Writing and transliterating Swahili in Arabic script with Andika!

Kevin Donnelly

2014-17
(This version compiled 1 December 2017)

Andika! is dedicated to the memory of Sheikh Yahya Ali Omar (1924–2008)

kevindonnelly.org.uk/swahili

Andika! is free software under the GPLv3 or later.
All trademarks belong to their respective owners.
## Contents

1 Introduction ................................................................. 2

2 Examples of Andika output ............................................. 3
   2.1 Converting Roman to Arabic script .......................... 3
   2.2 Replicating prose in Arabic script .......................... 5
   2.3 Replicating manuscript poetry: Bajuni fishing songs .... 5
   2.4 Replicating manuscript poetry: Utenzi wa Mkunumbi ... 6
   2.5 Replicating manuscript poetry: Kiswahili .................. 8
   2.6 Replicating manuscript poetry: Mama musimlaumu .... 9
   2.7 Replicating manuscript poetry: Utenzi wa Rasi 'Ighuli 10
   2.8 Replicating manuscript poetry: Qasida ya Burda ....... 10

3 Getting started .............................................................. 12
   3.1 Website ................................................................. 12
   3.2 Introducing Ubuntu ................................................. 12
   3.3 Typing Swahili in Arabic script .............................. 13
   3.4 Converting and annotating Swahili in Arabic script .... 13
   3.5 Next steps ............................................................. 13

4 Fonts .............................................................................. 14
   4.1 Missing glyphs in Arabic fonts ............................... 14
   4.2 Default fonts in Andika! ......................................... 15
   4.3 Adding missing glyphs to Arabic fonts .................... 16
   4.4 Scheherazade and Amiri .......................................... 16

5 A keyboard layout for Swahili in Arabic script ............... 18
   5.1 Introduction ............................................................. 18
   5.2 Governing principles for the layout ......................... 18
   5.3 Changing the layout ............................................... 19

6 Writing contemporary Swahili in Arabic script ............... 20
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.1</td>
<td>Introduction</td>
<td>20</td>
</tr>
<tr>
<td>6.2</td>
<td>General principles</td>
<td>20</td>
</tr>
<tr>
<td>6.3</td>
<td>Representation of consonants</td>
<td>20</td>
</tr>
<tr>
<td>6.4</td>
<td>Representation of vowels</td>
<td>22</td>
</tr>
<tr>
<td>6.5</td>
<td>Vowel sequences</td>
<td>23</td>
</tr>
<tr>
<td>6.5.1</td>
<td>Stressed + unstressed vowel sequences</td>
<td>23</td>
</tr>
<tr>
<td>6.5.2</td>
<td>Unstressed + stressed vowel sequences</td>
<td>24</td>
</tr>
<tr>
<td>6.5.3</td>
<td>Unstressed vowel sequences</td>
<td>24</td>
</tr>
<tr>
<td>6.5.4</td>
<td>Longer vowel sequences</td>
<td>24</td>
</tr>
<tr>
<td>6.6</td>
<td>Comparing conventions</td>
<td>25</td>
</tr>
<tr>
<td>6.6.1</td>
<td>Sakani on long vowels</td>
<td>25</td>
</tr>
<tr>
<td>6.6.2</td>
<td>Marking short vowels</td>
<td>25</td>
</tr>
<tr>
<td>6.6.3</td>
<td>Sakani on consonants</td>
<td>25</td>
</tr>
<tr>
<td>6.6.4</td>
<td>Distinction between syllabicity and prenasalisation</td>
<td>26</td>
</tr>
<tr>
<td>7</td>
<td>Converting from one script to the other</td>
<td>27</td>
</tr>
<tr>
<td>7.1</td>
<td>Introduction</td>
<td>27</td>
</tr>
<tr>
<td>7.2</td>
<td>Cut-and-paste converters</td>
<td>27</td>
</tr>
<tr>
<td>7.2.1</td>
<td>Arabic to Roman</td>
<td>27</td>
</tr>
<tr>
<td>7.2.2</td>
<td>Roman to Arabic</td>
<td>28</td>
</tr>
<tr>
<td>7.2.3</td>
<td>Convert a webpage</td>
<td>28</td>
</tr>
<tr>
<td>7.3</td>
<td>Command-line converter</td>
<td>28</td>
</tr>
<tr>
<td>7.3.1</td>
<td>Point-and-click interface</td>
<td>29</td>
</tr>
<tr>
<td>7.3.2</td>
<td>Command-line input</td>
<td>30</td>
</tr>
<tr>
<td>7.3.3</td>
<td>Layout of the input document</td>
<td>30</td>
</tr>
<tr>
<td>7.3.4</td>
<td>Converting pdfs</td>
<td>31</td>
</tr>
<tr>
<td>7.4</td>
<td>General notes on usage</td>
<td>31</td>
</tr>
<tr>
<td>7.4.1</td>
<td>Adjusting the import's stanza numbering</td>
<td>31</td>
</tr>
<tr>
<td>7.4.2</td>
<td>Warning when using the convert tools</td>
<td>32</td>
</tr>
<tr>
<td>7.5</td>
<td>How the conversion works</td>
<td>32</td>
</tr>
<tr>
<td>7.5.1</td>
<td>Arabic to Roman</td>
<td>32</td>
</tr>
<tr>
<td>7.5.2</td>
<td>Roman to Arabic</td>
<td>33</td>
</tr>
<tr>
<td>8</td>
<td>Typesetting poetry</td>
<td>35</td>
</tr>
</tbody>
</table>
8.8.3 Making entries in the noshow field ........................................ 52

9 Typsetting multiple versions of the same poem .................................. 54
  9.1 Set up a manuscript information table ........................................ 54
  9.2 Set up a running order table .................................................. 54
  9.3 Add manuscript data to the running order table .............................. 54

10 Computer-aided analysis of poem texts .......................................... 56
  10.1 Additional annotation fields .................................................. 56
      10.1.1 root field ..................................................................... 56

A Installing Andika! ............................................................................ 58
  A/1 How much of this do I need to do? ............................................ 58
  A/2 Ubuntu Linux .......................................................................... 58
  A/3 Conventions ............................................................................ 59
  A/4 Running Ubuntu as a virtual machine ....................................... 59
  A/5 Change the desktop to KDE ..................................................... 60
  A/6 Download Andika! ................................................................. 60
      A/6.1 Option 1: snapshot ......................................................... 60
      A/6.2 Option 2: easy update .................................................... 61
      A/6.3 Move the andika directory .............................................. 61
  A/7 Install fonts ............................................................................ 62
  A/8 Set up a new language and keyboard ........................................ 63
      A/8.1 Activate the new keyboard in KDE .................................... 63
      A/8.2 Activate the new keyboard in Unity .................................. 63
      A/8.3 Interaction with the unlock screen in KDE ......................... 64
  A/9 LibreOffice .............................................................................. 64
      A/9.1 Configure the word-processor ....................................... 64
      A/9.2 Install a template ............................................................ 65
  A/10 PHP ....................................................................................... 65
      A/10.1 Install PHP .................................................................. 65
      A/10.2 Configure PHP ............................................................. 66
  A/11 PostgreSQL ............................................................................ 66
      A/11.1 Install PostgreSQL ......................................................... 66
## List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Part of the Swahili Wikipedia page on utamaduni (culture)</td>
<td>3</td>
</tr>
<tr>
<td>2.2</td>
<td>The page in Figure 2.1 automatically transliterated into Arabic script</td>
<td>4</td>
</tr>
<tr>
<td>2.3</td>
<td>Bajuni fishing songs as written out by Sheikh Yahya Ali Omar</td>
<td>6</td>
</tr>
<tr>
<td>2.4</td>
<td>Stanzas 3–5 of Utenzi wa Mkunumbi</td>
<td>7</td>
</tr>
<tr>
<td>2.5</td>
<td>The first stanza of Mama musimlaumu</td>
<td>9</td>
</tr>
<tr>
<td>2.6</td>
<td>Stanza 2280 of Utenzi wa Rasi 'lGhuli</td>
<td>10</td>
</tr>
<tr>
<td>2.7</td>
<td>The first two stanzas of Qasida ya Burda</td>
<td>11</td>
</tr>
<tr>
<td>4.1</td>
<td>Stanza 6 from Muhammad Kijuma’s manuscript of Utenzi wa Mwana Kupona</td>
<td>14</td>
</tr>
<tr>
<td>5.1</td>
<td>Keyboard layout for writing Swahili in Arabic script</td>
<td>18</td>
</tr>
<tr>
<td>5.2</td>
<td>Accessing the glyphs on the keys</td>
<td>18</td>
</tr>
<tr>
<td>7.1</td>
<td>Vipande separated by a space</td>
<td>30</td>
</tr>
<tr>
<td>7.2</td>
<td>Vipande separated by a star</td>
<td>30</td>
</tr>
<tr>
<td>8.1</td>
<td>Default output</td>
<td>44</td>
</tr>
<tr>
<td>8.2</td>
<td>Output with larger font size</td>
<td>45</td>
</tr>
<tr>
<td>8.3</td>
<td>Option: firstcolour</td>
<td>46</td>
</tr>
<tr>
<td>8.4</td>
<td>Option: nocolour</td>
<td>46</td>
</tr>
<tr>
<td>8.5</td>
<td>Option: alignright</td>
<td>46</td>
</tr>
<tr>
<td>8.6</td>
<td>Option: noenglish</td>
<td>47</td>
</tr>
<tr>
<td>8.7</td>
<td>Option: nostandard + noenglish + alignright</td>
<td>47</td>
</tr>
<tr>
<td>8.8</td>
<td>Option: noarabic + alignleft</td>
<td>47</td>
</tr>
<tr>
<td>8.9</td>
<td>Option: close-lr</td>
<td>48</td>
</tr>
<tr>
<td>8.10</td>
<td>Option: close-rl</td>
<td>48</td>
</tr>
<tr>
<td>8.11</td>
<td>Option: close-rl + nostandard</td>
<td>49</td>
</tr>
<tr>
<td>8.12</td>
<td>Making an entry in the noshow field</td>
<td>49</td>
</tr>
<tr>
<td>8.13</td>
<td>Suppressing the transcription for a word</td>
<td>50</td>
</tr>
<tr>
<td>8.14</td>
<td>Omitting a word</td>
<td>50</td>
</tr>
</tbody>
</table>
8.15 Marking the words of the first line in the stanza ........................................ 51
8.16 Suppressing a line ......................................................................................... 51
8.17 Suppressing a line but retaining a translation ............................................... 51
8.18 Swapping in the close transcription .............................................................. 52

A.1 Setting up the Swahili keyboard for KDE ...................................................... 63
List of Tables

4.2 Glyphs commonly missing in fonts ........................................ 15
5.1 Typing examples .................................................................... 19
6.1 Representation of consonants .................................................. 22
6.2 Representation of single vowels .............................................. 23
6.3 Vowel-carriers ..................................................................... 23
Chapter 1

Introduction

For centuries, Swahili was written in Arabic script, and hundreds of manuscripts in collections around the world testify to its long tradition of written literature. Over the last century, however, Swahili in Roman script has become the norm.

**Andika!** (meaning Write! in Swahili) has two aims. The first is to make Swahili in Arabic script as easy to use as Swahili in Roman script – it is equally easy to read and write the the language in either script. The tools, based on the work of Marehemu Mu’allim Sheikh Yahya Ali Omar (Omar and Frankl, 1997) provide a consistent, standardised transliteration of Swahili in Arabic script, and a one-to-one mapping of this to Swahili in Roman script. Documents can be typed in either script, and automatically transliterated to the other.

- New writing in Swahili can be composed in Arabic script and published easily via word-processors, webpages, or pdfs created by typesetting systems such as LaTeX.
- The ability to convert Arabic script at any time into Roman script means that there is very little overhead involved in choosing to write Swahili in Arabic script. Material can be produced simultaneously in both scripts with the minimum of effort (although the converted text will need minor editing to cover such things as capital letters, which do not exist in Arabic script).
- Existing Swahili content in Roman script can be converted to Arabic script, making it possible to reuse content already published in Roman script. This means that large amounts of material in Arabic script can be be made available very quickly.
- The Roman-to-Arabic conversion can be adjusted to convert numerals, to add or remove markers such as sakani (sukun), and so on.

The second aim of **Andika!** is to allow the creation of digital versions of existing Swahili manuscripts written in Arabic script.

- Perishable Swahili manuscripts in Arabic script can be directly transcribed and made available in digital format, which is more versatile than a photocopy or scan of the manuscript. At present, most Swahili literature from earlier periods has only been published in Roman transliteration, even though the manuscripts were written in Arabic script.
- A direct transcription can be augmented with a fully-vocalised Arabic transcription, a close phonetic transliteration (a variety of different ones can be easily created), a transliteration in the standard Roman orthography, and so on. The tools allow much of these to be generated automatically, reducing the effort this would otherwise involve.
- A critical apparatus (English translation, notes on words, variant readings, emendations, etc) can easily be added to the digital version, with high-quality typeset output in a variety of formats.
- Apart from allowing easier typesetting and dissemination, having manuscripts in digital form will make it possible for the first time to use computers to look at word frequency, stylistic variation, etc, within the texts, to build corpora for classical Swahili, and so on.

**Andika!** is licensed under version 3 of the Free Software Foundation’s General Public License.¹ This means that, apart from costing nothing to use, it can be adapted and extended as required by the user, subject to the same license being used for any new version thus created.

¹[http://www.gnu.org/licenses/gpl.html](http://www.gnu.org/licenses/gpl.html)
Chapter 2

Examples of Andika! output

2.1 Converting Roman to Arabic script

Existing text in Roman script can be easily converted to Arabic script. Figure 2.1 is a section from the Swahili Wikipedia page on utamaduni (culture), and Figure 2.2 shows this page after being converted automatically to Arabic script using the conventions for standard spelling proposed in Andika!

![Wikipedia Swahili page on utamaduni](image)

Below are typeset versions of one paragraph in both scripts:

Dhana hii ilipoibuka kwanza katika karne ya kumi na nane na kumi na tisa barani Ulaya, ilimaanisha mchakato wa kilima au kuboresha kilimo au kilimo cha mboga. Katika karne ya kumi na tisa dhana hii ilimaanisha kuboreshwa kwa mja kupitia kwa elimu na hali kadhalika ilimaanisha kutekelezwa kwa maazimio ya kitaifa au maadili. Katikati mwa karne ya kumi na tisa, wanasyanshi wengine waliitumia dhana ya utamaduni kurejelea uwezo wa kiubia wa binadamu.

ذَان هُمْ يَلْبِسُوْنَ كَيْبَكَ كَارَنْ يَكُومُنِّ نَزَانِ نَكُومُ نَتَّيْسِ نْزَانُ أَلَاََّيِّ،

إِلْبَشِيَّةُ مَيْكِكَاتْ وَ كُلِيمُ أوْ كَلِيمُ كَلِيمُ أوْ كَلِيمُ كَلِيمٍ تُمَيْنُ مَيْنُ. كَيِيبَكَ كَارَنْ يَكُومُ
Mafanikio ya Siti yameelezwa kwa kufanana matatizo aliyoyapata katika jamii yake katika kuendelea na hatua za kujinynayua kiuchumi, hatua hivyooalipambana nayo na alweza kufaniko. Historia ya Siti imejitokeza kuwa ya kipekee kutokana na matendo yake katika jamii iliyo na utamaduni wa kuwaweza wanawake kutojitokeza hadharani hasa kwa kuimba kwa wakati huo. Siti akiva mwanamke aliyeypata misukosuko mbalimbali ya kukathisha tamaa katika maisha yake ikiwemo ya ndoa, alweza kuhimili na kupambana nayo na kuweza kufikia kuwa mtu marufu na wa kuheshimika ndani na hata nje ya mipaka ya jamii yake.
2.2 Replicating prose in Arabic script

The following is a copy of the specimen text from Appendix C of Omar and Frankl (1997), which was included to show how their system would look in practice – the text itself is from Omar (1998). The conventions used here (eg the omission of short vowels in certain circumstances) differ slightly from those proposed in Andika! – see Section 6.6 for further discussion.

All of a sudden we saw a very high mountain which blocked the road. So we climbed the mountain; its sand was like gold, and its stones were like rubies and seed-pearls. Well then, as we continued on our way, we came across a tree the like of which I had never before seen. Beneath it was a youth tending goats. The horns of those goats were green like emeralds, and their silken fleeces were of divers colours, while their milk which dripped down was as white as the milk of the rivers of Paradise.

2.3 Replicating manuscript poetry: Bajuni fishing songs

Figure 2.3 is part of a manuscript rendering of Bajuni fishing songs collected by Sheikh Yahya Ali Omar (Donnelly and Omar, 1982). A letter-for-letter transcription of that follows, with an automatically-generated close transliteration in Roman script. The Roman conversion uses various diacritics to reconcile the manuscript’s representation of the Bajuni dialect with standard orthography.
Figure 2.3: Bajuni fishing songs as written out by Sheikh Yahya Ali Omar

Peace to you, and to you peace. The salaam is for those outside, the hodi is for those inside. And this greeting is for war -- do not think it is for peace. Now we will burn incense -- what learned man shall we call? We’ll call an Mfirado, and then a man from Koyamani. A horn is my sign of strength -- I eat with molars inside.

2.4 Replicating manuscript poetry: Utenzi wa Mkunumbi

Harries (1967) is one of the few books of Swahili classical poetry to include the text in Arabic script in addition to the Roman transcription, in this case a photocopy of a copy made by Sheikh Yahya Ali Omar of the original manuscript. The Arabic script in that manuscript is less well-adapted to Swahili --
for instance, o is not used consistently. Figure 2.4 shows stanzas 3-5 of the utenzi.

![Image of handwritten text]

Figure 2.4: Stanzas 3–5 of Utenzi wa Mkunumbi

A letter-for-letter copy of the manuscript is shown below. In this case the automatically-generated transcription was suppressed and replaced by Harries’ own transcription, which was added manually and coloured green.

3. Two powers were in conflict / Shekuwe and Bwana Simba / opposing one another for sport / by day and by night.

4. When three days had passed / Shekuwe wanted men / to bring his offering / and he bought himself a cow.

5. And he sent the cow on the way / a good one without blemish / and Sheikh Simba observed it / [and said] What is the point of a single cow?

An automatically-generated close transcription can be printed out separately if desired, as shown below. In this case, an alternative layout has been selected, where the vipande are each in their own column,
instead of both being on one line.

<table>
<thead>
<tr>
<th>3a/b</th>
<th>3c/d</th>
</tr>
</thead>
<tbody>
<tr>
<td>ḏōla mbili ziliwāna</td>
<td>kamātezo kushindāna</td>
</tr>
<tr>
<td>shikuwe nāsimba mbawāna</td>
<td>mtāna nālayliya</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>4a/b</th>
<th>4c/d</th>
</tr>
</thead>
<tbody>
<tr>
<td>zikiṭimu siku tātu</td>
<td>kṣtukuwa chāke kiṭu</td>
</tr>
<tr>
<td>shikuwe kāṭaka wātu</td>
<td>ngūbe kāy nunuliya</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>5a/b</th>
<th>5c/d</th>
</tr>
</thead>
<tbody>
<tr>
<td>kaṭiya ngūbe ndiyāni</td>
<td>simba shīi kabaīni</td>
</tr>
<tr>
<td>mema asiyu laḥāni</td>
<td>mpāni ngūbe mmuya</td>
</tr>
</tbody>
</table>

2.5 Replicating manuscript poetry: Kiswahili

Abdulkadir and Frankl (2013) presents an annotated edition of the first author’s poem, كِسْوَاحِلِ. It is rare among published work on Swahili in including the original Arabic script of the poem. The following is a letter-for-letter transcription of stanza 4 of Sheikh Mahmoud’s manuscript, with the exception that the damma-with-tail occasionally used by him to signify o is denoted here with inverted damma, since the font does not yet include that glyph. The full text of the poem is in Appendix D, with a slightly different layout in Appendix F.

The layout includes an automatically-generated close and standard transliterations (the latter corrected manually where necessary), and the English translation and notes from the paper. The Arabic text and the close transcription are set out in columns, so that the close transcription relates directly to the kipande above it, while the standard transliteration and the English translation are set out on a single line, so that they can be read in conjunction.

Different fonts can be used for each layer of the text (the transliterations use sans serif fonts, while the translation uses a serif font in a smaller size), and each layer can be coloured (the standard transliteration is in green, while the close transcription and translation are in shades of grey. An epenthetic vowel has been added in blue in kipande 4b. The footnotes are marked in red and appear at the bottom of the page.

---

1 Bwana Muyaka was the outstanding Swahili poet of 19th century Mombasa. After his death many of his verses were recalled by Mu’allim Sikujua Abdallah al-Batawi (died 1890) and transcribed with annotations by W.E. Taylor (1856-1927). After Taylor’s death his papers were acquired by the library of the School of Oriental and African Studies (SOAS), London.
2 Mwengo Athmani: this 18th century poet from Pate composed the Utendi wa Tambuka (The Epic of Heraklios).
CHAPTER 2. EXAMPLES OF ANDIKI! OUTPUT

2.6 Replicating manuscript poetry: Mama musimlaumu

The following is a stanza from another poem by Sheikh Mahmoud. In this transcription of the first stanza of the manuscript, the close and standard transliterations have been suppressed, leaving only the Arabic-script text and the English translation. Each *kipande* is centred, and the Arabic-script text has been coloured green.

---

**Figure 2.5: The first stanza of Mama musimlaumu**

1. I have been compelled to talk before my time

2. because I have seen them condemning my mother

3. my mother is blameless, she is not the only one who is guilty

---


5 Bwana Mataka's full name is Muhammad bin Shee Mataka al-Famau (1825-1868). He was ruler of Siyu, as was his father. His mother was Mwana Kupona, famous for the poem of advice written to her daughter. Bwana Mataka died in Mombasa's fort while imprisoned by the Busa'idi.
2.7 Replicating manuscript poetry: Utenzi wa Rasi 'lGhuli

This is a transcription of a stanza (2280) from the MS sample included in Leo van Kessel’s edition of *Utenzi wa Rasi ‘lGhuli* by Mgeni bin Faqhi (bin Faqhi, 1979). The published standard Swahili text is shown in green. This ballad was composed around 1850, and at over 4,500 stanzas is the longest Swahili ballad in existence. The copyist of this MS writes only 3 vowels, uses *ain* with dot to represent ng[w], and tends not to mark nasalised or labialised consonants (e.g. كَذِرَعُ in 2280d).

**Figure 2.6: Stanza 2280 of Utenzi wa Rasi ‘lGhuli**

2280

<table>
<thead>
<tr>
<th>سانى</th>
<th>ڤهَمُ</th>
<th>مَبَنَيِتَبَيِنِ</th>
<th>نَاكَحَادِقٌ</th>
<th>فَهَامُٕني</th>
<th>مَباَنَيِتَبَيِنِ</th>
<th>نَاكَحَادِقٌ</th>
</tr>
</thead>
<tbody>
<tr>
<td>مَباَنَيِتَبَيِنِ</td>
<td>نَاكَحَادِقٌ</td>
<td>فَهَامُٕني</td>
<td>مَباَنَيِتَبَيِنِ</td>
<td>نَاكَحَادِقٌ</td>
<td>فَهَامُٕني</td>
<td>مَباَنَيِتَبَيِنِ</td>
</tr>
</tbody>
</table>

2.8 Replicating manuscript poetry: Qasida ya Burda

*Qasidas* are panegyric poems in Arabic eulogising the Prophet, often performed as part of a religious ceremony. They have spread widely throughout the Muslim world, and are often adapted to local languages (Knappert, 1971; Sperl and Shackl, 1995). The *Qasida ya Burda*, قَصيْدَةُ الْبُرْدَةُ, was composed by Muhammad bin Sa’idi al-Busiri in the 1300s, and rendered into Swahili verse by Sheikh Muhammad bin Athman Hajji al-Hilali Mshela (1840-1930) (wa Mutiso, 1996). In the extract below, the Arabic verse is in blue, and the Swahili verse is in green. The transliteration is given as footnotes to each line.

