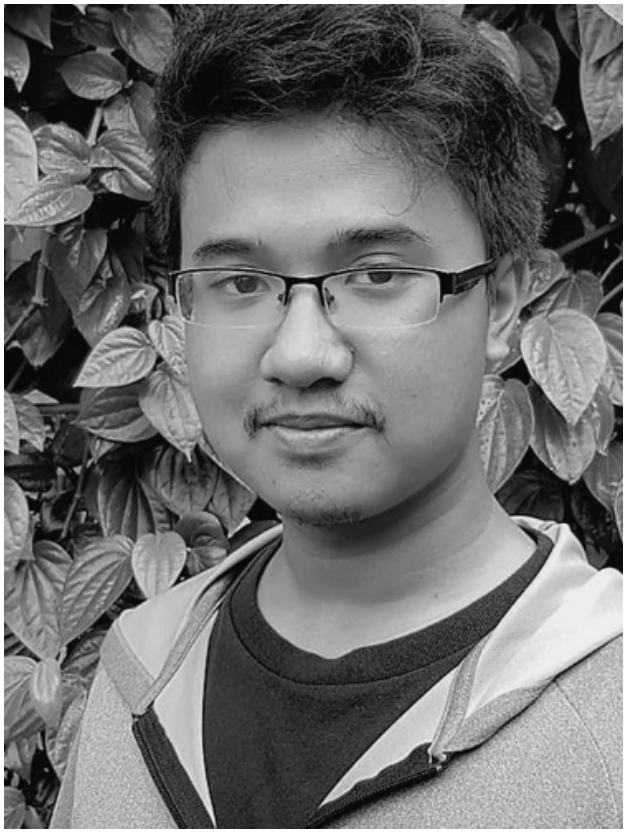
Learnings from encoding Kawi

Norbert Lindenberg

𑼥𑼾𑼴𑼂𑼨𑼾𑼡𑽂𑼭𑼶𑼥𑽂𑼣𑼾𑼥𑽂𑼨𑼾𑼂𑼔𑽁𑽄

Kawi proposal authors



Aditya Bayu Perdana

𑼄𑼣𑼶𑼡𑽂𑼫𑼨𑼫𑼸𑼦𑽀𑼂𑼣𑼥𑽄

Ilham Nurwansah

𑼆𑼭𑽂𑼲𑼪𑽂𑼥𑼸𑼂𑼮𑼥𑽂𑼱𑼃𑽄

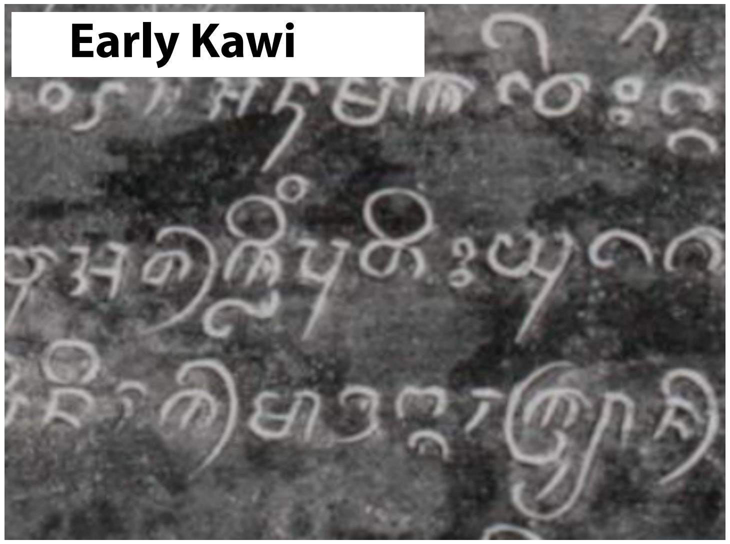
Agenda

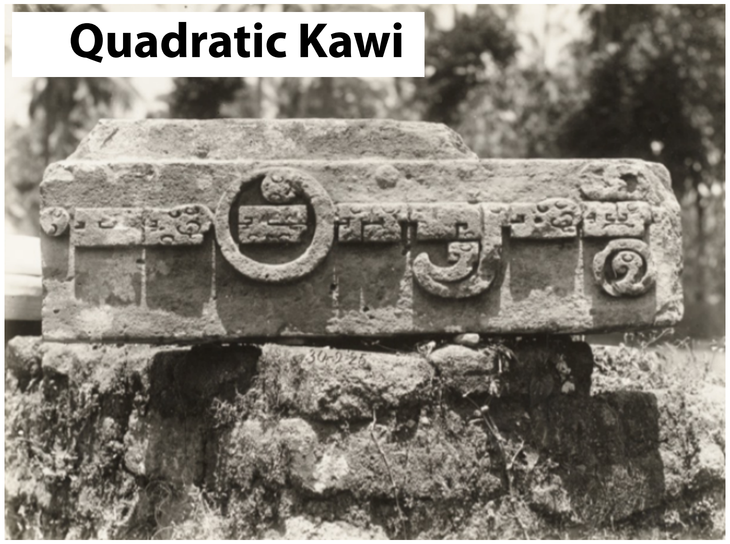
- Kawi
- Encoding characters
- Encoding clusters
- Test implementation with font and keyboard
- Summary

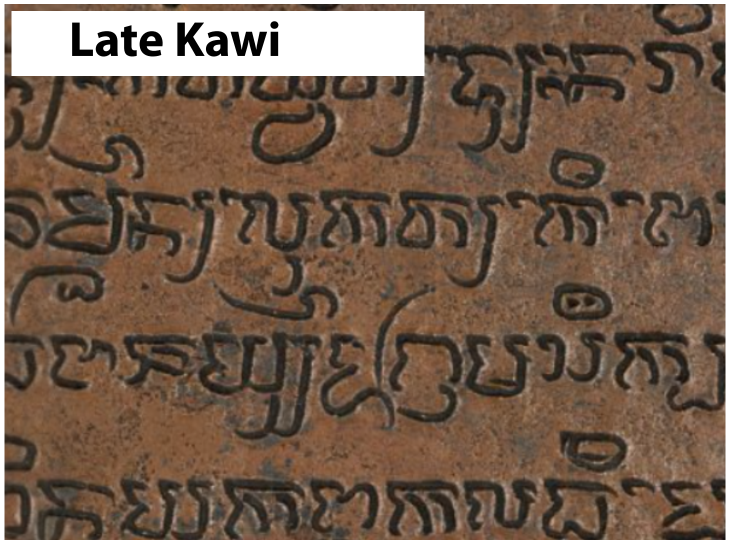
Kawi

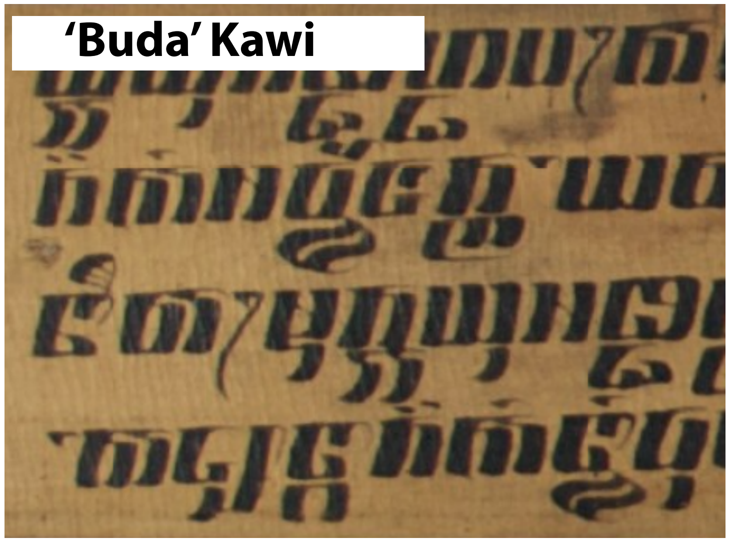
- Historic script of Java, Sumatra, Malay peninsula, Bali, Philippines
- Used to write Old Javanese, Sanskrit, Old Malay, Old Balinese, Old Sundanese
- Used 8th to 16th century
- Derived from Brahmi via Pallava

Kawi

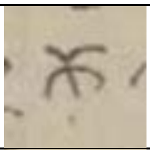
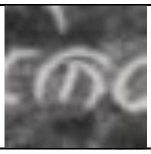
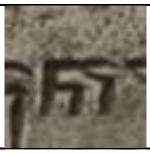
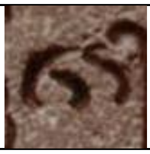
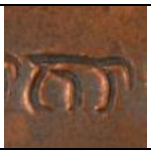
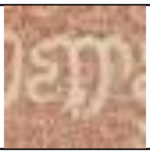








Kawi: One script?

