and typewriter), “real” fonts (regular TFM) and large virtual fonts (OVF and OFM files). There are also rudimentary FD and STY files, a MAP file, Perl scripts for conversion to (and from) UTF-8, the Perl script which builds the font and finally adjustment files, in case the user wants to change kerning and diacritic placement.

It requires Lua\TeX for change of direction and OVF/OFM compliance.

To typeset in Arabic, one need only load the dad package and use the macro \arab, which is a \texttt{long} macro: its argument may have multiple paragraphs.

Arabic text can be input in transliteration as described in Table 1 or in UTF-8. To obtain, for example,\footnote{1} the Perl script which builds the font and

\arabtt{}
\arab{

A simple Arabic typesetting system for mixed Latin/Arabic documents: ﺭ
8. finally, there is yet another use of the dash: when doubled, it produces a kashida stroke: compare ليل = 1Yl and ليل = 1--Y--1. There is also a \kesh command for extensible kashida (equivalent to a \hrulefill using the default rule thickness font dimension \fontdimen8): 1--\kesh--Y--\kesh--1. produces:

| Table 1: Transliteration of شـ system |
|-------------------------------|-----------------|
| ش | ا | ع | و | م | ن | ه | ع | م |
| a | A | h | x | s | N | h | R | m |
| t+ | b | d | S | c | + | f | q | n |
| t+ | th | dh | ة | i | t+ | k | l | b |
| t+ | j | z | ء | i | u | aN |
| t+ | + | a | u | aN |
| t+ | + | i | u | u |
| t+ | aN | iN | + + | a | o |
| t+ | aN | iN | +uN | a* | +a* |
| t+ | LLh |
| p | g | C | ِ | ُ |
| t+ | f | q |
| t+ | a** |

9. some digraphs start with an apostrophe: the hamza-carriers i = 'a, i = 'i, َ = 'u, ِ = 'i, ُ = 'A and also undotted letters bāʾ = 'b, nūn َ = 'n, fāʾ = 'f and qāf ُ = 'q:

10. other digraphs end with one or more asterisks: the most frequent one is the tāʾ marbūta َ = t* (which can be used also in initial and medial forms, and then becomes a regular tāʾ). The asterisk is also used for the wasla (which is only placed on the 'alif) َ = A* as well as for the vertical fathā (as in حَد = ha*dhA) and the madda. The latter is normally used only on the 'alif (َ = 'A) but can be found also in the notorious muqaddāštāt in the Koran, as in كَبَرَ (Koran 42:2) or كَبِيتَ حَنْسُ (Koran 19:1) — sometimes it is even combined with a šadda (as in كَبِيطَ حَلْسِ, Koran 7:1 and [21, p. 111] for the šadda); 11. a special transcription is provided for the ligature الل هـ = LLh used for the اسم الجلالة "noun of majesty", which is the name of God الل ه in this case — and in this case only — an uppercase L is used. The reason is that we wish to avoid ambiguity with other uses of the trigram لـ لـ (Koran 6:39) where we encounter letters لل but not with the meaning "God". In contrast to other systems, the لل ligature is available also in final form (for which occurs six times in the Koran, for example Koran 6:149), and it is possible to add diacritics to its first glyph (as in الله, Koran 2:115 or الله, Koran 2:165).

3.2 Unicode input

Input can be transliterated or provided directly in Unicode Arabic: \arab{Yanis} or \arab{Yanis} or even \arab{Yanis} or \arab{Yanis} will produce the same result: يانيس. All cells of Table 1 can be obtained by the corresponding Unicode characters (mostly via a single character, except for šadda + vowel combinations which require two characters). There is a special case, though: the الل ligature (see next section).

For the convenience of the user who wants to write kashida (so that Arabic input is not disrupted) we have defined a command (in Arabic characters) (which are the first two letters of لـ (tawыل, the Arabic name of kashida) which is exactly equivalent to \kesh and has to be placed between Unicode U+0640 ARABIC TATWEEL characters.

Yannis Haralambous
Weak. Weak verbs are those with one or more weak letters (ء و or ي) as radicals. There are four sub-classes:

- **Assimilated.** Assimilated verbs have *initial* ء or (much more rarely) ي, and two sound radicals or middle َ and a sound final radical. Typical doubled roots are رَحَّل, يُؤُثِر.
- **Hollow.** Hollow verbs have *middle* و or ي and two sound radicals or initial َ and a sound final radical. Typical hollow roots are ُل، يد.
- **Defective.** Defective verbs have *final* و or ي and two sound radicals. Typical roots are ُن، يد.
- **Doubly weak.** Doubly weak verbs have two weak radicals، ي or َ. Typical doubly weak roots are ُ، ُ. ُل، يد.

In the middle of a verse *hamzah* is merged with the final vowel of the preceding word, e.g., کَتَبَ اللَّهُ الْإِنْسَانَ، He created man. Note that the ء or ي has joined the ج or ي and while the *hamzah* sign has disappeared from the text, ‘الْف’ is retained but it is not pronounced.