![Figure showing a page from a Swahili manuscript]

<table>
<thead>
<tr>
<th>أَمِنْ</th>
<th>نَذَكُّرِ</th>
<th>جِيْرَانٍ</th>
<th>نِذِيْسَلَمٍ</th>
<th>مَزَجْتَ</th>
<th>دَمْعًا</th>
<th>جَرَى</th>
<th>مِفَقْلَةٍ</th>
<th>بِدَمٍ</th>
</tr>
</thead>
<tbody>
<tr>
<td>نِكَڪُكُمْ</td>
<td>بُكَجِرَنِ</td>
<td>نْيَٖمَ</td>
<td>وَلِيٗکٗهَپٗ</td>
<td>نِذِىسَلَمِ</td>
<td>مَبَنَيِتَبَيِنِ</td>
<td>نَاكَحَادِقٌ</td>
<td></td>
<td></td>
</tr>
<tr>
<td>اُمِلْتَنْ</td>
<td>نَگَنْيَ</td>
<td>تَزِكْ</td>
<td>كْوَمَبَمَعَنَا</td>
<td>يِهَيْسٖ</td>
<td>سَمَآَ</td>
<td>كَوْمُبَ مَعَنَا</td>
<td>نَفِقٌ</td>
<td>سَمَآَ</td>
</tr>
</tbody>
</table>

6. амин надхаккури жирани нидхи саламин * мазажта дан’анярау мин муаллатин бидамин
7. НИККЕКУМБАК ЖЕН НИМ * УЛИКЕ ХЕБ НИДИ СЕЛМ
8. АМЕЛТЭНГИ НЕ КУМО ДЕМ * КООМБ МУУНЯНИ НЕФИ СЕМУА
Figure 2.7: The first two stanzas of Qasida ya Burda
Chapter 3

Getting started

3.1 Website

The website allows you to experiment with Andika! regardless of the operating system (Microsoft Windows, Apple Mac OS, GNU/Linux, Android, etc) on your computer or device. All you need to do is install the Scheherazade font so that all the Arabic glyphs (characters) used in Swahili are available.

In the Roman to Arabic section of the website you can type into a box in Roman script and have the input converted into Arabic script, or you can input a web address and have that whole page converted into Arabic script. You can cut and paste the converted Arabic text into a word-processor. The Arabic to Roman section of the website lets you convert Arabic script into standard Roman orthography.

3.2 Introducing Ubuntu

The website offers only limited functionality – to use Andika! fully, it is best to install it on your own computer. Andika! was developed on GNU/Linux, a free, secure, and versatile operating system which is not owned by any one company – much of the internet runs on GNU/Linux, and large internet companies such as Google, Amazon and Facebook use it extensively.

The specific “flavour” of GNU/Linux used is Ubuntu. Ubuntu was started by a South African, Mark Shuttleworth, and the name is cognate with Swahili أُتُ (utu, humanity), so it is apt for a project like Andika! It is highly recommended to download Ubuntu and install it as your main operating system, but if that is not possible the next best thing is to run it in a virtual machine on top of Microsoft Windows or Apple Mac OS by installing VirtualBox and then installing GNU/Linux into that.

Microsoft Windows or Apple Mac OS, which are owned by single companies, offer only a single desktop (interface to the operating system). But with GNU/Linux it is possible to choose from a variety of desktops. By default, Ubuntu comes with the Unity desktop, but the instructions here are mostly for the KDE desktop, since that is what I use.

Detailed instructions for installing Andika! and the other software it requires are in Appendix A.

---

1 kevindonnelly.org.uk/swahili
2 scripts.sil.org/cms/scripts/page.php?item_id=Scheherazade
3 en.wikipedia.org/wiki/Linux
4 Most of Andika! will work on Microsoft Windows or Apple Mac OS, but the crucial part (keyboard layout and activation) will not, since keyboard handling differs between operating systems – I would be happy to accept appropriate layout files for operating systems other than GNU/Linux.
5 ubuntu.com
6 ubuntu.com/download/desktop
7 ubuntu.com/download/desktop/install-ubuntu-desktop
8 virtualbox.org
9 unity.ubuntu.com
10 kde.org
11 I would be happy to include details for other desktops if anyone sends them to me.
3.3 **Typing Swahili in Arabic script**

If you simply want to type Swahili in Arabic script, it’s very easy to get started:

1. Download **Andika!** (A/6.1) in a zip file.
2. Unzip the file.
3. Move into the `andika` folder created.
4. Install the Scheherazade font so that all the Arabic glyphs (characters) used in Swahili are available (A/7).
5. Install a keyboard so that the Arabic letters can be typed (A/8).
6. Configure the LibreOffice word-processor to handle Arabic script (A/9).

3.4 **Converting and annotating Swahili in Arabic script**

The above will not allow you to convert automatically from one script to the other – for that you need to do the full installation in Appendix A. This will also allow you to transliterate, edit and annotate Swahili documents in Arabic script – see Chapter 7 and Chapter 8.

3.5 **Next steps**

Chapter 4 reviews some font-related issues.

Chapter 5 explains the keyboard layout used in **Andika!**, and how to access the various glyphs it caters for.

Chapter 6 sets out proposed conventions for standard spelling of Swahili in Arabic script, which are used when converting between the standard Roman script and Arabic script and vice versa.

Chapter 7 demonstrates how to convert between both scripts, in either direction, and gives an overview of how the conversion works.

Chapter 8 shows how Swahili poetry manuscripts in Arabic script can be transcribed to produce attractive output in various digital formats, including transliteration, translation, notes, emendations, variant readings, and so on, with the added benefit that the contents of the manuscripts are then available for computer analysis of language, vocabulary, word-frequency, etc.
Chapter 4

Fonts

4.1 Missing glyphs in Arabic fonts

In order to see Arabic script properly, the font you are using must contain Arabic glyphs (characters). A number of fonts developed especially for Arabic are available, but many of them contain only the glyphs needed to write standard Arabic.

If you are using Andika! to transcribe older manuscripts, it may be that these glyphs will be all you need, since many Swahili writers in the past used the Arabic script to provide only an approximation to the Swahili sounds, and depended on the linguistic knowledge of native speakers to interpret the text correctly (Omar and Drury, 2002, p14-15). As a further example, a copy of stanza 6 of the Utenzi wa Mwana Kupona (Mwana Kupona’s Ballad), as written by Muhammad Kijuma (c.1855–1945), is given below:

Figure 4.1: Stanza 6 from Muhammad Kijuma’s manuscript of Utenzi wa Mwana Kupona

The following is a letter-for-letter transcription of stanza 6, along with an automatically-generated close transliteration and a manually-added standard Roman transliteration and English translation:

\[
\begin{align*}
6b/a & \quad \text{mwana adamu si kitu, na ulimwengu si wetu} \\
6a/b & \quad \text{mankind is as nothing, and the world does not belong to us} \\
6c/d & \quad \text{walau hakuna mtu ambao atasaliya} \\
6d/c & \quad \text{and there is no person who will live forever}
\end{align*}
\]

However, if you are not dealing solely with older manuscripts, and you wish to use the spelling conventions proposed in Andika! in order to unambiguously represent current-day Swahili, and allow transliteration between Arabic script and the standard Swahili Roman script, then you will need the additional glyphs. If you see squares or boxes in the Arabic script, or just the glyph for the isolated form when initial, medial or final forms are required, the reason is that the font you are using is missing the glyphs that it would make it useable with Swahili.

The missing glyphs are likely to be one or more of those in Table 4.2. The first seven glyphs are the most important.

---

1For example, in the Ubuntu packages fonts-arabeyes or fonts-kacst, or on the web from the Open Font Library (openfontlibrary.org/en/search?query=Arabic).
2The examples are: نِغِمَاوَغُبِنْتِ (negemawangubinti), يِيِكِمُولَوَكُ (nyenyekeaMolawako), and مْتُهُنِنياَكِرَ*اَسِيَابَرَكِتَرَ (mtuhunenaakenda * asiyapanda kitanda).
3MS HS5, Asia Africa Institute/CSMC, University of Hamburg.
<table>
<thead>
<tr>
<th>Glyph</th>
<th>Unicode name</th>
<th>Unicode number</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>پ</td>
<td>peh</td>
<td>U+067E</td>
<td>p</td>
</tr>
<tr>
<td>ڤ</td>
<td>veh</td>
<td>U+06A4</td>
<td>v</td>
</tr>
<tr>
<td>چ</td>
<td>tcheh</td>
<td>U+0686</td>
<td>ch</td>
</tr>
<tr>
<td>ئ</td>
<td>ain with three dots above</td>
<td>U+06A0</td>
<td>g in ng'</td>
</tr>
<tr>
<td>ئ</td>
<td>ain with two dots above</td>
<td>U+075D</td>
<td>g</td>
</tr>
<tr>
<td>ّ</td>
<td>subscript alef</td>
<td>U+0656</td>
<td>short e</td>
</tr>
<tr>
<td>ٔ</td>
<td>inverted damma</td>
<td>U+0657</td>
<td>short o</td>
</tr>
<tr>
<td>ݝ</td>
<td>keheh with two dots above</td>
<td>U+063B</td>
<td>used by some writers for ch</td>
</tr>
<tr>
<td>ٹ</td>
<td>tteh</td>
<td>U+0679</td>
<td>alveolar t (Mombasa)</td>
</tr>
<tr>
<td>ڈ</td>
<td>ddal</td>
<td>U+0688</td>
<td>alveolar d (Mombasa)</td>
</tr>
<tr>
<td>ژ</td>
<td>waw with dot above</td>
<td>U+06CF</td>
<td>w (North)</td>
</tr>
<tr>
<td>ؤ</td>
<td>jeh</td>
<td>U+0698</td>
<td>zh (North)</td>
</tr>
</tbody>
</table>

Table 4.2: Glyphs commonly missing in fonts

At the time of writing, the only fonts which contain all of these additional glyphs in Table 4.2 are Scheherazade⁴ (by Bob Hallissy and Jonathan Kew), Amiri⁵ (by Khaled Hosny), and the fonts from the PakType project.⁶ Fonts containing all the glyphs in Table 4.2 apart from keheh are Droid Arabic Naskh⁷ and Droid Arabic Kufi⁸ (by Pascal Zoghbi), and Lateef.⁹

4.2 Default fonts in Andika!

When typing Swahili in Arabic script, the fonts can be changed directly in LibreOffice. For typesetting of existing manuscripts, Andika! uses four fonts as defaults when generating pdfs:

- Scheherazade for the Arabic transcription. A possible alternative here is Amiri.
- Linux Biolinum O¹⁰ for the close transcription into Roman script, since it is especially good at handling diacritics. A possible serif alternative here is Gentium¹¹ (by Victor Gaultney).
- Liberation Serif¹² for English translations.
- GranadaKD in andika/fonts for poem titles in Arabic. This is a Kufic-style font from Arabeyes¹³ that has been adapted by me to add the characters in Table 4.2 except keheh.

These default fonts can be changed by replacing the name of the font in the relevant command in convert/tex/fontdefs.tex. Thus, to change the transliteration font from Linux Biolinum O to Gentium, you would first install Gentium:

```
sudo apt-get install fonts-sil-gentium
```