		TVIT32	OD 13695	OD 3871	OD 741a	MSS Jav 106	KERN E29	various
KA	ᬓ							

- Character shapes vary widely over history
- But:
  - No significant structural changes
  - Shapes can be handled by fonts

→Encode as single script

Kawi as Brahmic script

- Consonants with inherent vowel
- Dependent vowel signs override inherent vowel; attach on any side of consonant
- Virama sign suppresses inherent vowel
- Conjunct forms of consonants suppress inherent vowel of base consonant
- Repha sign for cluster-initial *r-*

Kawi as Brahmic script

- Kawi is 64th Brahmic script in Unicode
- Could there be an automated process for Unicoding Brahmic scripts?
- Review of key decisions in encoding Kawi, and see what can be learned from them

Encoding characters

User-level characters

- Significant research by Bayu, Ilham, and collaborators
- Over 50 inscriptions and manuscripts evaluated
- Documentation of characters and their shapes

User-level characters

- Sumberwatu gold plate (Yogyakarta)
- Contains all 33 Kawi consonant letters
    - Some previously known only in subscript form

Virama

- Before computers: Visible mark to suppress inherent vowel (Kawi: ◌𑽁)
- Unicode: Also used to form conjuncts ([illegible], [illegible], [illegible])
- Three kinds of viramas
  - Visible mark – Pure_Killer
  - Invisible conjunct former – Invisible_Stacker
  - Shape-shifting depending on context – Virama

Virama

- Shape-shifter: Font is in control
  - Useful in scripts with optional conjunct ligatures (Devanagari)
  - Users can use ZWNJ and (sometimes) ZWJ to influence shape
    - But: ZWNJ, ZWJ are hard to work with

Virama

- Visible mark + invisible conjunct former
  - User is in control (important for scholars!)
  - ZWJ and ZWNJ not needed
  - Sufficient for script with fixed set of conjunct forms

→Chosen for Kawi: ◌𑽁, ◌𑽂

Repha

- Mark representing cluster-initial *r-*, often above-base – Kawi: ◌𑼂
- Unicode has 14 ways to represent repha
  - Many use ZWJ to distinguish repha from nominal form of initial *r-* or from eyelash *ra*
  - Most encode repha as first part of cluster, some don’t

Repha in Kawi

- Cluster-initial *r-* usually shows as repha ◌𑼂, but occasionally as nominal *ra* glyph 𑼬
  → Encode repha separately to avoid need for ZWJ
- Repha sign usually means repha; rarely final *-r*
  - Opposite of Balinese, Javanese, where cognates of Kawi repha usually mean final *-r*, rarely repha

  → Encode repha before base consonant

Multi-part characters

- Kawi has several independent and dependent vowels that visually consist of multiple parts
  - 𑼄𑽀𑼴 letter *euu* ↔ 𑼄 letter *a*, ◌𑽀 sign *eu*, ◌𑼴 sign *aa*
  - ◌𑼾𑼴 sign *o* ↔ ◌𑼾 sign *e*, ◌𑼴 sign *aa*

Multi-part characters

- Unicode has 3 ways to handle multi-part characters
    - Encode multi-part characters atomically, with canonical decomposition – e.g. Balinese
    - Encode multi-part characters atomically; prohibit representation as sequence – e.g. Devanagari ("do not use"); Khmer (max. 1 vowel)
    - Do not encode multi-part characters; use sequence of components instead – e.g. Javanese

Multi-part characters in Kawi

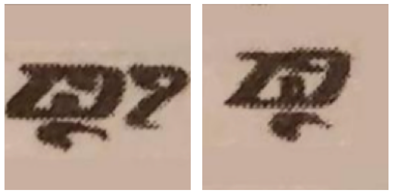
- Components of multi-part characters in Kawi always also are vowels by themselves
- Duplicate encoding with canonical decomposition has no advantage
- “Do not use” lists or maximum number of vowels complicate implementation