**Practice text 11**
1. Man says — َالْإِنْسَانُ
2. Does man think? — أَيْضًا الْإِنْسَانُ
3. He said: How long hast thou tarried? — قَالَ كَمْ — خَصْصَ الْحَقَّ
4. The truth is out — َُلا بَالْأَمْثَالِ
5. He created man from dry clay — َُلا بَالْأَمْثَالِ

In the middle of a verse *emph* و is merged with the final vowel of the preceding word, e.g., َالْإِنْسَانُ، He created man. Note that the ء or ي has joined the ج or ي and while the *hamzah* sign has disappeared from the text, ‘الْف’ is retained but it is not pronounced.

\begin{itemize}
\item Weak verbs are those with one or more weak letters (ء or ي) as radicals. There are four sub-classes:
\item **Assimilated.** Assimilated verbs have *initial* ء or (much more rarely) ي, and two sound radicals or middle َ and a sound final radical. Typical doubled roots are رَحَّل, يُؤُثِر.
\item **Hollow.** Hollow verbs have *middle* و or ي and two sound radicals or initial َ and a sound final radical. Typical hollow roots are ُل، يد.
\item **Defective.** Defective verbs have *final* و or ي and two sound radicals. Typical roots are ُن، يد.
\item **Doubly weak.** Doubly weak verbs have two weak radicals، ي or َ. Typical doubly weak roots are ُ، ُ. ُل، يد.
\end{itemize}

Figure 2: Sample \LaTeX document using $\LaTeX$ (16, p. 5) and (21, p. 54–55)
3.2.1 The ﬁnal ligature and Unicode

The ﬁnal ligature is traditionally used for writing the name of God: ﷲ. It can be found in religious texts, but also in expressions (for example, إِن شاء الله which means “hopefully” appears even in the French language as inchallah and in Portuguese as oxalá) and in the very common surname Abdullah.

The problem with this ligature is that it contains a rather rare diacritic (a َساندا combined with a vertical ﬁatha — the latter is available in the Apple Arabic keyboard layout but not the Microsoft one) and, as a convenience, most standard fonts will replace the character string ﻟﻢ--سر (which would normally look like ﷲ) by the complete ligature ﷲ: in other words: the font not only changes the glyphs but, at the same time, also adds the diacritics. This behavior is barely legitimate: a ligature (as in ‘i’ or ‘y’) is normally limited to a change of glyphs, and should not add new characters (in this case, characters U+0651 ARABIC SHADDA and U+0671 ARABIC LETTER SUPERSCRIPT ALEF) since this means that what is rendered no longer corresponds to the underlying Unicode character string.

Nevertheless, for the user’s convenience, we have adopted that behavior also in ﺚ،، but only in the case of Unicode input. Therefore when the user types Unicode ﻟﻢ--سر (the ﬁrst ﻟﻢ must not be preceded by a quadriform letter), the system will produce the ﷲ ligature.

This method will not work if a diacritic is inserted between the two ﻟﻢ, or if the ﬁrst ﻟﻢ follows a quadriform letter and hence will be medial. For that case, we have deﬁned a macro ﻟﻢ/ (the macro name is in Arabic script so that right-to-left direction is not disrupted) which takes an argument: the vowel between the two ﻟﻢ. Hence, to obtain ﻟﻢ/ the user can choose between one of the following:

\( \text{فَلَّا} \)

faLhi

The dotted circle, used to show the combining nature of short vowels and other diacritics, can be obtained by the macros \arabdottedcircle or دائره/ with the macro name in Arabic script.

4 TExnicalities

The ﺚ، font is a tour de force of smart ligature use: for example, to obtain \( k_{2}t_{2}b_{3} \) (كتب) out of \( kb_{t} = k_{0}t_{0}b_{0} \) (or \( "0643"062A"0628 \) input, one needs the following smart ligatures:

\[
\begin{align*}
& k_{0} \text{ LIG} / t_{0} \rightarrow k_{1} t_{0} \\
& k_{1} \text{ LIG} / t_{0} \rightarrow k_{3} t_{3} \\
& t_{3} \text{ LIG} / b_{0} \rightarrow t_{2} b_{0} \\
& t_{2} \text{ LIG} / b_{0} \rightarrow t_{2} b_{3}
\end{align*}
\]

Yannis Haralambous

![Figure 3: Finite state automaton starting with an isolated ﻟﻢ (’اَلِиф ١ stands for the set of letter ﺉ = \( \{ l, ١, ٢, ٣ \} \); ٤ stands for any Arabic letter besides ﻟ and set ﻟ.](image)
OVF, it is mandatory to use tool `wovp2ovf` of version higher than “1.13 (build 34787)”, which will be included in TeX Live 2015.

Names of PostScript glyphs are standard,² so that copy-paste from a PDF file results in almost perfect Unicode strings.

4.1 Conversion to and from UTF-8
As a tool for users, we provide two Perl scripts allowing conversion from UTF-8 to our transliteration scheme and back. These scripts can be applied selectively using, for example, the feature of many advanced text editors of applying text filters to selected text areas.

5 Conclusion
There was a period (in the early days of non-Latin-alphabet TeX [4, 5, 14]) where transliteration of input text was the only available method. Then, when Unicode was sufficiently widespread, TeX switched to tools allowing direct non-Latin input. In the case of Arabic, because of the particular characteristics of this script, this is — even today — not always the optimal solution, especially when we are dealing with short extracts of Arabic text combined with Latin-alphabet text and TeX commands. Maybe now is the time to return to methods based on transliteration, as an alternative to direct-input methods. We have implemented this approach, using only smart ligatures, as defined by Donald Knuth in 1990 [13], and the large virtual font format introduced by Ω and taken over by LuaTeX.

We hope that this package will be useful to users seeking a straightforward method to introduce short Arabic extracts into Latin-alphabet documents.

References
[5] Yannis Haralambous. Towards the revival of traditional Arabic typography through TeX.

² These names are either taken from the Adobe Glyph List [1], or using the standard convention `uniXXX.var` where `XXX` is the Unicode position of the character and `var ∈ {ini, med, fin, iso}`. Ligatures are named by the names of their components, concatenated using underscores.


A simple Arabic typesetting system for mixed Latin/Arabic documents: ِذَد

Yannis Haralambous
Institut Mines Télécom, Télécom Bretagne,
UMR CNRS 6285 Lab-STICC
Technopôle Brest Iroise CS 83818,
29238 Brest Cedex 3, France
yannis.haralambous (at) telecom-bretagne dot eu