and then open convert/tex/fontdefs.tex in a text-editor¹⁴ and change the line:

```
4 scripts.sil.org/cms/scripts/page.php?item_id=Scheherazade, available in the Ubuntu package fonts-sil-scheherazade, but see also Appendix A/7.
5 amirifont.org
6 paktype.sourceforge.net, available in the Ubuntu package fonts-paktype.
7 openfontlibrary.org/en/font/droid-arabic-naskh
8 openfontlibrary.org/en/font/droid-arabic-kufi
9 scripts.sil.org/cms/scripts/page.php?item_id=Lateef
10 linuxlibertine.org
11 scripts.sil.org/cms/scripts/page.php?site_id=nrsi&item_id=Gentium
12 fedorahosted.org/liberation-fonts
13 openfontlibrary.org/en/font/granada
14 A text-editor is an application specialising in the editing of text. Word-processors should never be used to edit files in Andika!, because they will quietly change the file in ways which will prevent it working. There are multiple text-editors such as Kate, Geany, and Gedit available in Ubuntu.
The \texttt{Color} command sets the colour of the font using the hex version of the RGB value\textsuperscript{15} (in this case, dark grey) – delete it if you want the transliteration in black. The \texttt{Scale} command alters the size of the font – if you want it a bit smaller than normal, enter (say) 0.8 instead of 1.

As a general point, the readability of diacritics (or even whether they are displayed at all) depends crucially on the font – not all will be capable of showing all diacritics, or of placing them in the right location, so if something is not looking right in your transliteration, try using Linux Biolinum O (sans-serif) or Gentium (serif) as suggested.

4.3 Adding missing glyphs to Arabic fonts

If you are anxious to use a particular Arabic font that does not have all the glyphs required by Swahili, it is possible to add them to the font using the font editor FontForge,\textsuperscript{16} originally developed by George Williams. Appendix B shows how to use FontForge to add missing glyphs,\textsuperscript{17} and a version with screenshots is also available at the website for the book \textit{Design with FontForge}.\textsuperscript{18}

You can also develop your own fonts using FontForge, though the creation of an attractive font is a highly specialised task requiring artistic flair as well as technical skill. The next version of the drawing program Inkscape,\textsuperscript{19} 0.49, will allow initial glyph designs to be created there and then imported into FontForge for finalisation.\textsuperscript{20}

4.4 Scheherazade and Amiri

The Scheherazade webpage\textsuperscript{21} notes that:

Scheherazade provides a “simplified” rendering of Arabic script, using basic connecting glyphs but not including a wide variety of additional ligatures or contextual alternates (only the required lam-alef ligatures). This simplified style is often preferred for clarity, especially in non-Arabic languages, but may not be considered appropriate in situations where a more elaborate style of calligraphy is preferred.

Scheherazade is the default in \textit{Andika}! because it fits the proposed full vocalisation better. For instance, Amiri places all the vowels at the same height from the main letter, eg \texttt{کُبٗرٖیشَ} (\textit{kuboresha}, to boost) compared to Scheherazade \texttt{کُبٗرٖیشَ}, and \texttt{وَنَسَیَانْسِی} (\textit{wanasayansi}, scientists) compared to Scheherazade \texttt{وَنَسَیَانْسِی}.

This can lead to the upper vowels from the current line of text colliding with the lower vowels from the previous line.

However, Amiri may be more appropriate for use with text that is not fully vocalised (eg quotation of Arabic within Swahili), particularly since it includes more of the ligatures commonly used in Arabic, making

\textsuperscript{15}colorspire.com/rgb-color-wheel
\textsuperscript{16}fontforge.github.io
\textsuperscript{17}Note that unless the font you are adapting is available under an open license, this may constitute a breach of copyright.
\textsuperscript{18}designwithfontforge.com/en-US/Adding_Glyphs_to_an_Arabic_Font.html
\textsuperscript{19}inkscape.org
\textsuperscript{20}understandingfonts.com/blog/2011/11/typography-extensions-in-inkscape-0-49. In the meantime, this functionality can be accessed by using the “bleeding edge” packages available from Inkscape Trunk – launchpad.net/~inkscape.dev/+archive/ubuntu/trunk.
\textsuperscript{21}scripts.sil.org/cms/scripts/page.php?item_id=Scheherazade
for more attractive text. For instance, Amiri (jalitumia, they used), compared to Scheherazade, has the letters ltm combined in one ligature.
Chapter 5

A keyboard layout for Swahili in Arabic script

5.1 Introduction

The keyboard layout proposed here is a work-in-progress, and can be adjusted in the light of experience – I would be happy to receive any suggestions for improvement. As well as describing the keyboard and explaining the conventions governing the layout, this chapter also includes information on how to edit the layout to suit individual needs.

The Andika! keyboard allows Swahili in Arabic script to be typed directly into a GNU/Linux computer using a standard English (UK or US) keyboard. Input speed is comparable to typing in Roman script. As well as allowing contemporary Swahili to be easily typed in Arabic script, the keyboard will enable most older manuscripts to be transliterated letter-for-letter.

The complete keyboard layout is depicted in Figure 5.1.

As can be seen from Figure 5.1, up to four glyphs may be accessed from one key. To access the contents of each key, the Shift and AltGr keys are used in combination where appropriate, as shown in Figure 5.2.

5.2 Governing principles for the layout

The basic governing principle behind the keyboard layout is that the relevant Arabic glyph will usually be produced by pressing the same key that produces the Roman glyph. It is thus very easy to use:

---

1I am grateful to Wikimedia for the original layout image.
just switch your keyboard to use Arabic script – in KDE, Ctrl + Alt + K (see Appendix A/8.1 for further information) – and start typing almost as if the keyboard is being used to type Roman script. Some examples are given in Table 5.1.²

<table>
<thead>
<tr>
<th>Arabic</th>
<th>Keystrokes</th>
<th>Roman</th>
<th>English</th>
</tr>
</thead>
<tbody>
<tr>
<td>مِيمِ</td>
<td>m, i, Shift + i, m, i</td>
<td>mimi</td>
<td>I, me</td>
</tr>
<tr>
<td>سسس</td>
<td>s, a, Shift + a, s, a</td>
<td>sasa</td>
<td>now</td>
</tr>
<tr>
<td>لكنك</td>
<td>l, a, k, i, Shift + i, n, i</td>
<td>lakini</td>
<td>but</td>
</tr>
<tr>
<td>نفيفك</td>
<td>n, i, m, e, f, i, Shift + i, k, a</td>
<td>nimefika</td>
<td>I have arrived</td>
</tr>
</tbody>
</table>

Table 5.1: Typing examples

The other main principle behind the layout is the consistent placement of glyphs that are related by shape or sound in either script:

- The digraphs dh gh th sh zh are on the same keys as d g t s z, and are accessed using the Shift key.
- The pharyngeal consonants ص ض ط ظ are on the same keys as z t d s, and are accessed using the AltGr key.
- Similar Arabic glyph shapes are placed on the same key where possible – for instance يى are on the y key, and زژ are on the w key.
- Long and short vowels are located on the same key, with the long vowel accessed by Shift, so for instance the u key produces ة and Shift + u produces  UIButton.
- The vowel carriers أإئؤ are all accessed using the AltGr key.
- The alveolar consonants ٹڈ used in Mombasa Swahili are accessed using the AltGr + Shift keys.
- The glyphs وي are repeated on w y for use when they represent semi-vowels.
- The palatal digraph ch is accessed using the c key, and an alternate representation used by some writers, چک, is accessed using Shift + c.
- The occasionally-used digraph kh is accessed using the X key.
- Non-alphabetic characters from the UK keyboard are currently available via AltGr and AltGr + Shift, in case they might be of use.

Further information on the glyphs accessible from each key is available in Table 6.1 (consonants) and Table 6.2 (vowels).

5.3 Changing the layout

The layout of the keyboard is specified in the file layout/tz. Once copied to the appropriate place (see Appendix A/8), the layout is available for use. The file (reproduced in Appendix E) is a simple text file, and can be easily adapted to add new glyphs or change the position of existing glyphs – see Appendix C for instructions on doing this.

²For an explanation of the penultimate long vowels accessed by the Shift keys, see Chapter 6.
Chapter 6

Writing contemporary Swahili in Arabic script

6.1 Introduction

The spelling conventions suggested here for writing contemporary Swahili in Arabic script are based on those developed by Sheikh Yahya Ali Omar, as evidenced in his own manuscripts and in Omar and Frankl (1997). However, I am wholly responsible for the conventions set out here, and for any unwitting misinterpretation! In particular, the issue of vowel sequences (Section 6.5 below) is a complex one, and may need revision based on input from first-language speakers who are literate in Swahili in Arabic script. I would be happy to hear from anyone who has any comments on the conventions.

6.2 General principles

Word segmentation is as for standard Swahili in Roman script. This means that items such as لَزَيَنَ na, ya, za, la are written separately from the following word, even though in older manuscripts they may be written attached to that word.

All short vowels are marked. Although short vowels are usually omitted in Arabic, this is inadvisable in Swahili because of the different structure of the language, and also because Swahili has five vowels instead of three.

The penultimate syllable of a word has its stress marked by writing it with a long vowel. لُزَيَنَ is used for a, لُزَيَنَ for e and i, and لُزَيَنَ for o and u.3 This also helps to delimit individual words in the Arabic script.

Initial vowels use the vowel-carriers أ (AltGr+A, for a, o, u) or إ (AltGr+\, for e, i), eg أَنَسٖيمَ (anasema, he is speaking), إِذِينِ (ugali, porridge), إِذِينِ (idhini, permission).4 The order of typing is: vowel carrier, then short vowel, then long vowel (if applicable).

Arabic sounds in loanwords should ideally use the original Arabic glyph, but they can also be written as an Arabic transliteration of the Roman letter, eg ذ instead of ص.4

6.3 Representation of consonants

The representation of Swahili vowels in Arabic script is set out in Table 6.1.

<table>
<thead>
<tr>
<th>Roman</th>
<th>Arabic</th>
<th>Keystrokes</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
<td>ب</td>
<td>كِبْرُ</td>
<td>kiburi (arrogance)</td>
</tr>
</tbody>
</table>

1I have tried to build on the discussion in Omar and Frankl (1997): Appendix B: The Hamza in Swahili Arabic script.
2The short vowels a, i, u may be omitted when they occur before a long vowel, eg سَاسَ (sasa, now), but this is not recommended.
3Omar and Frankl (1997, p69) recommends omission of the hamza, presumably in order to limit the number of diacritics in the text, but the current convention in Andika! is to write it.
4Note that the Roman to Arabic converter will always do this, since standard Swahili in Roman script does not preserve these distinctions.
<table>
<thead>
<tr>
<th>Arabic Letter</th>
<th>Romanization</th>
<th>English Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>ch</td>
<td>ج c</td>
<td>чungwa (large orange)</td>
</tr>
<tr>
<td>ch (aspirated, Mombasa)</td>
<td>چ c, h</td>
<td>ч’ungwa (medium-sized orange)</td>
</tr>
<tr>
<td>d</td>
<td>د d</td>
<td>kudangany (to deceive)</td>
</tr>
<tr>
<td>d - alveolar d (Mombasa)</td>
<td>د AltGr + Shift + d</td>
<td>тundu (chicken coop)</td>
</tr>
<tr>
<td>dh</td>
<td>ذ Shift + d</td>
<td>дhababu (gold)</td>
</tr>
<tr>
<td>dh (pharyngeal)</td>
<td>ض AltGr + d</td>
<td>дhiki (distress)</td>
</tr>
<tr>
<td>dh (pharyngeal)</td>
<td>ظ AltGr + z</td>
<td>дhuhuri (noon)</td>
</tr>
<tr>
<td>f</td>
<td>ف f</td>
<td>figo (kidneys)</td>
</tr>
<tr>
<td>g</td>
<td>غ g</td>
<td>гуния (sack)</td>
</tr>
<tr>
<td>gh</td>
<td>غ h</td>
<td>гhadhabu (anger)</td>
</tr>
<tr>
<td>h</td>
<td>ه h</td>
<td>hako (he is not here)</td>
</tr>
<tr>
<td>h (pharyngeal)</td>
<td>ح Shift + h</td>
<td>hаsan (Hasan [name])</td>
</tr>
<tr>
<td>[k]h</td>
<td>خ x</td>
<td>[k] habari (news)</td>
</tr>
<tr>
<td>j</td>
<td>ج j</td>
<td>jан (yesterday)</td>
</tr>
<tr>
<td>k</td>
<td>ك k</td>
<td>kuku (large hen)</td>
</tr>
<tr>
<td>k (aspirated, Mombasa)</td>
<td>ك k, h</td>
<td>k’uku (medium-sized hen)</td>
</tr>
<tr>
<td>l</td>
<td>ل l</td>
<td>Kulima (to dig)</td>
</tr>
<tr>
<td>m</td>
<td>م m</td>
<td>mimi (I)</td>
</tr>
<tr>
<td>n</td>
<td>ن n</td>
<td>nani (who?)</td>
</tr>
<tr>
<td>ng’</td>
<td>نغ n, Shift + n</td>
<td>ng’ombe (cattle)</td>
</tr>
<tr>
<td>p</td>
<td>ب p</td>
<td>kupaka (to paint)</td>
</tr>
<tr>
<td>q</td>
<td>ق q</td>
<td>qафu (consecrated)</td>
</tr>
<tr>
<td>r</td>
<td>ر r</td>
<td>kurudi (to come back)</td>
</tr>
<tr>
<td>s</td>
<td>س s</td>
<td>kusimama (to stand)</td>
</tr>
<tr>
<td>s (pharyngeal)</td>
<td>ص AltGr + s</td>
<td>sahibu (friend)</td>
</tr>
<tr>
<td>sh</td>
<td>ش Shift + s</td>
<td>kushika (to hold)</td>
</tr>
<tr>
<td>t</td>
<td>ت t</td>
<td>fitina (intrigue)</td>
</tr>
<tr>
<td>t (aspirated dental, Mombasa)</td>
<td>ت t, h</td>
<td>t’upa (bottle)</td>
</tr>
<tr>
<td>t (alveolar, Mombasa)</td>
<td>ت AltGr + Shift + t</td>
<td>tundu (chicken coop)</td>
</tr>
<tr>
<td>t (pharyngeal)</td>
<td>ط t</td>
<td>kutahirisha (to purify)</td>
</tr>
<tr>
<td>th</td>
<td>ث Shift + t</td>
<td>thamání (eighty)</td>
</tr>
<tr>
<td>v</td>
<td>ف v</td>
<td>kuvimba (to swell)</td>
</tr>
<tr>
<td>z</td>
<td>ظ z</td>
<td>kuzima (to extinguish)</td>
</tr>
<tr>
<td>zh (Northern)</td>
<td>ظ Shift + z</td>
<td>zhina (name)</td>
</tr>
</tbody>
</table>
CHAPTER 6. WRITING CONTEMPORARY SWAHILI IN ARABIC SCRIPT

6.4 Representation of vowels

The representation of Swahili vowels in Arabic script is set out in Table 6.2.

<table>
<thead>
<tr>
<th>Roman</th>
<th>Arabic</th>
<th>Keystrokes</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>a-...</td>
<td>ا</td>
<td>AltGr + a, a</td>
<td>asoma (he reads)</td>
</tr>
<tr>
<td>...a-...</td>
<td>َا</td>
<td>a, Shift + a</td>
<td>baharini (in the sea)</td>
</tr>
<tr>
<td>...aCV</td>
<td>َء</td>
<td>a, Shift + a</td>
<td>sasa (now)</td>
</tr>
<tr>
<td>...aV</td>
<td>َاء</td>
<td>a, Shift + a, Shift + h</td>
<td>mafaa (usefulness)</td>
</tr>
<tr>
<td>...e-...</td>
<td>َى</td>
<td>e, Shift + e</td>
<td>njema (good)</td>
</tr>
<tr>
<td>...eV</td>
<td>َى</td>
<td>e, AltGr + e</td>
<td>kupea (to sweep)</td>
</tr>
<tr>
<td>...i-...</td>
<td>َى</td>
<td>i, Shift + i</td>
<td>mashizi (soot)</td>
</tr>
<tr>
<td>...iCV</td>
<td>َى</td>
<td>i, Shift + i</td>
<td>isipokuwa (unless)</td>
</tr>
<tr>
<td>...i</td>
<td>َى</td>
<td>i, Shift + i</td>
<td>kitabu (book)</td>
</tr>
</tbody>
</table>

NOTE: In the Keystrokes column, the comma stands for followed by.
CHAPTER 6. WRITING CONTEMPORARY SWAHILI IN ARABIC SCRIPT

...IV  i, AltGr+i  كُتِئَ  kutia (to place)

<table>
<thead>
<tr>
<th>Vowel</th>
<th>Carrier</th>
<th>Arabic</th>
<th>Keystrokes</th>
</tr>
</thead>
<tbody>
<tr>
<td>o...</td>
<td>AltGr+a, o</td>
<td>أكتوب (Oktober)</td>
<td></td>
</tr>
<tr>
<td>...o...</td>
<td>o</td>
<td>كليلم (cultivation)</td>
<td></td>
</tr>
<tr>
<td>...oCV</td>
<td>o, Shift+o</td>
<td>مكُونڠَ (elephant’s trunk)</td>
<td></td>
</tr>
<tr>
<td>...oV</td>
<td>o, AltGr+o</td>
<td>كُوپٖئَ (to cool)</td>
<td></td>
</tr>
<tr>
<td>u...</td>
<td>AltGr+a, u</td>
<td>عليمي (tongue)</td>
<td></td>
</tr>
<tr>
<td>...u...</td>
<td>u</td>
<td>كُشعُرٖ (to give thanks)</td>
<td></td>
</tr>
<tr>
<td>...uCV</td>
<td>u, Shift+u</td>
<td>كُومٖ (ten)</td>
<td></td>
</tr>
<tr>
<td>...uV</td>
<td>u, AltGr+u</td>
<td>كُسُڠُ (to rub)</td>
<td></td>
</tr>
</tbody>
</table>

NOTE: In the Roman column, C stands for consonant or consonant cluster and V for vowel, and the entries refer respectively to (1) initial, (2) non-initial and non-penultimate, (3) penultimate followed by a consonant, (4) penultimate followed by a vowel. For a discussion of vowel-sequences, see Section 6.5. In the Keystrokes column, the comma stands for followed by.

Table 6.2: Representation of single vowels

6.5 Vowel sequences

Vowel sequences have matching vowel-carriers inserted between them, as set out in Table 6.3.

Table 6.3: Vowel-carriers

<table>
<thead>
<tr>
<th>Vowel</th>
<th>Carrier</th>
<th>Arabic</th>
<th>Keystrokes</th>
</tr>
</thead>
<tbody>
<tr>
<td>e, i</td>
<td>yeh + hamza</td>
<td>كُنِ  AltGr+ Shift+I or E or Y</td>
<td></td>
</tr>
<tr>
<td>o, u</td>
<td>waw + hamza</td>
<td>كُؤِ  AltGr+ Shift+O or U or W</td>
<td></td>
</tr>
<tr>
<td>a</td>
<td>alef + hamza</td>
<td>كُؤِ  Shift + A, AltGr + Shift + H</td>
<td></td>
</tr>
</tbody>
</table>

6.5.1 Stressed + unstressed vowel sequences

When the vowel is first in the vowel sequence and is also stressed (which will only happen when it is in penultimate position in the word), a vowel-carrier is inserted after it. Where a is concerned, the hamza on the carrier is written as a full letter rather than a diacritic.

kupea (to sweep)  kupe•a  كُپٖئَ
kupokea (to receive)  kupoke•a  كُپٖكٖئَ
kutia (to place)  kuti•a  كُتِئَ
kupoa (to cool)  kupoa•a  كُوپٖئَ
Since the vowel-carrier accompanies the stress, there is no need to add another long vowel to mark the stress. Thus كُسُعظَ (kupia, to sweep), and not كُسُعظَ, and كُتٖؤَ (kutoa, to produce), and not كُتٖؤَ.

6.5.2 Unstressed + stressed vowel sequences

When the vowel e, i, o, u is second in the vowel sequence and is also stressed (i.e. again appearing in penultimate position), it has a matching vowel-carrier as in Table 6.3 inserted before it. Since the vowel-carrier comes before the stress, the stressed vowel is marked as normal with a long vowel.

shairi (poetry) → shajiri → شَجِيرٌ
kiini (pith) → kiini → كِينٌ
kuita (to call) → kujita → كُجِيِتَ
shauri (advice) → shauri → شَورِ
meupe (white [class 6]) → meupe → مَوْعِ
kuona (to see) → kuona → كُوُنَ

However, where the second (stressed) vowel of the sequence is a, the vowel-carrier matches the preceding vowel unless that preceding vowel is itself a, in which case the hamza on the carrier is written as a diacritic rather than as a full letter.

viazi (potatoes) → viazi → فياز
akaacha (then he left behind) → akaacha → أَكاَاچَ

6.5.3 Unstressed vowel sequences

In vowel sequences where there is no stress (i.e. none of the vowels in the sequence appear in penultimate position), the vowel-carrier matches the first vowel. Again, in the case of a, the hamza on the carrier is written as a diacritic rather than as a full letter.

tuondoke (let us leave) → tuondoke → تُؤُندُوكَ
kuanika (to write) → kuanika → كُونِيدِكَ
maandishi (manuscripts) → maandishi → مَانِدِيشِ

6.5.4 Longer vowel sequences

Longer sequences are handled in line with the principles above.

kuua → kuua → كُؤُؤَ
6.6 Comparing conventions

Table 6.4 summarises the differences between the writing systems used in Sheikh Yahya’s manuscripts, Omar and Frankl (1997), and Andika!

<table>
<thead>
<tr>
<th>Feature</th>
<th>Manuscripts</th>
<th>Article</th>
<th>Andika!</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sakani is marked on long vowels</td>
<td>✓</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>All short vowels are marked</td>
<td>✓</td>
<td>x</td>
<td>✓</td>
</tr>
<tr>
<td>Sakani on consonants denotes syllabicity only</td>
<td>x</td>
<td>✓</td>
<td>x</td>
</tr>
<tr>
<td>Distinction between syllabicity and prenasalisation</td>
<td>✓</td>
<td>✓</td>
<td>x</td>
</tr>
</tbody>
</table>

6.6.1 Sakani on long vowels

In Sheikh Yahya’s manuscripts, ي و ء carry a sakani when used to mark length/stress in the penultimate syllable, eg مزئو (maziwa, milk). However, in Omar and Frankl (1997), sakani is not used here (eg مزئو). The suggested spelling in Andika! reflects this (though users can of course mark sakani if they wish).

6.6.2 Marking short vowels

In Sheikh Yahya’s manuscripts, all short vowels are marked, and Andika! follows this. However, Omar and Frankl (1997) proposed that marking these is unnecessary in certain situations:

- If the short (unstressed, non-penultimate) vowel they represent is identical to a preceding short vowel. For example, in تاميين (thamanini, eighty) the second a is omitted because it is preceded by an a (fataha).
- If the short vowel they represent is identical to a preceding or following stressed (penultimate) vowel represented by ي و ء. For example, in تاميين (thamanini, eighty) the last i (kasiri) is omitted because it is preceded by ي, and in ذهاب (dhahabu, gold) the first a (fataha) is omitted because it is followed by ء.
- Where all the vowels in a word are identical, except for stress. For example: تپاكاز (tapakaza, scatter), فکیر (fikiri, think), شکور (shukuru, give thanks).

However, the suggested spelling convention in Andika!, as in Sheikh Yahya’s own manuscripts, is that all short vowels are marked, thus: ﯽ ﯽ ﯽ. There are a few practical reasons for this:

- Short e, o need to be marked anyway, since Arabic script has no way otherwise of distinguishing ي meaning i from و meaning e, or و meaning o from و meaning u.
- Omitting short vowels may conceivably save time when writing, once the rules above are mastered, but this is unlikely to apply when typing – it is probably faster simply to type more or less what would be typed when using Roman script, including short vowels.
- The omission of short vowels means that transliteration into Roman script would require post-editing to add vowels. It might be possible to automate the application of the above rules to avoid this, but the resulting system would likely be cumbersome, and simply typing the short vowels is a more practical solution.

6.6.3 Sakani on consonants

Arabic sukun marks the absence of a vowel after a consonant. In Sheikh Yahya’s manuscripts, sakani is used consistently for this purpose (alongside its use on long vowels). Thus: أُنَڤو (unavyeweza, how
you can), َكْوَ (kwa, to, by, for). Its most common occurrence is on a nasal before another consonant: ْنَجِيٰمَ (njema, good).

Its use on nasals means that sakani can also denote syllabicity, and in Omar and Frankl (1997) its function appears to be limited solely to that. The aim, as with the omission of short vowels, was most likely to limit the number of diacritics in the text.

The suggested convention in Andika! currently is to follow the manuscript practice, and use sakani on the first consonant of multi-consonant clusters. However, since sakani is not strictly necessary if all vowels are being marked, this convention is open to change. (If users feel that marking sakani leads to clutter, they can of course omit it).

### 6.6.4 Distinction between syllabicity and prenasalisation

Although the Roman orthography does not distinguish these two sounds, both Sheikh Yahya’s manuscripts and Omar and Frankl (1997) make a distinction between a syllabic nasal followed by a voiced plosive (eg mb) and a prenasalised voiced plosive (eg nb). The former is written with a preceding ْمَ, and the latter with a preceding ْنَ, as in ْمُبَيٰ (mbaya, bad [Class1]) compared to ْنَايٰ (mbaya, bad [Class9]).

Andika! will of course allow this distinction to be made in the Arabic script should a writer wish to do so. However, the Roman to Arabic converter cannot do this (since the distinction is not reflected in the standard orthography), and will always convert mb to ْمَبَيٰ, so automatically-converted text will need post-editing to reflect this distinction if the user wishes to make it.
Chapter 7

Converting from one script to the other

7.1 Introduction

Andika! includes a number of options to convert between Arabic and Roman scripts. Because Andika! is a work in progress, it is a good idea to check the output before re-using it in other contexts, since it may require some manual editing – for instance, Arabic script does not have capital letters, so capitals (other than most sentence-intial capitals) need to be added by hand to Roman output.

7.2 Cut-and-paste converters

The simplest option is to use the cut-and-paste converters on the website.\(^1\) If you have followed the instructions in Appendix A/16 these will also be available on your own machine.\(^2\)

To use these converters, type or paste text into the input box. Input is truncated to 900 characters, but if your text is longer than this you can convert it in chunks. The truncation limit can be changed by editing or commenting out the line:

\[
\text{$\text{mystring}=\text{strip\_tags}($\text{mystring, 0, 900})}\]

in \text{convert\_rom\_ar.php} and:

\[
\text{$\text{input}=\text{strip\_tags}($\text{input, 0, 900})}\]

in \text{convert\_ar\_rom.php}.

If you have large amounts of text to convert, the command-line converter should be used – see Section 7.3 below.

7.2.1 Arabic to Roman

The Arabic to Roman converter transliterates Arabic script into standard Roman orthography. The correspondence should be perfect if the input text follows the spelling conventions for Arabic script (Chapter 6).\(^3\) Where this is not the case (eg with text copied from manuscripts), the converter transliterates the Arabic text as best it can.

If you have installed the webpages locally, you can replace the standard Roman transliteration with a close transliteration containing diacritics by replacing $\text{standard}$ with $\text{close}$ at the end of \text{convert\_ar\_rom.php}.

Note that when converting from Arabic to Roman script, Firefox's spellchecker will underline every word in the Arabic script entry area. To avoid this, turn off as-you-type spellchecking: click on the \text{Open Menu} button, select \text{Preferences $\rightarrow$ Advanced}, and on the \text{General} tab, untick \text{Check my spelling as I type} in the \text{Browsing} section.

\(^{1}\)kevindonnelly.org.uk/swahili/rom_ar.php and kevindonnelly.org.uk/swahili/ar_rom.php.

\(^{2}\)andika/rom_ar.php and andika/ar_rom.php.

\(^{3}\)If you take the Roman output and paste it into the Roman to Arabic converter, you should get your Arabic input back as the output from that.
7.2.2 Roman to Arabic

The Roman to Arabic converter transliterates standard Roman orthography into Arabic script.

The default is to show sakani on a consonant where it does not have an accompanying vowel (e.g., kwa, kuboreshwa, sayansi). This can be changed by ticking Do not show sakani (sukun) on consonants - then no sakani will be shown.

The default is to show numerals in Western-Arabic form (1234567890). This can be changed by ticking Convert numerals to Arabic-Indic forms - then numerals will be shown as ١٢٣٤٥٦٧٨٩٠.

Some writers use sakani on و and ي when used as long vowels in the penultimate syllable. The default is not to show this, but this can be changed by ticking Show sakani (sukun) on و and ي as long vowels.

Andika! is not a translator - non-Swahili words are simply transliterated letter-for-letter from Roman script into Arabic script. English c is transliterated as ڮ, and x as كْس. Examples: Shrewsbury, Creative Commons License. A sakani is used where it would occur in Swahili (depending on the settings above), but is not applied elsewhere.

Andika! is not a spelling or punctuation corrector - any errors in the text entered will be carried over into the transliteration. The conversion may contain lines with out-of-sequence words if the source contains a mixture of Swahili and another language with letters that do not occur in the standard Swahili Roman orthography (the Swahili will be converted to RTL Arabic script, but the non-Swahili letters will be passed through as LTR Roman script). The transliteration equivalents chosen here mean that line continuity is not a problem where the “other language” is English. However, be aware that problems may occur if the “other language” is French, German, or something else.

Note that the converter will always use the "commonest" Arabic letter. For instance, it will convert dh to ذ instead ofكو or ض, which might be the original Arabic letter in the word. There is no way around this, since the standard Swahili Roman orthography does not preserve these distinctions, and the only option in such cases is to edit the output afterwards.

7.2.3 Convert a webpage

The website also includes a tool to transliterate entire webpages from Roman script to Arabic script. Although it should work on most webpages, most testing has been done on Wikipedia pages.

To use the tool, simply enter the webpage address in the box – the initial http:// can be omitted if desired. Only a subset of characters are allowed in the web address: alphanumeric characters (a-z, 0-9), full-stops (.), hyphens (-), underscores (_), single quotes (‘), colons (:) and slashes (/). Non-existent web-addresses will produce a blank conversion page.

While there should be no problems transliterating the main text of the webpage, some peripheral "page furniture" (e.g., menus, lists of links, etc) may not be transliterated properly. All links on the converted page will go to unconverted (Roman script) pages.

7.3 Command-line converter

Cutting and pasting does not make sense for long documents. Andika! therefore includes a converter which will act directly on the document, provided it is laid out in a particular way – see Section 7.3.3. The document can be in either Arabic or Roman script, in odt (libreOffice Writer) or txt (plain text) format, and can be converted to pdf, odt or txt format, in three possible layouts, with or without Roman transliteration.

The converter can be used in two modes: via a point-and-click interface (Section 7.3.1), or via a command typed directly into a terminal (Section 7.3.2). The latter option also makes it possible to automate the
use of the converter if you have a number of documents that need conversion.

The converter also offers the option of importing the text of the document into a database table. This is the option recommended for any serious editorial work, and is dealt with in detail in Chapter 8.

It is recommended that files to be converted are stored in `andika/convert/inputs` – they can each be put in their own folder beneath that if desired. The converted documents, along with related files, will be stored in `andika/convert/outputs` in a folder named after the document. Thus, converting a document called `mkunumbi.odt` to PDF format will result in a file `mkunumbi.pdf` in the folder `andika/convert/output/mkunumbi`. Note that each invocation of the converter will create output that overwrites the previous output, so if you want to keep multiple layouts of a particular converted document, you need to save the output separately.

It is a good idea to keep the input filename lower-case and all-one-word. In contrast to Microsoft Windows, Ubuntu will consider files with capitalised names as different files from the lower-case equivalent, and filenames containing spaces may not be handled as anticipated. If you need to include multiple words in the filename, link them with an underscore.

In PDF output, lines can sometimes appear ill-aligned when you change the desired layout. This is due to LaTeX having to compile the PDF again to apply the new layout. It can be fixed by simply repeating the import.

With an ODT input file, if you get an error message similar to the following:

```
Warning: array_combine() expects parameter 2 to be array, null given in /srv/www/andika/convert/convert.php on line 163
Warning: Invalid argument supplied for foreach() in /srv/www/andika/convert/convert.php on line 166
```

it means that you have two blank lines at the end of the file instead of one.

### 7.3.1 Point-and-click interface

To start the converter in this mode, open a terminal and enter:

```
convert/convert.sh
```

A series of windows will open, allowing you to make the following choices:

1. The document (file) to be converted. For poetry, the document needs to be in a specific layout – see Section 7.3.3 below.
2. The script in which the document is written (Arabic or Roman).
3. The genre of the document (poetry or prose).
4. The type of output required (PDF, ODT, TXT, or insertion into a database table). If database insertion is chosen, no further selections need be made.
5. For poetry, the layout required (two `vipande` per line, separated by space; two `vipande` per line, separated by asterisk; or one `kipande` per line). Only the latter two layouts are available for ODT and TXT poetry output. For prose, there is only one layout (the line in Arabic script followed by the line in Roman script), and this will be applied automatically.
6. Whether or not the Roman script (whether original text or generated transliteration) should be included in the output. The default is to include it.

Note that in the case of two `vipande` separated by a space, the transcription below them applies to each `kipande` separately; in the case of two `vipande` separated by an asterisk, the transcription below them applies to the whole line of two `vipande`. The stanza numbering reflects this. The two versions of a quatrain from the *Utenzi wa Jaafari* (*Ballad of Jaafari*) in Figure 7.1 and Figure 7.2 show the differences.
7.3.2 Command-line input

In this mode, the converter is launched by running:

```
php convert/convert.php
```

followed by the options in the order above (filename, script, genre, output, layout, transliteration) separated by +. For instance, the command:

```
php convert/convert.php convert/inputs/machozi.txt+roman+poetry+pdf+vip-star+TRUE
```

will convert the document `machozi.txt`, a poem in Roman script (Knappert, 1972, p.163), into a pdf with the layout of two vipande to the line, separated by an asterisk, and including the transliteration (in this case, the original standard Roman orthography).

All options are entered in lower-case, though “Prose/Poetry” and “Arabic/Roman” can use capitals if desired.

Part of the output from the point-and-click interface will be the above list of + - separated options, so that they can be cut-and-pasted for future use. This is useful if you are repeatedly converting one document, since you can re-run the command once (using up-arrow and then Return) rather than have to click through the screens of the interface.

7.3.3 Layout of the input document

The documents to be converted can be in either txt or odt format.

Poems in traditional metres should be laid out one kipande to the line, with a blank line between stanzas and one (odt) or two (txt) blank lines at the end.

An alternative format for longer poems (for example, tenzi) is available, which may be useful in helping to pinpoint your location in the transcription. After the blank stanza-separation line, each stanza can be preceded by a line giving the number of the stanza as written in the manuscript. The format for this is hash (#) followed directly (i.e. with no intervening space) by the manuscript number – an example
of this format is available in the file andika/convert/inputs/jaafari/jnum.odt. The hash symbol and Roman numerals are accessible from the Swahili keyboard by holding down AltGr and pressing the hash and numeral keys. Note that any stanza numbers added in this hash line will be ignored by the conversion to pdf, odt or txt in favour of its own stanza numbering (the scribe or the copyist may misnumber stanzas, so it is better to let the computer do it). However, when converting to a database table, these numbers will be imported into the table (in the msno field) alongside the stanza number calculated by the computer (in the stanza field) – see Section 8.2 for more information.

Prose will be converted by paragraph. Again, the document should end with one (odt) or two (txt) blank lines, but there should be no blank line between paragraphs. Where a manuscript has been transcribed line by line, each line in the transcribed document will therefore equate to a paragraph, and be transliterated separately to give an interlinear transliteration.

Poems in free metres, or prose poems, should be treated as prose.

7.3.4 Converting pdfs

Documents in pdf format can also be converted, but need first to be converted to text. To do this, ensure that the package poppler-utils is installed:

```
sudo apt-get install poppler-utils
```

Then move into the directory where the pdf is located, and run:

```
pdftotext file_to_be_converted.pdf
```

replacing file_to_be_converted with whatever the filename is. This will produce an equivalently-named txt file.

This step is not included in the converter because typically the resulting text file will require some editing before it is fit for use – in particular, headers and footers on the pdf page will usually interrupt the run of the text, and need to be removed.

Once the text file has been tidied, it can then be converted as explained above.

7.4 General notes on usage

7.4.1 Adjusting the import's stanza numbering

By default, the conversion process numbers stanzas from 1 onwards. However, if you are handling an excerpt and want the numbering to reflect the actual stanza numbers of the excerpt, it is easy to do this. Open the file in a text editor (not in a word-processor):