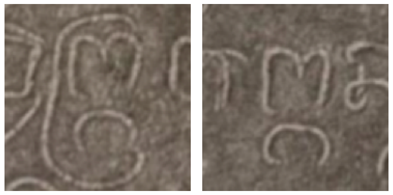
→ Encode as sequences of components

Multi-part characters in Kawi

- Some multi-part characters have visually distinct variants that aren’t multi-part

  - 𑼇 letter *ii* ↔ 𑼆 letter *i*, ◌𑼴 sign *aa*
  - 𑼇 letter *ii* ↮ 𑼆 letter *i*, ◌𑼴 sign *aa*





→Encode visually distinct variants separately

Encoding clusters

Clusters → logical order

- Clusters in Brahmic scripts are two-dimensional; code point sequences are linear
- Multiple encodings for strings that user can’t distinguish lead to problems in search and to spoofing
- Need to define *correct* code point sequences

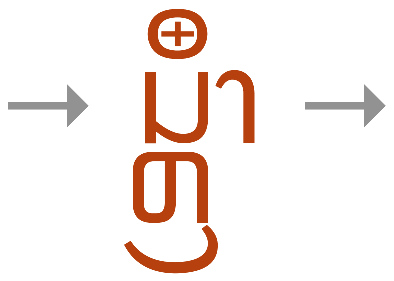
Logical order

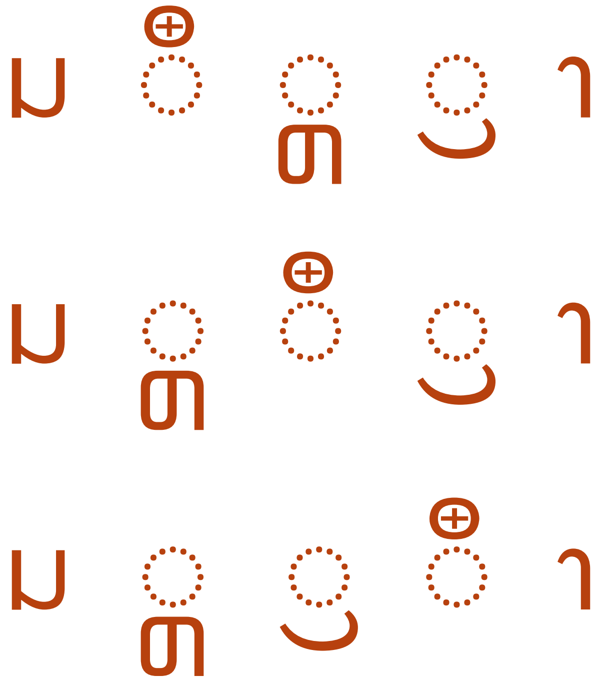
- Order in which text is stored in memory
- Need not match typing order – keyboards can reorder to match user expectations
- May be based on visual order or phonetic order

Visual order

- Thai, Lao + 2; most non-Brahmic scripts
- Encode spacing characters in writing direction
- Encode interacting nonspacing marks from base outwards
- Equivalence between sequences of non-interacting nonspacing marks
- "Interacting" ↔ same combining class