```
kate convert/convert.php
```

and locate the line:

```
$stanza_no=0;
```

Edit the number so that it is set to one below the lowest stanza number of the excerpt – for instance, for an excerpt beginning at stanza 138 you would edit the line to read:

```
$stanza_no=137;
```

Save the file and run the conversion.

Remember to reset the line to 0 when you have finished.

---

\[\text{If you are importing into a database, you can instead import the excerpt without editing convert/convert.php and adjust the stanza numbering directly in the poem table - see Section 8.2.3.}\]
7.4.2 Warning when using the convert tools

If you get a series of warnings like the following when using the convert tools:

Warning: array Combine(): Both parameters should have an equal number of elements in /var/www/andika/convert/convert.php on line 175
Warning: end() expects parameter 1 to be array, boolean given in /var/www/andika/convert/convert.php on line 178
Warning: key() expects parameter 1 to be array, boolean given in /var/www/andika/convert/convert.php on line 179
Warning: reset() expects parameter 1 to be array, boolean given in /var/www/andika/convert/convert.php on line 180
Warning: Invalid argument supplied for foreach() in /var/www/andika/convert/convert.php on line 184

it means that you have mistakes in your input file (eg you have typed only 5 vipande in a stanza when all the others have 6).

7.5 How the conversion works

This section gives an overview of how each script is converted into the other. It is not necessary to be aware of this in order to use the conversion tools. More details can be gleaned from inspecting the file andika/includes/fns.php, where the functions (self-contained pieces of code) which handle the conversion are located. The procedure used here could no doubt be further developed, and I would be happy to take suggestions for improvements.

The conversion procedure is built on applying a series of regular expressions (regexes)\(^5\) to the text, giving a versatile framework where the output can be fine-tuned to suit an individual user's requirements.

7.5.1 Arabic to Roman

Arabic to Roman conversion is handled in two stages. The first produces an intermediate “Romanised” conversion, and then the second “smoothes” the initial conversion to produce standard Roman orthography, and/or a close transcription. The benefit of using two stages is that different smoothers can be developed for different transliteration systems.

First, the function ar2rom() converts each Arabic glyph to a Roman or Unicode equivalent (the latter when the equivalent English glyph might be represented with a diacritic).

Thus سَاسَ (sasa, now) will have the following regexes applied to it:

\[
\text{text=preg_replace("/\x{0633}/u", "s", $text);} \\
to convert \text{seen (U+0633) to s}
\]

\[
\text{text=preg_replace("/\x{064E}/u", "a", $text);} \\
to convert \text{fatha (U+064E) to a}
\]

\[
\text{text=preg_replace("/\x{0627}/u", "L", $text);} \\
to convert \text{alef (U+0627) to L}
\]

The output from the function for this word will therefore be salsa.

L is used to designate alef (long a), w to designate waw (long u), and y to designate yeh (long i). Thus رَفِيكِ (rafiki, friend) will be converted to rafiyki, and كُومِ (kumi, ten) to kuwmi.

\(^5\)en.wikipedia.org/wiki/Regular_expression
In cases where the Arabic glyph might be represented in Roman with a diacritic or by more than one
glyph, the Unicode designator is carried across. Thus, in خبار ([k]habari, news), the initial kh is handled by the
regex:

\$text=preg_replace("/x{062E}/u", "U+062E", $text);

and the output after this first stage would be U+062EabaLri.

In the second stage, the functions standardise() or close() are applied to the output of the first stage.

standardise() will apply regexes such as the following to the words above:

\$text=preg_replace("/U+062E/", "h", $text);
to convert U+062E to h: U+062EabaLri → habaLri

\$text=preg_replace("/a?L/", "a", $text);
to delete L preceded by a: habaLri → habari, saLsa → sasa

\$text=preg_replace("/iy([^aeiou])/", "$1$2", $text);
to delete y preceded by e or i and followed by a non-vowel: rafiyki → rafiiki

\$text=preg_replace("/uw([^bcdfghjklmnpqrstvwyz'U])/", "ū$1", $text);
to convert uw followed by a consonant, semi-vowel or Unicode codepoint to ā: kuwmi → kūmi.

It then applies a (very simplistic!) regex to capitalise words occurring after a full-stop, since Arabic has no
capitalisation.

close() will apply slightly different regexes:

\$text=preg_replace("/U\+062E/", "h\U0331", $text);
to convert U+062E to ħ: U+062EabaLri → ḡabaLri

\$text=preg_replace("/a?L/", "ā", $text);
to convert L or al to ā: ḡabaLri → ḡabāri, saLsa → sāsa

\$text=preg_replace("/iy([^bcdfghjklmnpqrstvwyz'U])/", "U+0131U+0304$1", $text)
to convert iy followed by a consonant, semi-vowel or Unicode codepoint to Ĩ: rafiyki → rafiiki (the two
Unicode codepoints designate a dotless i and a macron)

\$text=preg_replace("/uw([^bcdfghjklmnpqrstvwyz'U])/", "ū$1", $text);
to convert uw followed by a consonant, semi-vowel or Unicode codepoint to ā: kuwmi → kūmi.

7.5.2 Roman to Arabic

Conversion from standard Roman orthography (or a close transcription) to the suggested standard Arabic
orthography is mainly done with one function, rom2ar(), but there are a number of helper functions that
tweak the output.

The first helper function, prep_rom(), adds a long vowel in the penultimate syllable, converting sasa
(now) into saLsa. Then the function rom2ar() converts each Roman glyph (including glyphs with dia-
critics in a close transcription) to the Unicode designator for the Arabic glyph.

Thus saLsa will have the following regexes applied:

\$text=preg_replace("/s/", "U+0633", $text);
to convert s to seen

\$text=preg_replace("/a/", "U+064E", $text);
to convert a to fatha

\$text=preg_replace("/L/", "U+0627", $text);
to convert \( L \) to alef

The output from the function for this word will therefore be سَاسَ.

The \( \ Helvetica \) in the close transcription \( \ Helvetica \) (news) will have it converted to \( \ Helvetica \) by the following regex:

```php
$text=preg_replace("/\ Helvetica/", "U+062E", $text);
```
giving خَبَارِ, but the \( h \) in the standard spelling, \( \ Helvetica \), will be converted to heh by the regex:

```php
$text=preg_replace("/\ Helvetica/", "U+0647", $text);
```
giving هَبَارِ.

In most cases of standard \( h \), heh will be the correct choice, but in the case of this particular word \( \ Helvetica \) is probably better, since it is the glyph used in the original Arabic loan. Currently, the only way around this is to manually edit the output of the converter, but future plans for \( \ Helvetica \) include adding a list of preferred spellings to prep_rom(), so that \( \ Helvetica \) is converted to \( \ Helvetica \), and it is the latter which is then passed to rom2ar().

On a more general point, converting an existing close transcription (perhaps one in a published paper or book) into the suggested Arabic spelling will of course hardly ever regain the original Arabic spelling (whatever that was) on which that transcription was based. However, even a partial conversion may be useful in providing an Arabic text that can be edited to reflect the original spelling, rather than starting to type out the original from scratch. If the Arabic spelling in the original has well-defined features, it may also be worthwhile to edit rom2ar() to reflect those features, so that the conversion will be as close as possible to the original, thus limiting the amount of editing to be done.

The suggested spelling in \( \ Helvetica \) is to show sukun on consonants that are not followed by a vowel (eg كَوْ، kwa, by), but the prep_rom() function includes an option to turn this off (giving كَوْ).  

Likewise, the suggested spelling in \( \ Helvetica \) is not to show sukun on waw or yeh (eg كُوْمِ and not كُوْمِ), but there is a helper function, waw_yeh_sukun(), to turn that on.

The final helper function, convert_numbers(), gives the user the option of changing Roman numerals into Western Arabic ones.
Chapter 8

Typesetting poetry

As noted in Chapter 1, a key aim of Andika! is to facilitate the production of digital versions of classical Swahili manuscripts. This chapter deals with the tools provided to do that – they are based on the concept of importing each word of the text into a database table, and then adding material such as notes on individual words or sections, variant readings and emendations, translations, etc. The enriched text can then be output in a number of formats allowing for both print and online publication – see Section 8.6. This approach also facilitates the automatic production of word frequency lists, glossaries, concordances, n-grams, and so on, which open the way for detailed linguistic analysis of the text.

This chapter focusses on the possibilities for typesetting traditional poetry in Arabic script. However, the tools can also be used for prose in Arabic script – this aspect could be developed further if there is demand for it.

8.1 Creating the input document

The first step is to manually transcribe the manuscript letter-for-letter into a LibreOffice odt document – virtually all the Arabic glyphs likely to be used in a manuscript are already on the layout described in Chapter 5, and it is actually faster to type the Arabic text into the computer than it is to type in a close transliteration.

Each kipande of a poem in traditional metre should be placed on its own line, and each stanza should be separated by a blank line. The LibreOffice document should end with one blank line.

Particles such as لَزَيَنَ (na, ya, za, la) are best written according to the manuscript rendering. In many instances the author or copyist may have attached them to the following word, or (in the case of non-connecting letters like ز) placed them very close to it. But where there is a larger space in the manuscript between the particle and the following word, it may be appropriate to write it separately from that word in the transcription. These decisions are subjective, and may produce some inconsistency in the transcription, but they will reflect the manuscript more faithfully. Connection or disconnection of the particles (in line with standard Roman orthography) will be handled by annotating the entries in the database.

Samples of input files are in andika/convert/inputs and subfolders.

8.1.1 Checking the input file

It is a good idea to check your input file every few stanzas or so by using the convert tools to output a pdf with transliteration. This will help you to note and correct typos as you go along.

8.2 Importing the contents into a database table

Once the manuscript is transcribed in the input file, it can be imported into a database table. This is done by using the conversion tools, but specifying database import instead of output in a particular document format. To use the point-and-click interface (Section 7.3.1), open a terminal and run:

convert/convert.sh
choose your poem, and then select Insert into database on the Output screen. To use the command-line interface (Section 7.3.2), run the following command (replacing vita_vikuu with the name of the poem):

```
php convert/convert.php convert/inputs/vita_vikuu/vita_vikuu.odt+Arabic+Poetry ↦ +db+kip-line
```

In both cases, a database table of the same name as the poem (in the above case, vita_vikuu) will be created in the andika database, and each kipande of the poem will be imported into that table. During this import, the Arabic text is transliterated into standard Roman orthography, and also into a close transliteration which more closely reflects the Arabic glyphs.

To view the database table, you can access it using phpPgAdmin: in a browser, enter localhost/phppgadmin and log in to the PostgreSQL server. In the left panel, click the + beside andika, and then Tables, and then click on Browse alongside the name of the poem (in this case, vita_vikuu). You should see the first part of the contents of the table, and for each kipande of the poem, there will be:

- A computer-assigned index number (poemline_id).
- The stanza number as written on the original manuscript (msno). If the original stanza numbers were not entered on the input document (see Section 7.3.3), a zero will be entered here instead. The difference between this and the following stanza is that human scribes or copyists make mistakes – they may miss out a number, or give two stanzas the same number. However, for manuscript work (see Chapter 9) it may be useful to record this number in any case.
- The stanza number as generated by the software (stanza). This will always be correct (i.e. increment by 1 for each stanza in the poem). The default import numbers the stanzas from 1 onwards, and for simplicity, that is retained in the examples in this section, but it is easy to change the numbering to reflect the actual stanza numbers – see Section 7.4.1.
- The location of the line in the stanza (loc). This is in the form of a lowercase Roman letter: a, b, c, d, etc.
- The Arabic text (arabic). Some cells in the Arabic column may appear empty, but this seems to be a display bug in phpPgAdmin – if you press Edit you can see the text is there, and a select query will also show it.
- The close transcription (close).
- The standard transcription (standard).

### 8.2.1 The import process hangs

If the database import process hangs (i.e. appears to be doing nothing), this is because you have the relevant poem table open in SQL Workbench (for example, to review the import, or add annotations). If SQL Workbench is showing data from an Andika database table, it places a lock on the table which prevents any other program accessing it for some operations (re-creating a table, renaming a field in the table, etc). If you then try to run another import (eg to add some more stanzas you have just typed out), the import will hang because the script convert/create_poemlines.php cannot delete that table and re-create it. To allow the import to proceed, all you need to do is close SQL Workbench. You can then reopen it after the import is complete.

### 8.2.2 Saving a particular import

Each time a poem text is imported, the import process will delete and then recreate the poem table containing the vipande and their transliteration. There is no real need for a backup, because the table (unlike the words table – see Section 8.3) is simply a copy, in a different format of the input document. However, if for some reason you want to preserve a specific conversion (perhaps for archival purposes),
you can save (dump) the database table before re-running the conversion. To create a file which can be loaded into a PostgreSQL database, open a terminal and run:

```
pg_dump -U dbuser --table=poem_name > poem_name.sql
```

If I wanted to do this for the poem *inkishafi*, I would run:

```
pg_dump -U kevin --table=inkishafi > inkishafi.sql
```

To create a file which can be opened in a spreadsheet, run:

```
psql -U dbuser -d andika
(Replacing `dbuser` with your database user)
\copy (select * from poem_name) to 'poem_name.csv' with delimiter ',' csv header
(Replacing `poem_name` with the name of the poem)
\q
```

You can do the same thing with phpPgAdmin.

1. On the *Tables* screen of the *andika* database, click on the name of the poem.

2. Click on the *Export* button at the top of the page.

3. For a backup which can be loaded into PostgreSQL:
   (a) Tick *Structure and data*.
   (b) Select *SQL* from the *Format* drop-down list.

4. For a backup which can be opened in a spreadsheet:
   (a) Tick *Data only*.
   (b) Select *CSV* from the *Format* drop-down list.

5. Tick *Download*, and then click *Export*.

6. Select a location in which to save the file (you may want to create a directory in which to store the backups), and rename it as appropriate (you may want to add dates to the filenames).

Andika! makes automatic backups for the next stage of import, splitting lines into words (see Section 8.3.1).

### 8.2.3 Adding stanzas to a poem import

This is an unlikely scenario, but is included here for completeness. Suppose (a) your copy of a poem was missing a page, (b) you have come across another copy of that poem which contains the missing page, and (c) you no longer have access to your typed out (odt) version of the poem, but you want to include the new stanzas in the poem table without retyping the whole poem.

Assume you have 5 new stanzas which fit in after stanza 115 of the existing poem as (new) stanzas 116 to 120. You would:

1. Type out the new stanzas as a separate document, and import that as a separate poem (say, *newbits*).

2. Adjust the numbering of the *newbits* table to reflect their proper position in the existing poem:

   ```
   update stanza set stanza=stanza+115
   ```

   This will change the stanza numbers from 1 to 116, 2 to 117, and so on.\(^1\)

---

\(^1\)Alternatively, you can edit *convert/convert.php* – see Section 7.4.1.
3. Make space for the new stanzas by moving the numbering of the existing poem stanzas forward:
   ```sql
   update stanza set stanza=stanza+5 where stanza>115
   ```
   This will change the stanza numbers from 116 to 121, 117 to 122, and so on.

4. Copy the new stanzas from from the `newbits` table into the poem table:
   ```sql
   insert into poem select * from newbits
   ```

### 8.3 Splitting lines into words

Now that the text of the poem is stored in a table (`poem_name`), the next step is to split each `kipande` of the poem into words, and store those in another table (`poem_name_words`), which will allow each word to have individual annotations added to it. To do this, open a terminal and run:

```
php db/import_words.php name_of_the_poem
```

replacing `name_of_the_poem` with whatever your poem name is. In the example, you would run:

```
php db/import_words.php vita_vikuu
```

Note that once the poem is imported into the database, the code relating to annotation and other editorial work is located in the directory `db`, and not in `convert`.

The import process will create a new table, `name_of_the_poem_words` (in this case, `vita_vikuu_words`), which you can again inspect using phpPgAdmin. Each word has an entry of its own, consisting of fields similar to those used for the poem table (see above):

- An index number (`word_id`).
- The stanza number in which the word occurs, as written on the original manuscript (`msno`).
- The stanza number in which the word occurs, as generated by the software (`stanza`).
- The location of the `kipande` in the stanza where the word occurs (`loc`)
- The position the word occupies in the `kipande`, eg first word, second word or whatever (`position`).
- The Arabic text (`arabic`).
- The close transcription (`close`).
- The standard transcription (`standard`).

However, other fields are also created to hold annotations for data such as variants, general notes, an English translation, and so on (see Section 8.4).

### 8.3.1 Re-running word-splitting: backups

Note that each time `import_words.php` is run (eg as you complete another batch of stanzas), it will delete and then recreate this words table. Since you may well have devoted considerable time to editing the table and adding annotations (see Section 8.4), it is important not to lose these! So `import_words.php` does the following things:

1. Creates (or, in the case of a re-import, re-creates) a backup of the `words` table, which will include all your annotations and edits. The name of this table is poem-name plus `_backup`, e.g. `vita_vikuu_backup`.
2. Archives any pre-existing backup (from earlier imports) in a table named poem-name plus `_backup` plus date plus time. An example would be `inkishafi_backup_2017_11_23_16.45.35`, indicating that this archive backup was made at around 4.45pm on 23 November 2017.
3. Does the new import, as explained above in Section 8.3.

4. Adds your annotations and edits from the backup into the new import (see Section 8.4 below).

In the normal course of events, you should never need to use the archive backups, but they are there to cover a case where something goes wrong with the import and you immediately re-run it. This could lead to data loss – the first import run creates a backup table with all your annotations, but the second then deletes that good backup table and creates a new one based on the faulty words table, where the annotations may be missing. The result is that your backup table no longer contains your annotations!

The archive backups allow you to recover from this situation with your annotations intact, because you can copy fields manually from them to your words table by using an SQL query like:

```sql
update inkishafi_words w set (edclose, edstan, emend, variant, note, english) =
  (b.edclose, b.edstan, b.emend, b.variant, b.note, b.english)
from inkishafi_backup_2017_11_23_16_45_35 b
where w.stanza=b.stanza and w.loc=b.loc and w.position=b.position;
```

The above query will copy the contents of the listed annotation fields from the archive backup table into the words table. IMPORTANT! If you run this query in SQL Workbench, you then need to commit (save) this update by selecting SQL → Commit, or by pressing Alt+M.

If you find that you are doing a lot of re-importing and this is leading to a good few archive backups, you can simply delete all but the most recent, provided your words import has gone smoothly.

### 8.3.2 Renumbering stanzas during word-splitting

A problem can arise during word-splitting (Section 8.3.1) if between two imports you have renumbered the stanzas by changing the stanza_no counter (Section 7.4.1). For instance, if you (a) import a set of excerpted stanzas with stanza_no set to 0 (the default), (b) add annotations, (c) realise that you should have set the stanza_no to (say) 275, because the excerpted stanzas begin at 276, and (d) reimport after having adjusted the stanza_no to read stanza_number=275, then although the annotations you made previously will be in the backup, they cannot be re-integrated into the new words table because they no longer have the same stanza number.

For the above case, you can resolve this by running the following:

```sql
update poem_words w set (edclose, edstan, emend, variant, note, english, noshow) =
  (b.edclose, b.edstan, b.emend, b.variant, b.note, b.english, b.noshow)
from poem_backup b
where w.stanza-275=b.stanza and w.loc=b.loc and w.position=b.position;
```

The above query subtracts 275 from the stanza numbers in the new excerpt table, so that they now match the numbers in the backup table, and then copies the annotation fields from the backup table to the words table. IMPORTANT! If you run this query in SQL Workbench, you then need to commit (save) this update by selecting SQL → Commit, or by pressing Alt+M.

### 8.4 Annotations

It is possible to create as many fields in the words table as are required to handle various annotation types, and the content of each of these fields can be selected and edited via code, which means that a versatile framework exists for any editorial apparatus. The most important annotation types are described below, but others will be discussed in Chapter 10, when we look at computer-aided analysis of the poem text.
8.4.1 edclose field

This field holds an edited version of the close transliteration, which can be used when you want to override the output given by the automatic close transcriber. For instance, suppose that in the Mwana Kupona text (Figure 4.1) you wanted to rewrite ghain (迦) where it is used to represent ng. You could do this by adding an edclose entry nawulimingu alongside the close entry nawulimu. When the poem is typeset, an entry (if there is one) in the edclose field will replace the equivalent one in the close field.

The edclose field also allows the re-use of a specific existing close transliteration – for instance, if you are typing out a manuscript which has already been published, and you need to replicate the close transcription from that publication. In this case, you would add an edclose entry alongside each close entry to reflect that existing close transliteration.

8.4.2 edstan field

This field holds an edited version of the standard transliteration, which can be used when you want to override the output given by the automatic standard transcriber. When the poem is typeset, an entry (if there is one) in the edstan field will replace the equivalent one in the standard field.

This can be used in a number of situations:

- To make spelling corrections where the automatic transcriber gets things wrong, e.g. to change akanyamaa to akanyamaa. (eg changing ḡ to ng, or i to e), the edclose field currently handles two other types of edits: and insertion of additional letters.

- To segment the transliteration to reflect standard Swahili word boundaries, e.g. to separate particles like ‏(na, ya, za, la), etc that have been written connected in the Arabic text. For example, in 1c of vita_vikuu, you could add na khubuzi in the edstan field alongside the standard field nahubuzi.

- To connect elements that have been written separately in the Arabic text. To do this, enter ~ in the edstan field alongside the standard field element to be joined, and move the element to the proper cell. For instance, in 1g of kiswahili, standard Swahili requires nimewatendani (what have I done to you?) instead of ni mewatendani. So against ni we put a ~ in the edstan field, and edit the edstan field of mewatendani to read nimewatendani.

- To add epenthetic vowels to support proper scansion, you can use \In{text here} (more information on annotation fonts is given in Section 8.5.1 below). For instance, in 1c of vita_vikuu an epenthetic vowel can be added to the transliteration of مِلْحِ (milḥi, and salt) to give miliḥi by editing the equivalent edstan field to read na mil\In{i}ḥi.

- To allow the re-use of a specific existing standard transliteration – for instance, if you are typing out a manuscript which has already been published, and you need to replicate the standard transliteration from that publication. In this case, you would add an edstan entry alongside each standard entry to reflect that existing standard transliteration.

With some writers (e.g. Sheikh Yahya), the standard transcriber will result in text that is very close to standard Swahili in Roman script, because their Arabic script marks distinctions that reflect the phonology of Swahili. Other writers (usually from earlier periods) have used the Arabic script as an approximation of the phonology of Swahili (see Chapter 4) so the standard transliteration will reflect standard Swahili in Roman script less well. In these cases, instead of re-typing entries into the edstan field, it may be easier to copy the contents of the standard field to edstan, and edit that (or delete entries where they are identical) – this may help reduce typos due to mistyping. The following SQL query will accomplish this:

```sql
update poem_word set edstan=standard;
```
### 8.4.3 emend field

This field can be used where the copyist has written an incorrect word that requires emendation. When the poem is typeset, an entry in this field will replace the equivalent entry in the standard field, and will be marked in the output by a dotted underline below the word. For example, in the Inkishafi one kipande reads نیسیکیا مان نکیپیاپی (nisikia maana nikwambiapo), which is one syllable longer than the expected 11. This can be resolved by adding sikia in the emend field alongside nisikia, and the line will then be output sikia maana nikwambiapo.

Note the distinction between the emend field and the edstan field. The former is used where the copyist has made an error that needs to be corrected. The latter is used where the the written text only approximates standard Swahili, and needs to be rectified – an example would be where the Inkishafi text above reads nikabiyaapo, and we need to change that to nikwambiapo to reflect modern standard Swahili.

### 8.4.4 variant field

This field allows the recording of variant readings (variae lectiones) of the word in different manuscript versions of the same poem. For instance, 6a of vita_vikuu reads: نِسِكِيَمَانَنِكِبِيَاپُ (nisikia maana nikwambiapo, and a good, stout sword). If you have another manuscript B where this kipande reads: نِسِفُنْجٖمَأَسِيْسِ (nasēfu njēma asisi), in the variant column against njēma we put (for instance):

B: \AS{نْزُوْرِ}، nzūri

which will be converted to a footnote giving:

B: بَنْزُورُ، nzūri.

### 8.4.5 note field

This field holds notes on the meaning or reference of the word, or any other material which may help to elucidate its meaning or usage. There is no practical limit on the length of these notes (the longest possible content that can be stored in any one field is about 1 GB). Appendix D demonstrates how Andika! caters easily for a significant number of notes – for more information the relevant database table, kiswahili_words, can be inspected.

To refer to other footnotes within notes, add a reference label to the source footnote: e.g.

\label{fn:mynote}This is a footnote that we also want to refer to later.

In the referring footnote, call that reference:

This footnote makes the same point we already made earlier - see footnote \ref{fn:mynote}.

### 8.4.6 english field

This field holds an English translation of the whole kipande, placed against the first word of each kipande. See the kiswahili_words table for examples.

### 8.4.7 noshow field

This field can be used to switch off the display of the standard transliteration for words or lines in the poem. It will be discussed in more detail in Section 8.8 below.
8.5 Adding and editing the annotations

The best tool for adding the annotations to be added directly to the table is SQL Workbench (Appendix A/12.2), since it allows direct entry into the individual fields. Open SQL Workbench and connect to the andika database. In the top panel, enter an SQL query to show the whole of the vita_vikuu_words table in order of stanza and kipande:

```
select * from vita_vikuu_words order by stanza, loc, position;
```

(note that the semi-colon at the end is essential)

Move the cursor somewhere inside the query and press Ctrl + Return. You can now edit individual cells in order to add annotations.

IMPORTANT! Remember to press the Save icon or select Data → Save Changes to Database to save any material you add. Even though you can see your edits in SQL Workbench, they are not written into the database until you save them!

8.5.1 Specifying fonts in the annotations

Changes from the default font can be marked in any of the annotation types, and it is recommended to make such changes directly in the annotation, rather than try to edit the output file afterwards. The alternate fonts use the font definitions as set up in the file andika/convert/tex/fontdefs.tex, and you can add more to that list using the patterns seen there (see also Section 4.2). To apply the font changes, the LaTeX command format is used: a backslash, then the abbreviation for the desired font, and then the text inside braces. Likely font changes that might be applied are:

- Arabic script: \AS{text here} – Scheherazade. To use Amiri, use \Am{text here}
- transliteration: \Tr{text here} – Linux Biolinum O in grey.
- inserted letters: \In{text here} – Linux Biolinum O in blue.
- standard Swahili: \Swa{text here} – Linux Biolinum O in green.
- English: \E{text here} – Liberation Serif in grey italics, smaller than the default.
- English italics: \Eit{text here} – Liberation Serif in black italics.
- standout type in footnote: \FN{text here} – Liberation Serif in green italics.

8.5.2 Inserting citations in the annotations

All the annotation fields can include citations, but they are perhaps most likely in the note field. The citations use a LaTeX package called biblatex, which has already been installed (Appendix A/13). This draws on a list of the citations you wish to use, which need to be in BibTeX format – see andika/bib/andika.bib for an example of a short bibliography file. It is possible to write this file using just a text editor, but it is easier to use a frontend such as JabRef, which has already been installed (Appendix A/14). With a frontend, you simply type the bibliographic details of the citation into a series of dialogue boxes. Each citation is referred to by a “key” – I use the name of the first author and the year, so the key for Sacleux’ Dictionnaire would be Sacleux1939, but you can use anything you like.

Using this dictionary as an example, we can then refer to a citation in the note by using:

```
\textcite{Sacleux1939}
```

where the work is a subject:

Sacleux (1939) was a major achievement in Swahili lexicography.

\textsuperscript{2}The older bibtex package does not support citations in footnotes.
CHAPTER 8. TYPESETTING POETRY

and using:
\parencite{Sacleux1939}
where the work is referred to parenthetically:

This word is found mainly in northern dialects (Sacleux 1939).

The author-year citation style is default, but it can be changed. However, since citation management and style is a wide-ranging topic not directly relevant to Andika!, this aspect is not dealt with here.

The bibliography file has to be called andika.bib, and has to be in the andika/bib directory. You can either add your citation details to that file, or delete it and start your own andika.bib. If you need to use a bibliography file of a different name in a different location, you can delete the existing andika.bib and set up a symbolic link to your own file. For instance, if your bibliography file were called thesis.bib, and it was located in /home/USER/thesis (where USER is your username), you would run the following commands in a terminal open at andika:

```bash
rm bib/andika.bib
# delete the existing andika.bib file
ln -s ~/thesis/thesis.bib bib/andika.bib
# create an andika.bib link (alias or shortcut) to your bibliography file – remember that ~ stands for /home/USER
```

It is worth noting that in cases where citations are not printed properly, it is virtually always the case that there is an error in the bibliography file, or that the key being used is incorrect.\footnote{I recently spent almost an hour trying to work out why the entry for Abdulkadir and Frankl (2013), key Abdulkadir2013, was not appearing correctly - there were definitely no mistakes in the bibliography entry. Yet LaTeX reported it could not find that entry. I tried various increasingly complicated fixes, until eventually I noticed that I had typed the key as Adbulkadir2013 when citing the work. When a problem like this arises, we tend to think that the most likely cause is a fault in the software, but it is far more likely to be a fault in the wetware connecting the chair to the computer.}

8.5.3 Adding annotations in batches

When working on a long poem, it is often more convenient to add annotations in batches of (say) 20. This means that your work can be reviewed easily, and typos corrected, without having to scroll through the whole words table in SQL Workbench, and it also means that generation of the pdf for checking will be faster, since it only includes a subset of the poem’s stanzas.

To do this, set the query in SQL Workbench to call only the relevant items from the words table. In the following query, you are working on stanzas 80-100 of jaafari, and need to see only the standard and edstan fields so that you can add an English translation and any notes that are appropriate:

```sql
select word_id, stanza, loc, position, standard, edstan, english, note from jaafari_words
order by stanza, loc, position;
```

Skipping ahead a little (see Section 8.7.1), to print out only this batch of stanzas, use the option MtoN:

db/output_pdf.php jaafari 80to100

8.6 Output to pdf

Once the database table holding the words of the poem has been edited to include all the annotations required, it can be output in a variety of formats. The most important is the pdf format,\footnote{Output to html will be added at a later stage.} since the LaTeX typesetting system produces a very attractive, beautifully laid-out text equally suitable for traditional printing or digital distribution. A variety of layouts is possible (see also Chapter 9) but the default is to print two vipande to the line, with an asterisk between them.
CHAPTER 8. TYPESETTING POETRY

To print your annotated poem in the default layout to a pdf file, run:

```
php db/output_pdf.php poem_name
```

The generated pdf will be located in a folder with the same name as your input document – in this case `andika/db/outputs/poem_name`.

To demonstrate the layout options, we will use one stanza from the beginning of the *Utenzi wa Jaafar*. The input document for this is in `convert/inputs/egoutput.odt`. After converting to database format, importing the words, and adding an English translation, we can run:

```
php db/output_pdf.php egoutput
```

to give `andika/db/outputs/egoutput/egoutput.pdf` – see Figure 8.1.

![Default output](default_output.png)

*Figure 8.1: Default output*

### 8.7 Output: global display options

The default output in Figure 8.1 can be adjusted in a number of ways by passing various options to `output_pdf.php`. Those in this section are “global” options, in that they apply to the poem as a whole. The next section (Section 8.8) will explain some further options which are “local”, in that they can be applied to individual words or lines in the poem.

#### 8.7.1 Stanzas to be printed

By default, all stanzas of the poem will be printed out. However, you can print out a contiguous subset of stanzas by using the option `MtoN`, where `M` is the number of the first stanza in the set, and `N` is the number of the last stanza in the set. For instance, the following command will print out only stanzas 80 to 100 of the poem *Jaafari*:

```
php db/output_pdf.php jaafari 80to100
```

#### 8.7.2 Font size

The default font size for the output is 10pt, which is good for printing. However, if you need a slightly larger font, you can use the option `12pt` to increase font size to 12 pt.

```
php db/output_pdf.php egoutput 12pt
```
CHAPTER 8. TYPESETTING POETRY

8.7.3 Changing the location of annotations

By default, all annotations are output as footnotes – since they refer to individual words in the text, having them on the same page is easier to read. But if you need endnotes instead, you can use the option endnote to remove the annotations from the bottom of the page, and move them into their own section after the poem:

```php
db/output_pdf.php egoutput endnotes
```

To compare the difference in output, look at Appendix D and Appendix F.

8.7.4 Colours

The Arabic script is printed in green, but this can be changed in `output_pdf.php` by editing the line:

```
$colour1="mygreen";
```

to use one of the default colours (red, green, blue, cyan, magenta, yellow, black, gray, white, darkgray, lightgray, brown, lime, olive, orange, pink, purple, teal, violet), or a new default colour can be set up at the bottom of `convert/tex/fontdefs.tex`.

The first line of Arabic script can be printed in a different colour by using the option firstcolour:

```php
db/output_pdf.php egoutput firstcolour
```

giving the output in Figure 8.3. Again, the colour can be adjusted in `output_pdf.php` by editing the line:

```
$colour2="blue";
```

or a new default colour can be set up at the bottom of `convert/tex/fontdefs.tex`. This option is useful for setting off lines in Arabic when they occur in the same stanza as lines in Swahili (e.g. in the Burda – see Section 2.8).
CHAPTER 8. TYPESETTING POETRY

Listen, my brothers, I went on a journey.

Let me give you my story, an account of what happened to me.

Figure 8.3: Option: firstcolour

If no colour other than black is desired in the output, use the option nocolour (see Figure 8.4):

php db/output_pdf.php egoutput nocolour

Figure 8.4: Option: nocolour

8.7.5 Alignment

As Figure 8.1 shows, the default alignment for the poem text is centred, but this can be changed to right-justified by using the option alignright (see Figure 8.5):

php db/output_pdf.php egoutput alignright

Figure 8.5: Option: alignright

The option alignleft is also available, but this is best applied where the Arabic script is not being printed out (see Section 8.7.6).
8.7.6 Omitting tiers

Even if an English translation has been added to the words table, it can be omitted by using the option `noenglish` (see Figure 8.6):

```bash
php db/output_pdf.php egoutput noenglish
```

![Figure 8.6: Option: noenglish](image)

The standard transcription can also be omitted by using the option `nostandard` – adding this to the `noenglish` option means that only the Arabic script will be printed (see Figure 8.7). This may be useful if the Arabic script is being relegated to an appendix, with the standard transcription and/or the English translation being extracted and typeset separately to form the main text. Since the only remaining tier is written right-to-left, it may also be appropriate to apply the option `alignright`. Multiple options can be passed (in any order) to `output_pdf.php` by linking them with a plus sign:

```bash
php db/output_pdf.php egoutput nostandard+noenglish+alignright
```

![Figure 8.7: Option: nostandard + noenglish + alignright](image)

Note that if the standard transcription is suppressed, any footnote numbers attached to it (for variant readings or notes) will be attached to the Arabic script instead. Also note that no emendations at all will be shown, since they are assumed to apply only to the standard transcription.

The complement of the example in Figure 8.7 is where you want to print only the standard transcription and the English translation – the `noarabic` option will suppress the output of the Arabic script (see Figure 8.8). Since the remaining tiers are written left-to-right, it may also be appropriate to apply the option `alignleft`.

```bash
php db/output_pdf.php egoutput noarabic+alignleft
```

![Figure 8.8: Option: noarabic + alignleft](image)

Listen, my brothers, I went on a journey.

Let me give you my story, an account of what happened to me.
8.7.7 Displaying the close transcription

In the case of older manuscripts where the Arabic script is more of an approximation to the Swahili it represents, it may be helpful to provide a close transcription as well. This will be printed in the same colour as the Arabic script above it, and the option close-lr will print it left-to-right (see Figure 8.9).

```
php db/output_pdf.php egoutput close-lr
```

Listen, my brothers, I went on a journey.

```
پُلِكَنِنْدُزَنْڠُ*kehんд茅mمبِرِ ينْغُ
пulikan nduzangu * kʰenda matembezi yangu
pulikan nduzangu * kenda matembezi yangu 1a/b
Listen, my brothers, I went on a journey.

١

پُلِكَنِنْدُزَنْڠُ*قهًد ماٮمٮر ينْغُ
پلیکانِ ندژانگُ * kʰندامِبِژی ینْگُ
pulikaninduzangu kʰenda*nduzangupulikani
Listen, my brothers, I went on a journey.

niwape khabari zangu * qîṣa chalonjiriya
niwape habari zangu * qîsa chalonjiriya 1c/d
Let me give you my story, an account of what happened to me.
```

**Figure 8.9: Option: close-lr**

The option close-rl is also available, and will print the close transcription right-to-left; that is, the close transcription of each word will appear directly under the Arabic script for that word (see Figure 8.10).

```
php db/output_pdf.php egoutput close-rl
```

Listen, my brothers, I went on a journey.

```
پُلِكَنِنْدُزَنْڠُ*قهًد ماٮمٮر ينْغُ
پلیکانِ ندژانگُ * kʰندامِبِژی ینْگُ
pulikan nduzangu * kʰenda matembezi yangu 1a/b
Listen, my brothers, I went on a journey.

١

پُلِكَنِنْدُزَنْڠُ*قهًد ماٮمٮر ينْغُ
پلیکانِ ندژانگُ * kʰندامِبِژی ینْگُ
pulikan induzangu kʰenda*nduzangupulikani
Listen, my brothers, I went on a journey.

niwape habari zangu * qîsa chalonjiriya
niwape habari zangu * qîsa chalonjiriya 1c/d
Let me give you my story, an account of what happened to me.
```

**Figure 8.10: Option: close-rl**

Simultaneously suppressing the standard transliteration will leave only the close transcription – see Figure 8.11. In this case, as noted above (Section 8.7.6), any annotations giving notes or variants will be attached to the Arabic script. Note that in this case there is no Roman numbering of the stanzas – if this is required, an alternative method is suggested below.

```
php db/output_pdf.php egoutput close-rl+nostandard
```
CHAPTER 8. TYPESETTING POETRY

8.8 Output: local display options

The output options described in Section 8.7 are “all or nothing” – they apply to the whole poem. Andika! provides a further mechanism to control output when you wish an option to apply only to particular words or vipande. This is based on the noshow field (Section 8.4.7). By default, this field is empty for each word, but if a tag is entered in the field, the display of that word can be controlled in detail. Currently, this covers the transcription output only, but it could easily be extended to handle other aspects of the display.

The principle can be demonstrated by looking at how to suppress the standard transcription for a word – let us assume that we want to suppress the printing of matembezi in 1b. To do this, open the words table in SQL Workbench and put something (anything) in the noshow field – the only condition is that the something should be less than 10 characters long (you are also recommended to make it at least two characters long, so that you can’t confuse it with the vipande notation). To focus on what we are doing, the SQL query will pick out only those fields relevant to this task:

```
select word_id, stanza, loc, position, arabic, close, standard, noshow from egoutput_words order by stanza, loc, position;
```

In the noshow field alongside matembezi, enter something so that the entry looks like Figure 8.12, and then press the Save icon or select Data → Save Changes to Database to save this edit.

Running output_pdf.php will then print the text with that word missing from the transcription (see Figure 8.13):

```
php db/output_pdf.php egoutput
```
CHAPTER 8. TYPESETTING POETRY

Listen, my brothers, I went on a journey.

Let me give you my story, an account of what happened to me.

Figure 8.13: Suppressing the transcription for a word

In order for omitted words to be shown as having been omitted, output_pdf.php has a preset to mark them. In SQL Workbench, change “blah” to “omit”, and then save the edit. Running output_pdf.php again will now show the omitted word’s location with three grey Xs (see Figure 8.14):

Figure 8.14: Omitting a word

8.8.1 Suppressing a line of transcription

It is difficult to envisage an editorial situation in which omitting a single word might be useful, but suppressing a transcription for all the words in a line might be so in some cases. An example might be stanzas which contain lines in either Arabic and Swahili (e.g. the Burda) – here you might want to present the Arabic without transcription and the Swahili with. This can be achieved by making an entry in the noshow field against all of the words in the line you want no transcription for. If we want to suppress the transcription for the first line of the sample stanza, for instance, we can enter “ar” against all of the words in both vipande5 (see Figure 8.15 – in effect, by doing this we are applying the global nostandard option to that line alone. This gives the output in Figure 8.16. Note that the stanza/vipande information in the margin is no longer printed, since there is no transcription for it to apply to,

5Strictly speaking, marking the words in just one vipande will suppress the transcription in both – i.e. marking the words in vipande c will suppress the transcription in d as well as that in c. This is to make the display neater, but it is possible to suppress only one vipande in the line instead of both – see the note in the output_pdf.php code.
### 8.8.2 Swapping in the close transcription

However, short of creating a new close transcription, another possibility is to use the existing one provided by Andikal, perhaps editing it as necessary by making entries in the edclose field (see Section 8.4.1). You can do this by using the option swapclose, which inserts the close transcription instead of the standard transcription in any line where the words have an entry in the noshow field. (Note that swapclose will have no effect if there is no entry in the noshow field - in that case, the standard tran-
scription will be printed out as normal.) The example in Figure 8.18 uses the “areng” entry against the
words in line 1 – the English translation could also be suppressed by changing that entry to “ar”.

```
php db/output_pdf.php egoutput swapclose
```

It is worth noting that swapping in the close transcription can be used for any lines where there is a
relevant entry in the noshow field – i.e. it does not just apply to vipande a and b.

![Figure 8.18: Swapping in the close transcription](image)

8.8.3 Making entries in the noshow field

It is easy to run an SQL query to add tags to each noshow field:

```
update egoutput_words set noshow='ar';
```

to delete all of them:

```
update egoutput_words set noshow='';
```
or to change one to another:

```
update egoutput_words set noshow='ar' where noshow='areng';
```

To simplify adding tags to each noshow field, Andika! provides a helper script, manage_noshow.php. (It
is also possible to use SQL queries directly on the words table itself, but the script makes things a little
faster.) Arguments are passed to the script in the following order:

1. Name of the poem (i.e. the words table shorn of _words).
2. The action to be taken. Two actions are possible: fill (to insert tags), and clear (to remove them).
3. The stanzas to be targeted. There are two possibilities: all, which will apply a tag to the words
   in every stanza in the poem, or a list of stanzas. The latter is a comma-separated list of numbers
   with NO spaces, which can also include ranges separated by a hyphen. For instance, 2,5,6 would
target those three stanzas, while 4,6-8,10-13,17,20 would target the stanzas 4, 6, 7, 8, 10, 11, 12,
   13, 17 and 20.
4. The locations (vipande) to be targeted. This is a comma-separated list of lower-case letters with NO
   spaces, which can also include ranges separated by a hyphen. For instance, a,b or a-b would target
   vipande a and b, while c-d,g-h would target vipande c, d, g and h. This argument can be omitted
   if you are using the default tag “ar” – if vipande within the stanza are not specified, the “ar” tag
   will be applied to the words in every kipande in that stanza. HOWEVER, if you are specifying a tag
   other than “ar”, you must include this argument.
5. The tag to be inserted, e.g. ar, areng, omit, or a tag of your own devising. This argument can be
   omitted, in which case the default tag “ar” will be used. With the clear action, no tag is necessary,
   but if one is given it will be ignored.
Some examples follow:

Add an “ar” noshow tag to every word in the table poem_words (remember that the “ar” tag is applied by default if no tag is specified):

```
php db/manage_noshow.php poem fill all
```

Delete all noshow tags from the poem_words table:

```
php db/manage_noshow.php poem clear all
```

Add an “ar” tag to every first line (specifying “ar” is optional):

```
php db/manage_noshow.php poem fill all a,b ar
```

Add an areng” tag to every third line:

```
php db/manage_noshow.php poem fill all e-f areng
```

Change all tags in the third line (“areng”) to another (“mytag”):

```
php db/manage_noshow.php poem fill all e-f mytag
```

Add an “ar” tag to every word in stanzas 1, 3 and 5:

```
php db/manage_noshow.php poem fill 1,3,5
```

Add an “areng” tag to every word in stanzas 2, 4 and 6 – assuming 6 vipande in each stanza (remember that if you are using a tag other than the default “ar” you not only have to specify that, you also have to specify the vipande):

```
php db/manage_noshow.php poem fill 2,4,6 a-f areng
```

Add an “ar” tag to the first line of stanzas 6, 7, 8, 11, 12, 13 and 21:

```
php db/manage_noshow.php poem fill 6-8,11-13,21 a-b
```

---

6 SQL: update poem_words set noshow='ar';
7 SQL: update poem_words set noshow='';
8 SQL: update poem_words set noshow='ar' where loc in ('a', 'b');
9 SQL: update poem_words set noshow='areng' where loc in ('e', 'f');
10 SQL: update poem_words set noshow='mytag' where loc in ('e', 'f'); or: update poem_words set noshow='mytag' where noshow='areng';
11 SQL: update poem_words set noshow='ar' where stanza in (1, 3, 5);
12 If you accidentally omit the vipande locations, there’s no harm done – no changes will be made to the table.
13 SQL: update poem_words set noshow='areng' where stanza in (2, 4, 6) and loc in ('a', 'b', 'c', 'd', 'e', 'f');
14 SQL: update poem_words set noshow='ar' where stanza in (6, 7, 8, 11, 12, 13, 21) and loc in ('a', 'b');
Chapter 9

Typsetting multiple versions of the same poem

[Draft – to be revised]

Information on aligning stanzas and printing them out, with especial attention to tenzi.

9.1 Set up a manuscript information table

First, run:

```
php ms/create_mslist.php <name of poem>
```

which sets up a table to hold key information about the MSs: the name of the table holding the database version of that MS, the code (id) for that MS, and the colour you want the MS text printed in. Once the table is set up, add this information manually (eg using SQL Workbench). MS identifiers can be entered in either uppercase or lowercase, but they will be changed to all lowercase by init_runord.php (because having single capitals as table names causes problems in PostgreSQL).

9.2 Set up a running order table

Then create a table to hold the running order for all the MSs, and fill it with one of the MS's stanza numbers.