Visual order

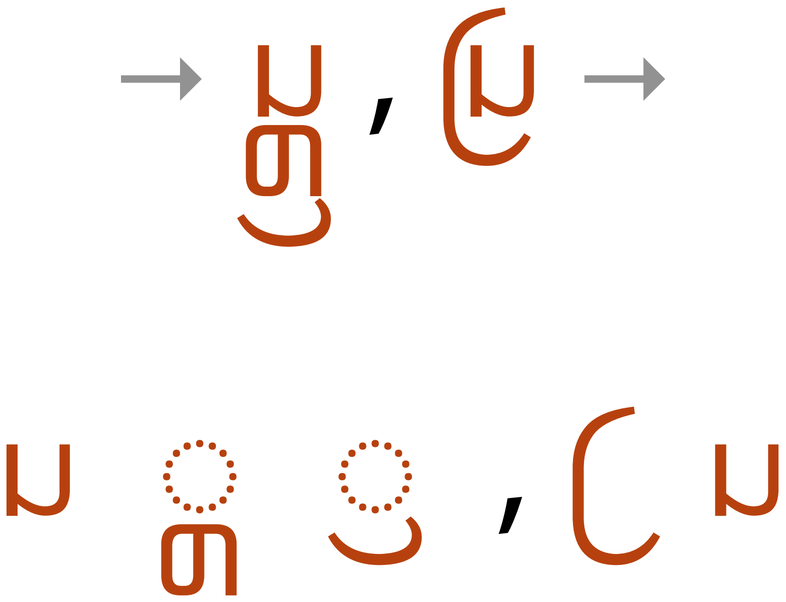




Visual order

- Problems for Brahmic scripts
  - Combining class can’t be defined for marks encoded as virama-consonant sequences
  - Combining class for marks can’t be corrected when minority languages use them differently or mistakes were made

Visual order



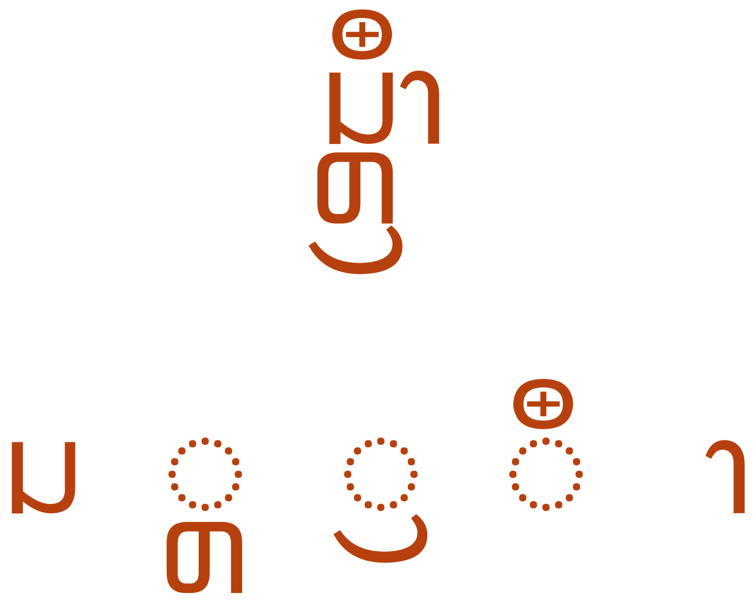
Visual order

- Problems for Brahmic scripts
  - Spacing/nonspacing contextual forms have to be encoded separately
  - Characters don’t appear in order needed for sorting

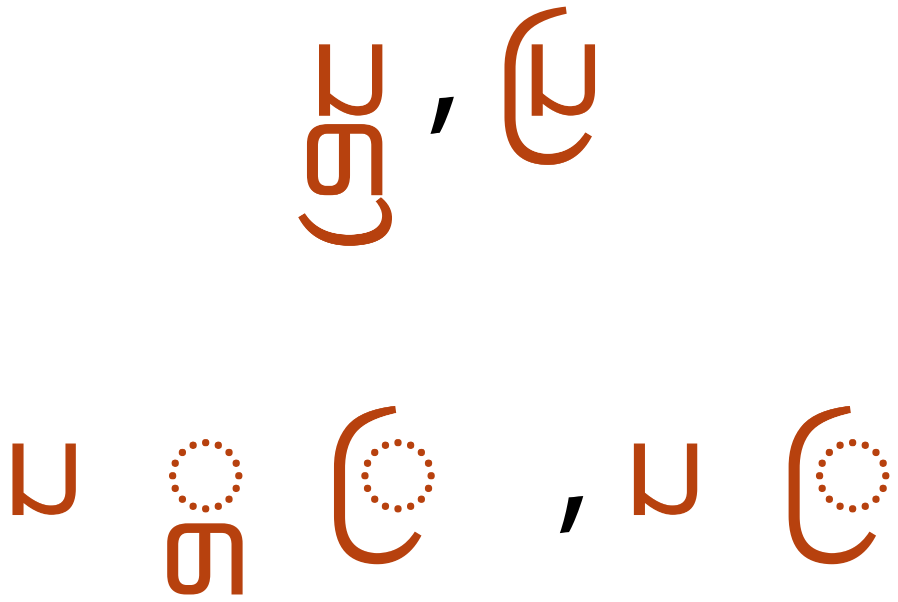
Phonetic order

- Most Brahmic scripts
- Encoding order is primarily phonetic order; secondarily position or other criteria