```
php ms/init_runord.php <name of poem> <MS code>
```

It doesn’t particularly matter which MS you choose to input as the reference MS here, but it’s probably best to input whatever one you consider “primary”, since it will be the one against which data from the other MSs is entered. The primary MS might be the longest, or the most important, the most “poetic”, or whatever.

9.3 Add manuscript data to the running order table

For each MS other than the primary one, manually align its stanza numbers to those of the primary MS:

```
php ms/runord.php <name of poem> Yn[-n] action Xn
```

This assumes that you have inspected the different MSs, and know which stanzas in each you want to select or align. Using the options passed in to the script, stanza numbers will be inserted or deleted to give a “running order” of the “best bit” stanzas from all the MSs (sequential), or giving analogue stanzas from each MS (parallel). The combined MS will be printed with the provenance of the different stanzas noted by means of font, fontsize, or colour. For ballads (tenzi), for instance, you can use db/stacktenzi.php.

X will usually be the reference MS, and Y is the MS you want to align with it. The MS ids can be given as either lowercase or uppercase - the latter is easier to read, but the MS ids will be changed to lowercase anyway.

n is a number or range of numbers (n-n) specifying (for Y, the new MS) the stanza or stanza span to be
inserted (the splice), or (for X, the reference MS) the insertion point (the target).

Yn[-n] insert Xn – insert stanza n or stanzas n-n from Y AFTER the insertion point Xn. This generates a gap in stanza numbers for X that occur after the insertion point.

Yn[-n] prefix Xn – insert stanza n or stanzas n-n from Y BEFORE the insertion point Xn. This generates a gap in stanza numbers for X that occur before the insertion point. It is recommended to use “insert” preferentially, and keep “prefix” for situations where space needs to be created at the head of the table, e.g. to allow stanzas to be added which precede the reference MS’s running order.

Yn[-n] align Xn – paste stanza n or stanzas n-n from Y alongside existing stanzas in X, starting at the insertion point Xn.

After each action, a unified running order is produced, which represents the consolidated order of all stanzas in all the MSs. This can be used in two ways:

(a) Aligned (stacked): Step through the running order, and for each stanza number print out aligned stanzas from each MS. This provides output where all the MSs can be compared in detail.

(a) Sequential: Step through the running order, and for each stanza number print out the equivalent reference MS stanza. Where one does not exist, print out the equivalent stanza from one of the other MSs. This provides output where the story content can be more fully appreciated.

Although X will usually be the reference MS, it can be any of the MSs - for instance, if MS A starts at stanza number 1, which is equivalent to stanza 5 in MS B, you would run (see below):

```bash
php ms/runord.php <name of poem> B1-4 prefix A1
```

to insert the first 4 stanzas from B in front of the existing A stanzas. If you then had another MS C where stanza 3 was equivalent to stanza 1 of B, you could then run:

```bash
php ms/runord.php <name of poem> C1-2 prefix B1
```

to insert the first 2 stanzas from C in front of the existing B stanzas.
Chapter 10

Computer-aided analysis of poem texts

[Draft – to be revised]

Using queries on the words table to assist the analysis of the poem, adding new annotations as necessary.

10.1 Additional annotation fields

10.1.1 root field

Recording the root of the word is useful for building concordances, frequency lists, etc. Instead of having to search for different forms of the word, it is possible to search on the root and have all forms presented.

For Bantu words, use the stem minus any elements such as class prefixes, verbal -a or verbal extensions. Thus **fik** would find **akafika** (he arrived), **watafikia** (they will arrive at), **mfiko** (arrival), and so on.

For Arabic words, use the triconsonantal root (١٥٢). Thus **klm** would find **katalama** (he spoke), **kalima** (word), and so on.

Ideally, the roots would be filled in automatically during import by looking up the word against a digital Swahili dictionary, such as the one used for my Swahili verb segmenter. That dictionary is not yet extensive enough for that purpose.

---

1 en.wikipedia.org/wiki/Semitic_root
2 kevindonnelly.org.uk/swahili/segmenter. The dictionary here is a heavily-customised version of Beata Wójtowicz’s English-Swahili dictionary at freedict.org.
References


Appendix A

Installing Andika!

I am grateful to Natalie Kontny, Student Assistant on Project C07, The Place of Swahili Manuscripts in East African Collections, at the University of Hamburg’s Centre for the Study of Manuscript Cultures,1 for road-testing these instructions and helping to write them up.

A/1 How much of this do I need to do?

If you simply wish to be able to type Swahili in a word-processor, you need not go through the full installation procedure in this appendix. After installing Ubuntu, you only need to follow Appendix A/6.1, move into the unzipped download, and then follow Appendices A/7 to A/9.

However, if you wish to transcribe, edit and annotate Swahili documents in Arabic script, providing Roman equivalents as well, then you will need to carry out the full installation procedure here. The reason there is so much to install is that Andika! tries not to reinvent the wheel – rather, it combines already-existing best-of-breed software to do new things.

Most of the install is carried out by typing in commands directly. This is because this method is much faster and more succinct than explaining how to point-and-click through various dialogue boxes.

A/2 Ubuntu Linux

Andika! was developed on Ubuntu 14.04,2 a variety of GNU/Linux,3 a secure and free operating system. “Free” here means not only that it is available at no cost (free as in beer), but also that the user is free to copy, change and distribute it without fear of copyright lawsuits (free as in freedom).4 Ubuntu is a user-friendly packaging of a wide variety of software which is suitable for any computing need – the name is a Southern Bantu cognate of the Swahili utu, and means “humanity” or “humanness”: it is intended to emphasise that the free software concept of sharing brings out the best in all of us.

You can download the current 64-bit5 version of Ubuntu 14.04 from http://www.ubuntu.com/download/desktop You can install Ubuntu as the sole operating system on a computer (highly recommended),7 or install it alongside an existing operating system so that you can “dual-boot” into either operating system.8

Another possibility is to run Ubuntu inside your existing operating system, as a “virtual machine”. This will work well on most machines, though there may be issues with some, and it is always less efficient than running Ubuntu directly “on the metal”. Notes on installing a virtual machine are in Appendix A/4.

---

1www.manuscript-cultures.uni-hamburg.de/Projekte_e.html
2ubuntu.com
3en.wikipedia.org/wiki/Linux
4fsf.org
5Most modern machines should be 64-bit capable.
6Although versions such as 15.04 are now available, it is best to stick to 14.04, since this is a Long Term Support release which will be supported for 5 years.
7ubuntu.com/download/desktop/install-ubuntu-desktop
8Doing an internet search on “dual-boot Ubuntu” should produce a number of guides, such as the one at linux.about.com/od/LinuxNewbieDesktopGuide/ss/The-Ultimate-Windows-81-And-Ubuntu-Dual-Boot-Guide.htm

58
A/3 Conventions

Unless otherwise indicated, lines in monospaced font are commands to be typed in.

Unless otherwise indicated, all commands should be activated by pressing Return at the end of the command.

The symbol → at the beginning of a line means that it is a continuation of the previous command, and therefore Return should only be pressed after the end of this line.

Keys separated by + should be pressed simultaneously. Thus Ctrl+X means “press the Ctrl key at the same time as the X key”.

When a command starts with sudo, you will be asked to type in your superuser (administrative) password, which you should have been asked to set up when you first installed Ubuntu, before the command is allowed to proceed. Note that you will get no feedback from the password entry (the line will stay blank) until you press Return.

If at any point the system suggests adding other packages (called dependencies) based on the ones you are installing, accept those suggestions by pressing Y or typing yes.

Unless otherwise indicated, it is assumed that all commands are run from the suggested base directory of the Andika! software, /var/www/andika – see Section A/6.

A/4 Running Ubuntu as a virtual machine

As noted above, this option is less versatile than a proper install of Ubuntu, so the following notes do not attempt to cover every issue.9

Install the version of Oracle’s VirtualBox software10 appropriate for your operating system. Once installed, open VirtualBox Manager and click the icon for New. Fill in Andika against Name, Linux against Type and Ubuntu (64 bit) against Version. Click Next. As the memory amount, set 2000Mb if you have at least 3Gb memory in your machine – raise the level if you have more memory (if you have less memory you will need to accept that VirtualBox may not run very well. Click Next. Tick Create a virtual harddrive now. Click Next. Tick VDI. Click Next. Tick Dynamically allocated. Click Next. Set 40Gb as the virtual hard drive size if you have a hard drive of at least 300Mb. Click Create.

You have now set up a virtual machine, and the next step is to install Ubuntu on it. Click the new Andika entry in the left-hand pane so that it is highlighted. Click the icon for Start. You will be asked to select a startup disk. Click the folder icon to the right of the textbox to navigate to wherever you stored your download of Ubuntu 14.04. Click OK. Click Start. The Ubuntu boot process will start, and after a couple of minutes you should see an screen with two large icons on it. Click Install Ubuntu. Click through the screens, accepting the defaults. It is important to make a note of your username and password. After about 20 minutes, you should have a new Ubuntu install using the Unity desktop.

However, the screen resolution is limited to 800x600. To get higher resolutions, you need to install VirtualBox’s Guest Additions. In the running Andika instance of Ubuntu, press Ctrl + Alt + T to get a command-line terminal. Update the lists of software packages on the machine:

```
sudo apt-get update
```

entering your password when requested (the one you entered during the Ubuntu install). Then upgrade any software packages to the most recent available versions:

```
sudo apt-get upgrade
```

---

9In particular, running Ubuntu as a guest on an Apple Mac OS X host throws up some keyboard problems – it is unclear how you access the | (pipe) and \ (backslash) keys on an Apple keyboard when you need to use them in Ubuntu. In an Apple UK keyboard they are accessed respectively by Shift + Alt + L and Shift + Alt + /, but these keystrokes do not work in virtual machines.

10virtualbox.org
Note that this may take 15-20 minutes, depending on your system. Install software that the system will use to build other software:

```
apt-get install dkms
apt-get install build-essential
```

Now, click **Devices** on the menu-bar of the VirtualBox software on your host machine, and select **Install Guest Additions**. A CDROM icon will appear on the Ubuntu desktop – click on it to install the additional software. Once completed (which may take 5 minutes or so), shutdown the Ubuntu Andika instance by closing the window it is running in. Then restart Andika again from VirtualBox. If you now resize the Andika window, or select **Fullscreen** mode in VirtualBox, you should get the full resolution possible on your screen.

### A/5 Change the desktop to KDE

Once you have Ubuntu installed (whichever method you choose), you can go on to install **Andika!**. Ubuntu comes by default with a desktop called **Unity**, but a variety of different Linux desktops are available, of which perhaps the most popular is **KDE**. Since KDE is easier to work with, it is a good idea to change the desktop to KDE (though this is not essential).

Open a terminal (**Ctrl+Alt+T** on Unity), and update all software:

```
sudo apt-get update
sudo apt-get upgrade
```

Then install KDE:

```
sudo apt-get install kubuntu-desktop
```

Note that this may take some time to complete.

Log out of Unity by pressing the wheel icon at the top right of the screen, and at the login screen select KDE as your preferred desktop by clicking on the Ubuntu symbol above the login box.

Once you have logged in to KDE, right-click the K on the lower left of the screen and select **Switch to Classic Menu Style**. You can then bring up a terminal by selecting **K → System → Konsole**. You can also drag the menu entry to the panel at the bottom of the screen to allow for faster access.

### A/6 Download Andika!

#### A/6.1 Option 1: snapshot

The **Andika!** software is available from the ThinkOpen website. The second-best option is to download an archive by going to [http://thinkopen.co.uk/git/andika](http://thinkopen.co.uk/git/andika), and clicking on the **ZIP** or **TAR** buttons. Save the archive to your home directory (**/home/USER** – USER here stands for the username you set up when you installed Ubuntu; replace it with your actual username) and uncompress it to create an **andika** folder there:

```
cd ~
```

(The tilde is a shortcut for **/home/USER**.)

For a zip file:

```
unzip -q andika-xxxxxx.zip
```

The xxxxxxx segment needs to be replaced with whatever is in the name of the file you download.

For a tar file:

```
```
tar -xf andika-xxxxxx.tar
The xxxxxx segment needs to be replaced with whatever is in the name of the file you download.

A/6.2 Option 2: easy update

The above option will give you a snapshot of Andika! at the time you downloaded it, but since Andika! is a work-in-progress, it is likely to change. A much better option, if you want to keep up with any changes, is to use Git, which keeps track of changes made to files. Many free and open-source software projects use Git to ensure that software developers and users can always access the most up-to-date version of the software they are working on or using.

Git is installed by default in Ubuntu. You can run:

```
sudo apt-get install git
```

if you want to check that it is installed. If so, you should get a message saying:

```
git is already the newest version
```

If not, it will be installed.

Move to your home directory (/home/USER – USER here stands for the username you set up when you installed Ubuntu; replace it with your actual username) and download Andika!:

```
cd ~
(The tilde is a shortcut for /home/USER)
git clone https://github.com/donnekgit/andika.git
```

After a minute or two, Andika! will be downloaded into an andika folder in /home/USER.

In the future, if you want to update Andika!, you can open a terminal in the andika folder and type:

```
git pull
```

Git will automatically update those parts of Andika! which have been changed.

A/6.3 Move the andika directory

We are going to move the working directory for Andika! to a location which will allow you to access a local copy of the website pages (kevindonnelly.org.uk/swahili) should you wish (see Section A/16 below). The /var/www directory is the default location for storing webpages on your machine.

```
sudo mv andika /var/www/
```

(note the final slash)

Give yourself ownership of the /var/www directory (by default, ordinary users are not allowed to access this):

```
sudo chown -R USER.USER /var/www/
```

Remember, USER here stands for your username, which you set up when you installed Ubuntu. For instance, here I would type:

```
sudo chown -R kevin.kevin /var/www/
```

Set up a link from your /home/USER directory to the /var/www/ directory:

```
ln -s /var/www/ web
```

If you go to your home directory in a file manager (in KDE K -> System -> Dolphin; in Unity, the second icon on the left-hand side, the one with a file on it), you will see an entry for web. If you click this, it will actually take you to /var/www/. This makes access to the webpage directory quicker and easier.

Move into the andika directory for the rest of the installation:

```
cd web/andika
```
You can check the layout of the directories and files by listing them:
ls

A7 Install fonts

A number of default fonts is used in printing the output from Andika!, and these need to be installed.

The most important font, Scheherazade, created by Bob Hallissy and Jonathan Kew, is a Naskh-style font used for the Arabic text. The version available by default from Ubuntu is somewhat old, so it is preferable to use SIL’s own package collection to get the latest package. SIL has a page with instructions on how to do this for the Unity desktop, but as usual it is quicker to use the terminal.

Fetch the authentication key:
wget http://packages.sil.org/sil.gpg

Add the key:
sudo apt-key add sil.gpg

Add the SIL repository to the list of repositories used by your Ubuntu install:
sudo add-apt-repository "deb http://packages.sil.org/ubuntu trusty main"

If you are not using 14.04 (codenamed Trusty Tahir), change trusty accordingly. To find the codename of your version, use:
lsb_release -sc

Update the software package lists to include software from the new repository:
sudo apt-get update

Remove the authentication key:
rm sil.gpg

We can now install all the fonts:
sudo apt-get install fonts-sil-scheherazade fonts-hosny-amiri fonts-liberation

Amiri, created by Khaled Hosny, is an alternate Naskh-style font – it is not currently used in Andika!, but could be a possible alternative to Scheherazade – see Section 4.4.

Liberation Serif in the fonts-liberation package is a tidy font used for the English translation in poetry.

Linux Biolinum O in the fonts-linuxlibertine package is especially good at handling diacritics, so it is a good choice for a close transcription into Roman script.

Use your desktop’s font installation utility to install the GranadaKD font in andika/fonts – this is a Kufic-style font from Arabeyes that has been adapted by me to add the characters necessary for it to be used for Swahili. It is used in Andika! for poem titles.

All of the fonts used by Andika! can be changed – see Section 4.2.

---

13scripts.sil.org/cms/scripts/page.php?item_id=Scheherazade
14packages.sil.org
15amirifont.org
16fedorahosted.org/liberation-fonts
17linuxlibertine.org
18In KDE, K → System → System Settings → Font Management.
19openfontlibrary.org/en/font/granada
A/8  Set up a new language and keyboard

Move the keyboard definition file to the appropriate location.

```
sudo cp layout/tz /usr/share/X11/xkb/symbols/
```

A/8.1  Activate the new keyboard in KDE

Click on K → Settings → System Settings.

In the settings dialogue, click on Input Devices → Keyboard.

On the Layouts tab, tick Configure layouts, and then click Add.

Fill in the pop-up dialogue as shown in Figure A.1.

![Figure A.1: Setting up the Swahili keyboard for KDE](image)

Click OK, and then Apply to exit.

You should now see an additional marker in the system tray at the bottom right of your screen, which will be the abbreviation for the default language on your desktop. For instance, if you have UK English as the default, you will see gb. Click on this, and it will change to tz, showing that the keyboard for Swahili in Arabic script is now operational. You can quickly switch between the two keyboards by pressing Ctrl + Alt + K.

Close the System Settings box.

Note that if you make changes to the keyboard layout, you need to re-apply the layout – see the end of Appendix C.

A/8.2  Activate the new keyboard in Unity

If you have decided to stick to Unity as your desktop, click the System Settings icon in the Launcher, or click the system icon in the top right-hand corner and select System Settings.

Click Text entry in the Personal section.

---

Swahili (Kenya) can be chosen instead of Swahili (Tanzania) if preferred.
Click on the + at the bottom of the left-hand pane, Input sources to use.

Roll down to Swahili (Tanzania).

Click on it and then click Add.

Close the Text entry box.

There should be a new icon on the menu bar. Either Click on the language chooser icon on the menu bar and choose Swahili from the list. Alternatively, press Super (usually the key with a Windows logo on it) + Space.

### A/8.3 Interaction with the unlock screen in KDE

If you have the unlock screen activated, this means that when you leave your machine for some time, it will power down the screen and then, when you resume work, present a login box so that you can unlock the desktop. A problem arises if you changed the keyboard to Swahili before leaving your machine – since the machine was powered down with that keyboard active, the login box will only allow you to type Arabic glyphs, which means that you cannot type in your (Roman glyph) password!

The easiest way to deal with this is to disable the unlock screen by going to K → Settings → System Settings → Power Management → Advanced Settings, and unticking Lock screen on resume.

If for some reason you wish to retain the lock screen, you can recover from this situation whenever it occurs by pressing Ctrl + Alt + F5 to get a terminal login. Type your username and password to log in.

Open the configuration file for the keyboard:

```
nano ~/.kde/share/config/kxkbrc
```

Find the line:

```
LayoutList=gb,tz
```

and change it to:

```
LayoutList=gb
```

Save the file: Ctrl + X, Y, Return.

(The example here uses the UK English keyboard, gb – replace this with whatever your own default keyboard is.)

Press Ctrl + Alt + F7 to return to the unlock screen. Click the login box, and you should be able to login as normal. However, you will need to re-add the Swahili keyboard as shown above.

### A/9 LibreOffice

LibreOffice, installed by default in Ubuntu, is a suite of office software (word processor, spreadsheet, presentation program, etc). The version used here is 4.2.4.2.

#### A/9.1 Configure the word-processor

Open LibreOffice Writer.

Click on Tools → Options → Language Settings → Languages.

Under Default languages for documents, tick Complex text layout (CTL), and select Arabic (Oman) in the dropdown. Click OK.

Click on Tools → Options → Language Settings → Complex Text Layout.
If you wish to use both Arabic-Indic numerals (on the numeral keys) and Western-Arabic numerals (AltGr + numeral), ensure Arabic or System is chosen here. The other two settings will convert Western-Arabic numerals to their Arabic-Indic equivalents.

Tick Visual under Cursor control, and then OK.

Right-click on the toolbar, and under Visible buttons select LTR. Do the same to select RTL. Two new buttons will now appear in the Formatting toolbar, one for left-to-right typing, and one for right-to-left typing.

Shortcuts are Ctrl + Shift + A for LTR and Ctrl + Shift + D for RTL.

A/9.2 Install a template

Andika! includes in andika/libreoffice a template (andika.ott) where styles for Swahili in Arabic script, Swahili in Roman script (standard spelling or close transcription) and translation are already set up – all these styles are right-justified. Installing the template is optional, but it will make typing out Swahili poetry much simpler.

Click on File → Templates → Manage.

On the Documents tab, double-click My Templates and then click Import.

Navigate to andika/libreoffice/andika.ott and click on it – it should now be listed there as a template.

If you want to set it as the default (nothing has been changed from the stock default apart from the addition of the three extra styles), click on it and then click Set as default.

To use the template without setting it as default, select File → New → Templates → andika.

Close the Templates box, and then restart LibreOffice Writer.

To use the styles, place the cursor in the line you wish to format, press F11 to open the Styles and Formatting list, and select the relevant style by double-clicking on its name – the new styles are at the bottom of the list.

Arabic style is RTL, Scheherazade 24pt. You may wish to make the font size smaller. In unvocalised Arabic, reading the text is possible at quite small font sizes. In Swahili, however, the vowel signs are essential, so the same reductions in font size are not possible. In typesetting poetry, the lines are usually short, and accuracy is improved by having a large font size.

Roman style is LTR, right-justified, Liberation Serif 12pt.

Translation style is LTR, right-justified, Liberation Serif 12pt, italic.

Obviously, the appropriate writing system (Arabic or Roman) also has to be selected on the keyboard before typing (see Appendix A/8).

A/10 PHP

PHP is a computer language which is used to convert text from one script to the other, and also for the import and export of text to and from the database.

A/10.1 Install PHP

sudo apt-get install php5 php5-cli
To test the install:
php --version
(note: two dashes)
You should get a message giving details about the version of PHP you've just installed.

A/10.2 Configure PHP

sudo nano /etc/php5/cli/php.ini

This command will open the system file php.ini in a lightweight text editor called nano, where you have to change some settings. Use the arrow keys on the keyboard to move around, and the Home and End keys to move to the beginning or end of a line.

Press Ctrl+W, then type
max_execution_time
into the searchline and press Return. Change the line to read:
max_execution_time = 300

Again press Ctrl+W, type
error_reporting
and press Return. Change that line to:
environment = E_ALL & ~E_NOTICE & ~E_DEPRECATED

A bit lower down from that (you can scroll down using the mouse), there is a display_errors line. Change it to read:
display_errors = On

Below that there is a log_errors line. Change it to read:
log_errors = Off

To save the file, press Ctrl+X, then press Y to confirm you want to save the modifications, and press Return to close the file.

A/11 PostgreSQL

PostgreSQL is a database which is used to store the words of the text for editing and enhancement.

A/11.1 Install PostgreSQL

sudo apt-get install postgresql postgresql-client postgresql-common

To test the install:
psql

You should get an error message saying that the role named after your username does not exist.

A/11.2 Set up a database user

On Ubuntu, PostgreSQL uses peer authentication by default. This means that creating a database user with the same name as your system (Ubuntu) user will mean you can log in to the database without
entering a password. The terminal prompt should tell you what your username is – it is of the form `user@computer`. Alternatively, you can run:

```
whoami
```
to find out your username.

```
sudo -i
```
The prompt will change to show that you are now root (the superuser, or administrator).

```
su - postgres
```
(note the space on either side of the dash)
The prompt will change to show that you are now the postgres master user.

Create a new database user with the same name as your system user (replace USER with your system username):

```
createuser -P -s -e USER
```
You will be asked to enter a password – note that you will get no feedback (the line will stay blank). Press `Return` and you will be asked to enter the password again. Press `Return` and you should get a message beginning `CREATE ROLE`, meaning that the new user has been created.

```
exit
to cease being the postgres user.
```
```
exit
to cease being the superuser.
```

**A/11.3 Set Andika! to use your database user**

```
nano andika/config.php
```
Change:
```
user=kevin password=kevindbs
```
to read:

```
user=USER password=yourpassword
```
and save the file (`Ctrl + X, Y, Return`).

Remember to replace USER with your username.

**A/11.4 Create the andika database**

```
createdb andika
```
This creates the *andika* database, owned by your new user.

Andika! comes with starter data in *andika/db/starter/andika.sql*, which can be imported into the new database:

```
psql -d andika < db/starter/andika.sql
```
If you chose a username other than *dbmaster*, use that instead.

---

21 If for security reasons you wish to enter a password each time your user accesses a database, open the configuration file: `sudo nano /etc/postgresql/9.3/main/pg_hba.conf`. Find the line: `local all all peer` and change it to read `local all all md5`. Save the file: `Ctrl + X, Y, Return`. Restart PostgreSQL: `sudo service postgresql restart`. You will now need to enter your database password even to connect to the database under your system username.
APPENDIX A. INSTALLING ANDIKA!

A/11.5 Connect to the andika database

```
psql -d andika
```

The prompt should change to `andika=#.`

\dt

(= display tables)

This should show a list with 18 rows, each representing a database table holding poem information in the `andika` database. To look at the table for the poem `kiswahili`:

```
select * from kiswahili;
```

(note: the semicolon at the end is an integral part of the command)

This will show everything in the `kiswahili` table. Exit the data display and go back to psql:

```
qu
```

To see something more selective:

```
select * from kiswahili where stanza=1 order by stanza, loc;
```

(again, remember the semicolon at the end)

You should get a listing of the `vipande` in the first stanza of the poem `Kiswahili` from the Abdulkadir and Franklp paper, in order of `kipande`. Exit the data display and go back to psql:

```
qu
```

to exit `psql`.

A/12 Database interfaces

To make it easier to read and edit the contents of the PostgreSQL database, it is best to install an interface. Two of these will be installed, each differing in their capabilities. The first is a web-based interface called phpPgAdmin, which first requires a webserver (Apache) to be installed. The second interface is called SQL Workbench, and it requires a computing language called Java to be installed.

A/12.1 phpPgAdmin

Install Apache

```
sudo apt-get install apache2 apache2-utils phpgadmin
```

Start the webserver:

```
sudo service apache2 start
```

If you want to get rid of the (harmless) message `Could not reliably determine the server’s fully qualified domain name, using 127.0.1.1. Set the ‘ServerName’ directive globally to suppress this message`, issue the following commands:

```
echo "ServerName localhost" | sudo tee /etc/apache2/conf-available/servername.conf
```

```
sudo a2enconf servername
```

```
sudo service apache2 restart
```

Test the install – open a web browser (preferably Firefox) and type:

```
http://localhost
```

into the address bar. A page should open, telling you that Apache is installed and working.
Configure phpPgAdmin

Activate the phpPgAdmin configuration file:
sudo cp /etc/apache2/conf.d/phppgadmin /etc/apache2/conf-enabled/phppgadmin.conf

Restart the webserver:
sudo service apache2 restart

In the web browser, type http://localhost/phppgadmin into the address bar.

You should see the phpPgAdmin homepage. On the left side there is a list of servers (in this case, there should only be one listed). Click on PostgreSQL and you should get a login form.

Fill in the username and password for PostgreSQL (which you created in Appendix A/11.2) and click Login.

The default session time for PHP is set to 24 minutes (1440 seconds). This means that if you do not use phpPgAdmin for 24 minutes, it will ask you to log in again before you can continue using it. If you find that this interrupts your workflow, you can change the setting in the PHP configuration file:
sudo nano /etc/php5/apache2/php.ini

Press Ctrl+W, then type:
session.gc_maxlifetime

and press Return. Change the line to read:
session.gc_maxlifetime = 144000

This will allow you 40 hours before logging you out, which should be sufficient.

Test phpPgAdmin

In the left-hand panel you should get a list of your current databases – there should only be two: andika and the system database postgres. (The right-hand panel shows you the same two databases.)

Click the + beside andika in the left-hand panel. It should open to show Schemas, public, Tables etc. Click on Tables. The right-hand panel should now show you all the tables inside the andika database, similar to what you saw in Section A/11.5.

Click on kiswahili and you will see the data fields in that table. To see the contents of the database you can click on the Browse button.

To see the data fields of each item in more detail, click on the Edit button beside each row in the table – changes to the fields can be made and saved here.

To make a database query, click the SQL link at the top right of the phpPgAdmin window. This will open another smaller window. In the large textbox, type:
select * from kiswahili where stanza=1 order by stanza, loc;
(again, remember the semicolon at the end)

Click Execute, and in the first window you should see a listing of the vipande in the first stanza of the poem, in order of kipande, as you did in Section A/11.5. In this case, though, the contents are a lot easier to read!
A/12.2 SQL Workbench

Install Java

Andrei Alin\(^{22}\) maintains links to up-to-date versions of Java in his software repository.

Check that the helper script `add-apt-repository` is installed:
```
sudo apt-get install software-properties-common
```

Add the new software repository:
```
sudo add-apt-repository ppa:webupd8team/java
```

Update the software package lists to include software from the new repository:
```
sudo apt-get update
```

Install the Java installer:
```
sudo apt-get install oracle-java8-installer
```

This installs a script that then downloads and installs Oracle Java 8 – it may therefore take a few minutes. To test the install:
```
java -version
```
This should return some text telling you that the Java version is 1.8.0.

Set the Java environment variables:
```
sudo apt-get install oracle-java8-set-default
```

Install JDBC

JDBC (Java Database Connector) is a driver which will allow SQL Workbench to connect to the PostgreSQL database.

```
sudo apt-get install libpostgresql-jdbc-java
```

Install SQL Workbench

Create a directory to hold the files:
```
mkdir sqlworkbench
```

Go to the website `sql-workbench.net`, click on the link for Build 116 (or whatever the current stable version is), and download the generic package. Save it in the `andika` directory.

Unzip the download into the new directory:
```
unzip -q Workbench-Build116.zip -d sqlworkbench
```

Make the launch script executable:
```
chmod +x sqlworkbench/sqlworkbench.sh
```

Launch SQL Workbench:
```
sqlworkbench/sqlworkbench.sh
```

If you wish, you can make a desktop shortcut or menu entry to make launching SQL Workbench easier.

Configure SQL Workbench

A Select Connection Profile box should come up.

\(^{22}\)webupd8.org
Change New profile to read andika.

Click the drop-down arrow on the Driver line and select PostgreSQL. Click Yes, when you’re asked whether you want to edit the driver definition.

On the Manage Drivers popup, click on the red postgresql entry already there and then click X to delete it. Click on the folder icon and navigate to /usr/share/java/postgresql-jdbc4-9.2.jar (which you installed in Section A/12.2. Click Open, and then OK.

Check that the URL line reads jdbc:postgresql://localhost:5432/andika – if not, edit it to make it so. Enter your PostgreSQL username and password (Section A/11.2), and then click OK. You should get a connecting message.

**Test SQL Workbench**

The main screen consists of a top pane where you type database queries, and a bottom pane where the results will appear.

In the top pane, type:

```sql
select * from kiswahili where stanza=1 order by stanza, loc;
```

(remember: the semicolon at the end is an integral part of the command)

Move the cursor somewhere in the middle of that query and press Ctrl + Return. In the bottom pane, you should see a listing of the vipande in the first stanza of the poem, in order of kipande, as you did in Sections A/11.5 and A/12.1.

The main benefit of SQL Workbench compared to phpPgAdmin is that a result set can be directly edited – this makes it easy to add data. To try this, select one of the cells under the english column, type something in, and press Return. A yellow diamond will appear in the leftmost column, showing that the record has been edited but not saved yet. To save it you need to click the disk icon, or select Data → Save changes to database and click OK when asked to confirm.

You can check the change was made by running the same query in phpPgAdmin’s SQL box (see Section A/12.1):

```sql
select * from kiswahili where stanza=1 order by stanza, loc;
```

To delete the change, click the cell in SQL Workbench, press Backspace and then Return, and then save as before.

Close SQL Workbench, clicking Yes to save the new andika connection profile you have set up.

**A/13 LaTeX**

LaTeX is a typesetting system that is capable of creating very complex layouts. It is used in Andika! to provide attractive output.

```bash
sudo apt-get install texlive texlive-xetex texlive-generic-extra texlive-humanities texlive-lang-arabic texlive-latex-extra texlive-bibtex-extra kile kbibtex biber
```

Note that these packages will take perhaps 20 minutes to download and install.

**A/14 JabRef**

JabRef is a bibliography manager.

---

23jabref.org
suo apt-get install jabref

A/15 YAD

YAD (Yet Another Dialogue), maintained by Victor Ananjevsky, is used by Andika! to provide a point-and-click interface to the conversion script. To install it, we need to add another of Andrei Alin’s repositories.

Add the new software repository:
```
sudo add-apt-repository ppa:webupd8team/y-ppa-manager
```

Update the software package lists to include software from the new repository:
```
sudo apt-get update
```

Install YAD:
```
sudo apt-get install yad
```

A/16 Access the Andika! website locally

Although not essential to use Andika!, it may be useful to have access to a local copy of the website (kevindonnelly.org.uk/swahili).

First, tell the webserver installed earlier (Apache – see Section A/12.1) where to find the Andika! webpages.

Open a configuration file:
```
sudo nano /etc/apache2/sites-available/andika.conf
```

Type the following lines into the file:

```
<VirtualHost *:80>
    ServerName andika
    DocumentRoot /var/www/andika/
</VirtualHost>
```

Save and exit the configuration file:

```
Ctrl+X, Y, Return
```

Activate the configuration:
```
sudo a2ensite andika
```

Restart the webserver:
```
sudo service apache2 restart
```

Then tell your web browser that the new website is on your machine, so it doesn’t have to look for it on the web.

Open a configuration file:
```
sudo nano /etc/hosts
```

After the line:
```
127.0.0.1 localhost
```
```
sourceforge.net/projects/yad-dialog/
webupd8.org
You may also wish to install Andrei’s own Y-PPA-Manager – sudo apt-get install y-ppa-manager. This is not used by Andika!, but is a very useful system tool to keep track of the software repositories on your machine.
```
APPENDIX A. INSTALLING ANDIKA!

add the following line:
127.0.0.1 andika

Save and exit the configuration file:
Ctrl + X, Y, Return

In a web browser, type:
http://andika/index.php
into the address bar. You should get the Andika! website loading from the files on your hard disk (in /var/www/andika), instead of from the internet.
Appendix B

Editing fonts

I am grateful to Khaled Hosny for his advice on using FontForge to edit Arabic glyphs, which has been incorporated in these instructions.

B/1 Introduction

Most Arabic fonts are missing some glyphs that are essential to allow them to be used for writing Swahili. This appendix deals with how to edit these fonts to add the missing glyphs. This will entail editing the font with FontForge (originally developed by George Williams). A version of this howto with screenshots is available at the website for the book Design with FontForge.

B/2 Install FontForge

There are two options here – the easiest is to use a pre-compiled package.

B/2.1 Use a pre-compiled package

The FontForge package included in Ubuntu 14.04 by default is too old, so it is preferable to install the more up-to-date package from the FontForge Personal Package Archive (PPA).

Check that the helper script add-apt-repository is installed:

```
sudo apt-get install software-properties-common
```

Add the FontForge PPA (which will also add the authentication key):

```
sudo add-apt-repository ppa:fontforge/fontforge
```

Update the package list:

```
sudo apt-get update
```

Install FontForge:

```
sudo apt-get install fontforge
```

B/2.2 Compile from the source code

Installing the pre-compiled package as above will be sufficient for most purposes, so you do not usually need to do anything else. In some cases, though, (perhaps because you want access to a feature not

---

1 khaledhosny.org
2 fontforge.github.io/en-US
3 designwithfontforge.com/en-US/Adding_Glyphs_to_an_Arabic_Font.html
4 https://launchpad.net/~fontforge/+archive/ubuntu/fontforge
yet available in the pre-compiled package) you may wish to compile your own version from the code available on GitHub.\(^5\)

**Install preliminary software**

Install packages to allow the building of software:

```
sudo apt-get install build-essential automake flex bison
```

Install the *unifont* package to get a full display of the reference glyphs. Unifont\(^6\) includes glyphs for all Unicode codepoints, and FontForge will use it if it is installed.

```
sudo apt-get install unifont
```

Install other required packages:

```
sudo apt-get install packaging-dev pkg-config python-dev libpango1.0-dev libglib2.0-dev libxml2-dev giflib-dbg libjpeg-dev libtiff-dev uthash-dev
```

**Build libspiro**

FontForge uses *libspiro*\(^7\) (by Raph Levien) to simplify the drawing of curves.

Download the code:

```
git clone https://github.com/fontforge/libspiro.git
```

Run the following commands in sequence (that is, wait for each one to complete before running the next):

```
cd libspiro
autoreconf -i
automake --foreign --Wall
./configure
make
sudo make install
cd ..
```

**Build libuninameslist**

FontForge uses *libuninameslist*\(^8\) to access attribute data about each Unicode code point.

Download the code:

```
git clone https://github.com/fontforge/libuninameslist.git
```

Run the following commands in sequence (that is, wait for each one to complete before running the next):

```
cd libuninameslist
autoreconf -i
automake --foreign
```
BUILD FONTFORGE

Download the code:

```
git clone https://github.com/fontforge/fontforge.git
```

Run the following commands in sequence (that is, wait for each one to complete before running the next):

```
cd fontforge
./bootstrap
./configure
make
sudo make install
cd ..
```

Makethesystemawareofthenewlibraries:

```
sudo ldconfig
```

B/3 Make a working copy of the font

The font we will add glyphs to is Graph9 (regenerated by Nadim Shaikli). A version of the following about which includes images is available on the Design with FontForge website.10

Download the font from the webpage into the andika directory. Unzip it, and delete the zip file:

```
unzip -q graph.zip -d fonts && rm graph.zip
```

Launch FontForge (in KDE, go to K → Graphics → FontForge). Note that FontForge is built using the programming language Tcl,11 and it therefore behaves slightly differently from other software you may be used to. For instance, every action requires at least one click (so the submenus for menus don’t appear as you move across the menu bar – you have to click each one).

The first time you open FontForge, it will ask you to load a font. Navigate to andika/fonts, select ae_Graph.ttf, and click OK. FontForge will display a chart of every glyph in the font, each in its own cell. The smaller cell above it is a reference glyph – not all reference glyphs will have a font glyph, since few fonts contain glyphs for every single Unicode code point. Where the font glyph is missing, the cell will contain a grey X.

Save it as an sfd file which will become your working copy: select File → Save, edit the suggested name to read GraphSwa.sfd and click Save.

B/4 Rename the font

If you do not rename the font, your adapted font will not install separately from the original – you will have to uninstall the original font first. It is also sensible to rename the font if you are going to distribute

---

9openfontlibrary.org/en/font/graph
11tcl.tk/
your adaptations – if the original author of the font has reserved the font name under the Reserved Font Name (RFN) mechanism, that original name can only be used with the original author’s version of the font.

If you adapt a font that was originally under an open license (e.g., GPL\textsuperscript{12} or OFL\textsuperscript{13}) and then distribute it, you must retain the original author’s copyright notices and licensing information, although you can append a note at the end of the copyright notice covering your contribution.

Note that adapting a font that was originally under a closed license (e.g., most fonts by Microsoft, Adobe, Bitstream, Linotype, etc.), may be a breach of copyright, depending on the terms of the license.

Select **Element → Font Info**, and in the **PS Names** panel, change **Fontname**, **Family Name**, and **Name For Humans** to **GraphSwa**.

In the **TTF Names** panel, the names for **Family** and **Fullname** are taken from the **PS Names** entries, and should already be showing **GraphSwa** (you can’t edit them directly). Change the entries for **Preferred Family** and **Compatible Full** to **GraphSwa**. These name changes will now allow you to install this font alongside the original one if you wish.

If desired, in the **TTF Names** panel you can also place a "Swahili glyphs added by" message after the text already in the entry for **Designer**.

Click **OK** to save these changes. You will get a message about generating a new UniqueID (XUID) for the font – click **Change**.

**B/5 Add the glyph for the isolated form of peh**

We will add the missing glyph peh (U + 067E) to the Graph font.

Go to the Arabic section of the font chart: select **View → Go to**, click the dropdown box and select **Arabic**, then click **OK**.

Clicking on a cell in the font chart will show its Unicode number and name in blue at the top of the panel. Go to position 1662 (0x67e) “uni067E” ARABIC LETTER PEH. The cell below the reference glyph contains a grey X, showing that the font does not include this glyph.

We will make peh by copying beh (U + 0628) and swapping its single dot for three dots.

Click on the beh cell (position 1576), then right-click and select **Copy**. Then right-click on the peh cell and select **Paste**. Now that beh is now copied into the peh cell, the next thing is to change the dot.

Find a glyph with three dots – sheen (position 1588, U + 0634) will do. Double-click on the cell – this will open a glyph design panel. Press V to ensure the pointer tool (arrowhead) in the toolbox is selected, and press Z and enlarge the panel to give you a good view of the glyph.

Click and drag so that the nodes of the three dots above sheen change colour from pink to beige. If you accidentally include or omit a node, deselect or select it by pressing **Shift** and clicking. Press **Alt + C** to copy.

Go back to the font chart and double-click on the peh cell – this will load peh into another tab in the glyph design panel, alongside the sheen tab.

Click and drag to highlight the dot below peh, then press **Delete**. Press **Alt + V** to paste in the three dots, which will likely appear above the body of peh. Leave the dot nodes highlighted so that you can invert and move them more easily.

Invert the dots: select the flip tool (two triangles with a red dashed line between them) from the toolbox.

\textsuperscript{12}gnu.org/copyleft/gpl.html
\textsuperscript{13}scripts.sil.org/OFL-FAQ_web
(Alternatively, right-click in the middle of the dots, and select **Flip the selection** from the popup.) Click on one of the dot nodes and drag the mouse slightly left or right.

Move the inverted dots: press **V** to select the pointer tool again, click on one of the dot nodes, and drag them down below the body of the glyph. Position them centrally, above the *Arabic Below* mark.

Close the glyph design panel. There should now be a new glyph for *peh* in the font chart. Save the adapted font (**File → Save**).

**B/6  Add the glyphs for the connected forms of peh**

However, this is only the isolated (standalone) form of the glyph. If you try to use your adapted font, you will find that initial, medial and final forms are not available. These have to be created separately.

"The[se] forms are built as unencoded glyphs (glyphs whose encoding is -1 in FontForge conventions). Th[ey] have no predefined slots.” (Khaled Hosny)

Select **Encoding → Add Encoding Slots** and enter the number of the glyphs you want – in this case 3. FontForge will add the same number of slots at the very end of the font, and you will be moved there in the font chart. The last three cells (positions 65537, 65538, 65539) have a question mark as a reference glyph, and it is in those cells that you will add the unencoded glyphs by repeating the process in Appendix B/5 above.

Note that if by mistake you start typing when the font chart still has focus, you get moved to the European section at the top. To get back to the bottom, select **View → Go to**, click the dropdown box and select **Not a Unicode Character**, and then click **OK**.

**B/6.1 Create the final form**

Roll the font chart up a bit until you come to a set of Arabic glyphs at position 65152 (U+FE80) onwards. At U+FE90 (position 65168) you will see a *behfinal* glyph – click on it and press **Ctrl+C** to copy it. Roll down to the third last cell in the chart (position 65537), click on it, and press **Ctrl-V** to paste in the *behfinal* glyph.

Right-click on the cell and select **Glyph Info**. The naming convention is to use the number of the isolated glyph + a suffix for the form, so change **Glyph Name** to uni067E.fina, and click **OK**. The question mark in the reference cell will change to *peh*.

Get the three dots: double-click on *sheen* (U+FEB5) to load it into the glyph design panel, select the three dots and press **Ctrl+C**.

Double-click on the new *pehfinal* to load it into the glyph design panel, click and drag to highlight the nodes of the dot and press **Delete**.

Ctrl+V to insert the three dots from *sheen*, flip them, and move them into position below the glyph body. Press **Ctrl+S** to save the revised font chart.

**B/6.2 Create the initial and medial forms**

Copy the initial form U+FE91 (position 65169) to the penultimate cell (position 65538), delete the single dot and paste in the three dots.

Right-click the cell, select **Glyph Info**, change **Glyph Name** to uni067E.init, and click **OK**.

Copy the medial form U+FE92 (position 65170) to the last cell (position 65539), delete the single dot and paste in the three dots.
Right-click the cell, select **Glyph Info**, change **Glyph Name** to **uni067E.medi**, and click **OK**.

Select **File → Save** to save the revised font chart.

### B/6.3 Add the lookups

The isolated form has to be mapped (linked) to its initial, medial and final forms.

Select **Element → Font Info → Lookups**.

Click on the + beside the entry *'init' Initial Forms in Arabic lookup 2*. This will open a submenu of the same name. Click on this submenu.

The **Edit Data** button on the right will now become available – click it.

In the **Lookup Subtable** panel that pops up, ensure that the **Unicode** button is checked. Roll the list of characters down until you come to the end.

In the box beside **Default Using Suffix**, enter the relevant suffix (in this case, *init*), and then click **Default Using Suffix**.

A new mapping will be added to the list of characters, from uni067E (the isolated form of *peh*) to uni067E.init (the initial form). Click **OK**.

Do the same for the submenus under the entries *'medi' Medial Forms in Arabic lookup 2* and *'fina' Terminal Forms in Arabic lookup 2*, choosing *medi* and *fina* as the relevant suffix.

Click **OK** again to close the panel, and save the font chart (**Ctrl + S**).

Note that **Default Using Suffix** only seems to work on glyphs in the Unicode 06 (*Arabic*) block – glyphs in Unicode 07 (*Arabic Supplement*), eg *ain* with two dots, may have to be added manually by clicking the line marked **New** and typing in the names.

### B/7 Generate the adapted font

Select **File → Generate Fonts**.

In the dropdown showing **PS Type 1 (Binary)**, select **TrueType**, and check that the filename reads **Graph-Swa.ttf**.

Navigate to where you want to save the font, and then click **Generate**. Click **Yes** and **Generate** to the two information messages that come up. You can then use your normal font installation procedure (in **KDE, K → System → System Settings → Font Management**) to install the adapted font.

### B/8 Next steps

You will need to carry out the above process to add all the missing glyphs listed in Table 4.2.

Note that if you make changes to a font, you need to restart LibreOffice in order to use the changed font, because it will see only the previous version of the font, and not the new changes.
Appendix C

Changing the Andika! keyboard layout

C/1 Introduction

The layout of the Andika! keyboard is specified in the file layout/tz. The file (reproduced in Appendix E) is a simple text file, and can be easily adapted to add new glyphs or change the position of existing glyphs.

Each line follows the pattern below:

```plaintext
key <AC03> { [Arabic_dal, Arabic_thal, Arabic_dad, Arabic_ddal] };
```

The key number (in this case AC03, for the D key) is followed by a sequence of 4 glyph names (in this case representing أضذد). The sequence specifies the glyph that will be output when (respectively) the user presses D, Shift+D, AltGr+D, and AltGr+Shift+D.

Some lines have less than four entries. For instance, the P key only has one entry (پ):

```plaintext
key <AD10> { [Arabic_peh] };
```

because that is the only glyph output by that key, and the S key only has three entries (صشس):

```plaintext
key <AC02> { [Arabic_seen, Arabic_sheen, Arabic_sad] }
```

giving the glyphs that will be output by pressing S, Shift+S and AltGr+S.

If it is desired to block one of the slots, to enforce a particular keypress for a glyph, the entry NoSymbol can be used. Thus in the line for the 5 key:

```plaintext
key <AE05> { [Arabic_5, NoSymbol, KP_5, percent] };
```

the output will be ٥ for 5, nothing for Shift+5, Western 5 for AltGr+5 and a percent sign for AltGr+Shift+5. Without the NoSymbol, the output would be ٥ for 5, Western 5 for Shift+5, a percent sign for AltGr+5 and nothing for AltGr+Shift+5.

Glyph names are available for some, but by no means all, of the possible glyphs. Where no name is available, the Unicode codepoint can be used instead. Thus, in the line for the n key:

```plaintext
key <AB06> { [Arabic_noon, U075D] };
```

ن will be output when the N key is pressed, and the glyph represented by Unicode 075D (٣, ain with two dots above) will be output when Shift+N is pressed. It would be possible to use nothing but Unicode codepoints in the file, but using the glyph names makes it a bit easier to read.

From the above, it will be obvious that adjusting the location of a particular glyph merely consists of moving it to the desired slot on the desired key. For example, if the user wanted ص to appear when Shift+D is pressed, and ذ when AltGr+D is pressed, all that needs to be done is to open the file:

```
sudo nano layout/tz
```

and change the line:

```plaintext
key <AC03> { [Arabic_dal, Arabic_thal, Arabic_dad, Arabic_ddal] };
```

---

APPENDIX C. CHANGING THE **ANDIKA!** KEYBOARD LAYOUT

...to:

```plaintext
key <AC03> { [Arabic_dal, Arabic_dad, Arabic_thal, Arabic_ddal] };
```

Then save the file by pressing **Ctrl+X**, **Y**, and **Return**.

Likewise, adding a new glyph to the keyboard is as simple as deciding which slot on which key it should occupy, and then inserting the Unicode codepoint (or the glyph name where one exists) at that slot. For instance, if the user needs to access the glyph *rreh* (*ra* with *tah* as a diacritic, Unicode 0691), and decides to put it on the **R** key so that it will be output when **AltGr+R** is pressed, all that needs to be done is to change the line:

```plaintext
key <AD04> { [Arabic_ra] };
```

to:

```plaintext
key <AD04> { [Arabic_ra, NoSymbol, U0691] };
```

(Remember that if **NoSymbol** is omitted here, *rreh* will appear when **Shift+R** is pressed.)

In either case, the new layout has to be activated. So, after saving the file, copy it to the correct location:

```
sudo cp layout/tz /usr/share/X11/xkb/symbols/
```

Delete the cache files relating to the old layout (new ones will be created when the new layout is activated):

```
sudo rm /var/lib/xkb/server-*/
```

Then remove the **tz** keyboard layout using your desktop's language setup utility and re-add it. For KDE, this simply means going to **K → Settings → System Settings → Input Devices → Keyboard → Layouts** tab, unticking **Configure layouts**, clicking **Apply**, and then reticking **Configure layouts** and clicking **Apply** again.

The new layout should then be ready for use.
Appendix D

*Annotated poem*, كِسْوَاحِلٍ (Kiswahili), 12pt, footnotes


The following is a letter-for-letter transcription of the author's manuscript as reproduced there, with the exception that the damma-with-tail occasionally used by him to signify o is denoted here with inverted damma (eg in كُوَأٗنَنَتَمَانِ in 1d), since the font does not yet include that glyph. The layout also includes an automatically-generated close transliterations, a standard transliteration based on that in the paper, and the English translation and notes from the paper. The notes also include citations, with the references printed at the end of the document.

The document was generated automatically from a database table which held all the data about the poem (words, translation, notes, etc) – see Chapter 8, and the kiswahili table in the andika/db/starter/andika database included in the **Andika!** download.

The font-size is 12pt, and notes appear as footnotes. For comparison, the same poem is printed in Appendix F in a font-size of 10pt, with notes appearing as endnotes.
Mtungaji: Ustadh Mau (Mahmoud Ahmad Abdulkadir), 2003

بسم الله الرحمن الرحيم
bismi llāhi arrahmanī arrahīmī

I am weary of staying silent. For how much longer am I to remain dumb?

My own children avoid me, though I long to see them.

And those who remain to embrace me are not my own, but are the offspring of others.

What have I done to you? Why do you wage war on me?

My own flesh and blood, the children of Swahililand,

are uninterested in knowing who I am,
What kind of fault is my fault? [O my children] why do you continue waging war on me?

I am your mother and am not yet infertile, nor has my ability to reproduce diminished.

I have given birth to children in Mambasa, and in the other islands [of the Swahili],

to politicians and to religious leaders,

to craftsmen in every field, and to war heroes.

I am the mother of Bwana Muyaka, and of Mwengo Athuman also,

---

2Bwana Muyaka was the outstanding Swahili poet of 19th century Mombasa. After his death many of his verses were recalled by Mu'allim Sikujua Abdallah al-Batawi (died 1890) and transcribed with annotations by W.E. Taylor (1856-1927). After Taylor's death his papers were acquired by the library of the School of Oriental and African Studies (SOAS), London.

3Mwengo Athmani: this 18th century poet from Pate composed the Utendi wa Tambuka (The Epic of Heraklios).
APPENDIX D. ANNOTATED POEM, (KISWAHILI), 12PT, FOOTNOTES

Who were those who composed them? They were my children who have passed on.

---

Footnotes:

5 Ali Koti of Pate: see S. Chiraghdin (1987, pp.31-7).
6 Bwana Mataka’s full name is Muhammad bin Shee Mataka al-Famau (1825-1868). He was ruler of Siyu, as was his father. His mother was Mwana Kupona, famous for the poem of advice written to her daughter. Bwana Mataka died in Mombasa’s fort while imprisoned by the Busa’idi.
7 The Inkishafi, according to W.E. Taylor (Stigand 1915, pp96-105), is “a great, if not the greatest, religious classic of [the Swahili-speaking peoples]”. The poem, concerned with the decay of Pate (formerly a flourishing town in northern Swahililand), may remind some readers of Thomas Gray’s Elegy written in an English churchyard (London 1751).
And the Bard of Mambasa, and Chiraghdin too,

they followed in my footsteps, they did not submit to lower standards.

He remains alone in the field, yet he stays strong.

I am still able to give birth. I have not yet reached the limit,

but you have all despised me. You have left me high and dry,

---

8The Bard of Mambasa refers to Ustadh Ahmad Nassir Juma Bhalo, see S. Chiraghdin (1971).
10In an unpublished commendation from 12 June 1974 J.W.T. Allen writes about Ahmad Sheikh Nabhany: “I am privileged to have a wide circle of friends and acquaintances among Swahili scholars of Swahili. I have some knowledge of their rating of themselves and I can name perhaps half a dozen (still living) who are always referred to as the most learned. To me they are walking dictionaries and mines of information and Ahmed is unquestionably one of them. He comes of a family of scholars whose discipline is as tough as any degree course in the world. They have no time for false scholarship or dilettantism. That this profound learning is almost wholly disregarded by those who have been highly educated in the western tradition affects almost everything written today in or about Swahili. When I want to know some word or something about Swahili, I do not go to professors, but to one of the bingwa known to me. One of these could give a much greater detail of assessment, but of course his opinion would not carry the weight of one who can put some totally irrelevant letters after his name”. For a biography see Said (2012).
For almost a century the principal publisher of standardized Swahili dictionaries has been the Oxford University Press (OUP). Clearly OUP has to be profitable, and profitable is what, over the years, their dictionaries of standardized Swahili have been. However, if one considers excellence in research and scholarship not one of the OUP’s standardized Swahili lexicons can begin to compare with the Oxford English Dictionary (‘more than 600,000 words over a thousand years’). Fortunately for Swahili and for Swahili studies there exists the monumental Dictionnaire swahili-français, compiled by Charles Sacleux (Sacleux 1939). Sacleux’s chef d’oeuvre (‘unprecedented in historical depth, dialectological detail and philological knowledge’) can now be accessed electronically, courtesy of Swahili Forum (uni-leipzig.de/~afrika/swafo/index.php/sacleux). Heartfelt thanks are due to Thilo Schadeberg and Ridder Samsom.
They are written neither by Rajabu, nor by Sudi nor by Shani.

The author is Njoroge, he is the helmsman.

Charo and his colleagues follow.

I feel exceedingly bitter that I do not see you all there.

My children, you have missed your opportunity. You have abandoned your own mother.
Students from Kibwezi, and from Kisumu by the lake,

they are the ones who are ahead, who are at the top;

and you, students from the coast, you lag far behind.

Amongst those who are researching for degrees at the universities,

Swahili students are few or non-existent.

Who is to be blamed? Whose fault is it?

You esteem me not at all, yet you have not replaced me by another.

---

15 Kibwezi and Kisumu are places in the East African interior.
16 The lake is Lake Nyanza, also known as Lake Victoria.
17 Over the years young people on Lamu Island (and indeed elsewhere in northern Swahililand) have received a raw deal in their primary and secondary education. They have ‘lagged far behind’ their counterparts from the interior, and so Mother Swahili grieves for her marginalised children.
When I hear those who are not mother-tongue speakers speaking, I feel sick at heart.

Inflection is no longer employed, while grammatical [Swahili] is what I desire!

Even [their speech] is wanting in flavour, like a plug of tobacco in one's mouth.

I do not understand what they are saying. Are they singing? Are they complaining?

Were Bwana Muyaka to return, were he to come back to the world, it would be necessary, my child, for him to go to a court of law, and he would need to call witnesses who know me well, and all of you would go to prison for the offence which you have committed against me.

APPENDIX D. ANNOTATED POEM, (KISWAHILI), 12PT, FOOTNOTES
15 Truly you have neither zeal nor self-confidence.

15 It irritates you not at all that you do not esteem me.

15 I am just like a ball in the play-ground,

15 I am given a kick by anyone who passes by in the street.

16 Even in the field of Swahili prosody, those who are not mine have invented free verse, imitating foreigners.

16 For myself, I cannot accept that. That is not Swahili poetry.

16 What is the point of it all? These are preparations for war.
I am told that I belong to nobody in particular. How extraordinary!

How can I be rootless below ground and yet have branches above?

Who gave me my name? And who are they who wrote me down?

If I do not hail from Swahililand, then whence do I come?

That many speak me, [Swahili], is not of itself proof of origins, or that I have no owner. What of the English language?

It is spoken by very many, in all corners of the world, yet the language remains firmly established in its homeland, its roots have not been severed.
References

Appendix E

The keyboard layout file (layout/tz)

This appendix contains the contents of the Andika! file layout/tz. Lines which begin with // are comment lines, intended to explain which glyphs will be output when a particular key is pressed. For more information, see Section 5.3.

// Keyboard layout for Swahili in Arabic script.
// This file is part of the Andika! project, and is licensed under GPLv3 or later.
// Version 2014-08-12
// Andika! -- kevindonnelly.org.uk/swahili
// Kevin Donnelly (kevin@dotmon.com)

xkb_symbols "swa"
{
  name[Group1] = "Swahili";

  include "level3(ralt_switch)"

  // 1=key, 2=Shift+key, 3=AltGr+key, 4=AltGr+Shift+key

  // ------------
  // ZXCV row
  // ------------
  key <LSGT> { [Arabic_superscript_alef, Arabic_maddaonalef, Arabic_hamzaunderalef, U0671] };
  // 1 superscript alef, 2 alef with madda above, 3 alef with hamza below (vowelcarrier), 4 alef wasla
  key <AB01> { [Arabic_zain, Arabic_jeh, Arabic_zah] };
  // 1 zain (z), 2 jeh (zh), 3 zah (zw)
  key <AB02> { [Arabic_khah] };
  // 1 khah (kh)
  key <AB03> { [Arabic_tcheh, U063B, U06AE] };
  // 1 tcheh (ch), 2 keheh with two dots above (kj), 3 kaf with three dots below (kj)
  key <AB04> { [Arabic_veh] };
  // 1 veh (v)
  key <AB05> { [Arabic_beh] };
  // 1 beh (b)
  key <AB06> { [Arabic_noon, U075D] };
  // 1 noon (n), 2 ain with two dots above (g in ng')
  key <AB07> { [Arabic_meem] };
  // 1 meem (m)
  key <AB08> { [Arabic_comma, Arabic_hamza_above, comma, leftcaret] };
  // 1 comma, 2 hamza as diacritic, 3 UK comma, 4 closing angle bracket
  key <AB09> { [Arabic_fullstop, Arabic_sukun, period, rightcaret] };
  // 1 fullstop, 2 sukun, 3 UK fullstop, 4 opening angle bracket
  key <AB10> { [Arabic_question_mark, NoSymbol, KP_Divide, question] };
  // 1 question mark, 3 forward slash, 4 UK question mark

  // ------------
  // ASDF row
APPENDIX E. THE KEYBOARD LAYOUT FILE (LAYOUT/TZ)

// -----------------
key <AC01> { [Arabic_fatha, Arabic_alef, Arabic_hamzaonalef, Arabic_fathatan] };
// 1 fatha (short a), 2 alef (long a), 3 alef with hamza above (vowelcarrier), 4 fathatan
key <AC02> { [Arabic_seen, Arabic_sheen, Arabic_sad] };
// 1 seen (s), 2 sheen (sh), 3 sad (sw)
key <AC03> { [Arabic_dal, Arabic_thal, Arabic_dad, Arabic_ddal] };
// 1 dal (d), 2 thal (dh), 3 dad (dw), 4 ddal (alveolar dr)
key <AC04> { [Arabic_feh ] };
// 1 feh (f)
key <AC05> { [U06A0, Arabic_ghain, Arabic_gaf] };
// 1 ain with three dots above (g), 2 ghain (gh), 3 gaf (g)
key <AC06> { [Arabic_ha, Arabic_hah, Arabic_tehmarbuta, Arabic_hamza] };
// 1 ha (h), 2 hah (h), 3 tehmarbuta, 4 hamza as letter
key <AC07> { [Arabic_jeem] };
// 1 jeem (j)
key <AC08> { [Arabic_kaf, U06AA] };
// 1 kaf (k), 2 swash kaf (k)
key <AC09> { [Arabic_lam] };
// lam (l)
key <AC10> { [Arabic_semicolon, NoSymbol, semicolon, colon] };
// 1 semicolon, 3 UK semicolon, 4 UK colon
key <AC11> { [Arabic_ain, Arabic_shadda, quoteright, at] };
// 1 ain, 2 shadda, 3 UK single quote, 4 UK @
key <BKSL> { [NoSymbol, NoSymbol, numbersign, asciitilde] };
// 3 UK hash, 4 UK tilde

// --------------
// QWER row
// --------------
key <AD01> { [Arabic_qaf] };
// 1 qaf (q)
key <AD02> { [Arabic_waw, NoSymbol, Arabic_hamzaonwaw, U06CF] };
// 1 waw (w), 3 waw with hamza above (vowel-carrier), 4 waw with dot above
key <AD03> { [U0656, Arabic_yeh, Arabic_hamzaonyeh] };
// 1 subscript alef (short e), 2 yeh (long e), 3 yeh with hamza above (vowel carrier)
key <AD04> { [Arabic_ra] };
// 1 ra (r)
key <AD05> { [Arabic_teh, Arabic_theh, Arabic_tah, Arabic_tteh] };
// 1 teh (t), 2 theh (th), 3 tah (tw), tteh (alveolar tr)
key <AD06> { [Arabic_yeh, Arabic_alefmaksura, Arabic_hamzaonyeh] };
// 1 yeh (y), 2 alef maksura, 3 yeh with hamza above (vowel carrier)
key <AD07> { [Arabic_damma, Arabic_waw, Arabic_hamzaonwaw, Arabic_dammatan] };
// 1 damma (short u), 2 waw (long u), 3 waw with hamza above (vowel-carrier), 4 dammatan
key <AD08> { [Arabic_kasra, Arabic_yeh, Arabic_hamzaonyeh, Arabic_kasratan] };
// 1 kasra (short i), 2 yeh (long i), 3 yeh with hamza above (vowel carrier), 4 kasratan
key <AD09> { [U0657, Arabic_waw, Arabic_hamzaonwaw] };
// 1 inverted damma (short o), 2 waw (long o), 3 waw with hamza above (vowel-carrier)
key <AD10> { [Arabic_peh] };
// 1 peh (p)
key <AD11> { [NoSymbol, NoSymbol, bracketleft, braceleft] };
// 3 UK opening square bracket, 4 UK opening brace
key <AD12> { [NoSymbol, NoSymbol, bracketright, braceright] };
// 3 UK closing square bracket, 4 UK closing brace

// --------------
// numeral row
// --------------
key <AE01> { [Arabic_1, NoSymbol, KP_1, exclam] }; // 1 digit 1, 3 UK digit 1, 4 UK exclamation mark
key <AE02> { [Arabic_2, NoSymbol, KP_2, quotedbl] }; // 1 digit 2, 3 UK digit 2, 4 UK double quote
key <AE03> { [Arabic_3, NoSymbol, KP_3, sterling] }; // 1 digit 3, 3 UK digit 3, 4 UK pound sign
key <AE04> { [Arabic_4, NoSymbol, KP_4, dollar] }; // 1 digit 4, 3 UK digit 4, 4 UK dollar sign
key <AE05> { [Arabic_5, NoSymbol, KP_5, percent] }; // 1 digit 5, 3 UK digit 5, 4 UK percent sign
key <AE06> { [Arabic_6, NoSymbol, KP_6, asciicircum] }; // 1 digit 6, 3 UK digit 6, 4 UK circumflex
key <AE07> { [Arabic_7, NoSymbol, KP_7, ampersand] }; // 1 digit 7, 3 UK digit 7, 4 UK ampersand
key <AE08> { [Arabic_8, NoSymbol, KP_8, KP_Multiply] }; // 1 digit 8, 3 UK digit 8, 4 UK asterisk
key <AE09> { [Arabic_9, NoSymbol, KP_9, parenleft] }; // 1 digit 9, 3 UK digit 9, 4 UK opening parenthesis
key <AE10> { [Arabic_0, NoSymbol, KP_0, parenright] }; // 1 digit 0, 3 UK digit 0, 4 UK closing parenthesis
key <AE11> { [U060D, NoSymbol, KP_Subtract, underbar] }; // 1 date separator, 3 UK dash, 4 UK underscore
key <AE12> { [NoSymbol, NoSymbol, KP_Equals, KP_Add] }; // 3 UK equals sign, 4 UK addition sign
Appendix F

*Annotated poem, كِسْوَاحِلِ (Kiswahili), 10pt, endnotes*

Appendix D presented the poem in Abdulkadir and Frankl (2013) at a font-size of 12pt, with annotations appear as footnotes. For comparison, this appendix prints the same poem in a font-size of 10pt, with annotations appearing as endnotes – see Section 8.7.2 and Section 8.7.3.
١ \(\text{كُنْيَمَانِمٖوْكَتَانْيَامَحَتَلِنِ} \) kunyamā ni mechoka ١ب/a  
I am weary of staying silent. For how much longer am I to remain dumb?  
kuwaona na ṭamāni wanangu huniepūka ١د/c  
My own children avoid me, though I long to see them.  
siwangu ni wa wendāni wālūbāki kunishika ١ف/e  
And those who remain to embrace me are not my own, but are the offspring of others.  
mbūna hunipija ziña mimi ni mewatendāni ١غ/g  
What have I done to you? Why do you wage war on me?

٢ \(\text{وَنَنْڠُمِمِوَدَمُوَانَوَأُسْوَاحِلِنِ} \) wāna wa uswāḥilini wanangu mimi wa ḏamu ٢ب/a  
My own flesh and blood, the children of Swahililand,  
yā kuniyuwa ni nāni asili hawana hāmu ٢د/c  
are uninterested in knowing who I am,  
na wana wa majirani wamenatia qaumu ٢ف/e  
and have left me to other peoples, and to the children of neighbours.  
mbūna hunipija ziña kūsa langū kūsa gani ٢ح/g  
What kind of fault is my fault? [O my children] why do you continue waging war on me?
I am your mother and am not yet infertile, nor has my ability to reproduce diminished.

I have given birth to children in Mambasa, and in the other islands [of the Swahili],

to politicians and to religious leaders,

to craftsmen in every field, and to war heroes.

I am the mother of Bwana Muyaka, and of Mwengo Athmani also,

and of Zahidi too, and many of his contemporaries,

they emerged from my womb, and shone like stars.

Look at Inkishafi. Read it attentively

and then you will understand, my dear friend,
What I am telling you. These verses are of enduring worth and will never die.

Who were those who composed them? They were my children who have passed on.

And the Bard of Mambasa, and Chiraghdin too,
they followed in my footsteps, they did not submit to lower standards.

He remains alone in the field, yet he stays strong.

I am still able to give birth. I have not yet reached the limit, but you have all despised me. You have left me high and dry,

now others have come forward to regulate me,

compiling standardized dictionaries.
I weep and lament when I look at the learned journals, for many of those who contribute are not my children, they are strangers to me. It is much the same with the media. Who are the ones who send in their compositions? Many do not come from the coast, although they may have a Mambasa address.

Look at the textbooks which are studied at our schools. They are written neither by Rajabu, nor by Sudi nor by Shani. The author is Njoroge, he is the helmsman.

When I am invited to conferences, I turn back before I arrive. I feel exceedingly bitter that I do not see you all there.
I bite my fingers in frustration, but what can I do?

My children, you have missed your opportunity. You have abandoned your own mother.

And I shed tears when I look at the results of the school exams.

Amongst those who are researching for degrees at the universities,

Who is to be blamed? Whose fault is it?

You esteem me not at all, yet you have not replaced me by another.
When I hear those who are not mother-tongue speakers speaking, I feel sick at heart.

Inflection is no longer employed, while grammatical [Swahili] is what I desire!

Even [their speech] is wanting in flavour, like a plug of tobacco in one's mouth.

I do not understand what they are saying. Are they singing? Are they complaining?

Were Bwana Muyaka to return, were he to come back to the world,

it would be necessary, my child, for him to go to a court of law,

and he would need to call witnesses who know me well,

and all of you would go to prison for the offence which you have committed against me.

Truly you have neither zeal nor self-confidence.

It irritates you not at all that you do not esteem me.
I am just like a ball in the playground,
highjacked to tend it

I am given a kick by anyone who passes by in the street.

Even in the field of Swahili prosody, those who are not mine have invented free verse, imitating foreigners.

For myself, I cannot accept that. That is not Swahili poetry.

What is the point of it all? These are preparations for war.

I am told that I belong to nobody in particular. How extraordinary!

How can I be rootless below ground and yet have branches above?

Who gave me my name? And who are they who wrote me down?

If I do not hail from Swahililand, then whence do I come?
That many speak me, [Swahili], is not of itself proof of origins,
or that I have no owner. What of the English language?

It is spoken by very many, in all corners of the world,

yet the language remains firmly established in its homeland, its roots have not been severed.
Endnotes


2. Bwana Muyaka was the outstanding Swahili poet of 19th century Mombasa. After his death many of his verses were recalled by Mu’allim Sikjuja Abdallah al-Batawi (died 1890) and transcribed with annotations by W.E. Taylor (1856-1927). After Taylor's death his papers were acquired by the library of the School of Oriental and African Studies (SOAS), London.

3. Mwengo Athmani: this 18th century poet from Pate composed the Utendi wa Tambuka (The Epic of Heraklios).


6. Bwana Mataka's full name is Muhammad bin Shee Mataka al-Famau (1825-1868). He was ruler of Siyu, as was his father. His mother was Mwana Kupona, famous for the poem of advice written to her daughter. Bwana Mataka died in Mombasa's fort while imprisoned by the Busa’idi.

7. The Inkishafi, according to W.E. Taylor (Stigand 1915, pp96-105), is “a great, if not the greatest, religious classic of [the Swahili-speaking peoples]”. The poem, concerned with the decay of Pate (formerly a flourishing town in northern Swahililand), may remind some readers of Thomas Gray's Elegy written in an English churchyard (London 1751).

8. The Bard of Mambasa refers to Ustadh Ahmad Nassir Juma Bhalo, see S. Chiraghdin (1971).


10. In an unpublished commendation from 12 June 1974 J.W.T. Allen writes about Ahmad Sheikh Nabhany: “I am privileged to have a wide circle of friends and acquaintances among Swahili scholars of Swahili. I have some knowledge of their rating of themselves and I can name perhaps half a dozen (still living) who are always referred to as the most learned. To me they are walking dictionaries and mines of information and Ahmed is unquestionably one of them. He comes of a family of scholars whose discipline is as tough as any degree course in the world. They have no time for false scholarship or dilettantism. That this profound learning is almost wholly disregarded by those who have been highly educated in the western tradition affects almost everything written today in or about Swahili. When I want to know some word or something about Swahili, I do not go to professors, but to one of the bingwa known to me. One of these could give a much greater detail of assessment, but of course his opinion would not carry the weight of one who can put some totally irrelevant letters after his name”. For a biography see Said (2012).

11. For almost a century the principal publisher of standardized Swahili dictionaries has been the Oxford University Press (OUP). Clearly OUP has to be profitable, and profitable is what, over the years, their dictionaries of standardized Swahili have been. However, if one considers excellence in research and scholarship not one of the OUP’s standardized Swahili lexicons can begin to compare with the Oxford English Dictionary (‘more than 600,000 words over a thousand years’). Fortunately for Swahili and for Swahili studies there exists the monumental Dictionnaire swahili-français, compiled by Charles Sacleux (Sacleux 1939). Sacleux's chef d’oeuvre (‘unprecedented in historical depth, dialectological detail and philological knowledge’) can now be accessed electronically, courtesy of Swahili Forum (uni-leipzig.de/~afrika/swafo/index.php/sacleux). Heartfelt thanks are due to Thilo Schadeberg and Ridder Samsom.

12. njoroge: a name representing those who have their origins in the East African interior (the bara).

13. charo: a name representing those who have their origins in the coastal hinterland (the nyika).

14. These words echo the words of the Inkishafi: “wakauma zanda na kuiyuta”. Readers unfamiliar with this Swahili gesture of regret could consult Eastman and Omar (1985).

15. Kibwezi and Kisumu are places in the East African interior.

16. The lake is Lake Nyanza, also known as Lake Victoria.

17. Over the years young people on Lamu Island (and indeed elsewhere in northern Swahililand) have received a raw deal in their primary and secondary education. They have ‘lagged far behind’ their counterparts from the interior, and so Mother Swahili grieves for her marginalised children.
References


APPENDIX F. ANNOTATED POEM, كِسْوَاحِلِ (KISWAHILI), 10PT, ENDTNOTES

References