Phonetic order



Phonetic order



Phonetic order

- Problems:
  - Incompatible with equivalences defined through combining classes → set CCC=0 (except virama)
  - Some characters aren’t phonetic → need to resolve where they fit in
  - Unicode doesn’t do that → compatibility issues

Universal Shaping Engine

- OpenType shaping engine for the rest of us
- Defines generic cluster model for Brahmic scripts based on Unicode data:
  - General category
  - Indic syllabic category
  - Indic positional category

Kawi cluster model

- Multiple positions for repha considered because of use as final *-r*

→Encode at start of cluster, even for final *-r*

- After defining Unicode data for Kawi, derived USE cluster model worked fine

→Kawi adopts cluster model provided by USE

Consonant_Preceding_Repha?

(Consonant | Vowel_Independent | Number | Consonant_Placeholder)

(Invisible_Stacker (Consonant | Vowel_Independent | Number))*

Vowel_Dependent-Left* Vowel_Dependent-Top* Vowel_Dependent-Bottom* (Vowel_Dependent-Right | Pure_Killer-Right)*

Bindu-Top*

Visarga-Right*

[◌៌]?

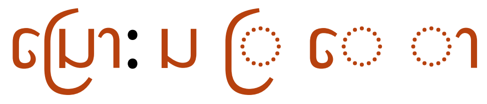
[ក ខ គ ឃ ង ច ឆ ជ ឈ ញ ដ ឋ ឌ ឍ ណ ត ថ ទ ធ ន ប ផ ព ភ ម
យ រ ល វ ឝ ឞ ស ហ ឡ អ ឣ ឤ ឥ ឦ ឧ ឨ ឩ ឪ ឫ ឬ ឭ ឮ ឯ ឰ ឱ ឲ
ឳ ◌]

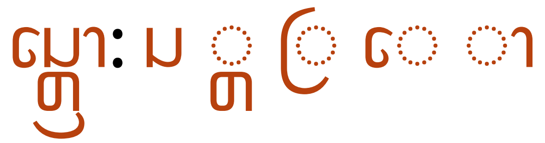
[◌្ក ◌្ខ ◌្គ ◌្ឃ ◌្ង ◌្ច ◌្ឆ ◌្ជ ◌្ឈ ◌្ញ ◌្ដ ◌្ឋ ◌្ឌ ◌្ឍ ◌្ណ ◌្ត ◌្ថ ◌្ទ ◌្ធ ◌្ន ◌្ប ◌្ផ ◌្ព
◌្ភ ◌្ម ◌្យ ◌្រ ◌្ល ◌្វ ◌្ស ◌្ហ ◌្ឡ ◌្អ ◌្ឣ ◌្ឥ ◌្ឧ]*
[◌េ ◌ែ]* [◌ិ ◌ី ◌ឹ]* [◌ុ ◌ូ ◌ួ]* [◌ា ◌ៅ ◌ៀ]*

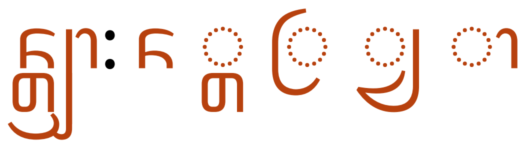
[◌៍ ◌៉]*

[◌ះ]*

Kawi cluster examples

-  (ni*sr*oma – hairless)

-  (dharmaśā*str*opadeśa – teaching of the treatises on dharma*)*

-  (ma*ntry*āgöng – great minister)

Test implementation

Implementation required

- Brahmic scripts are complicated
- Serious problems have occurred in several scripts

→Encoding should be tested before frozen

- See presentation “Integrating the development of encoding, font, and keyboard” at IUC 2018

Kawi implementation

- Font designed by Aditya Bayu Perdana, engineered by Norbert Lindenberg
- Based on Apple Advanced Typography
- Tested in Pages, Safari, Firefox, Chrome and right here in Keynote
- App with keyboard for iOS/iPadOS

Laguna copperplate

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

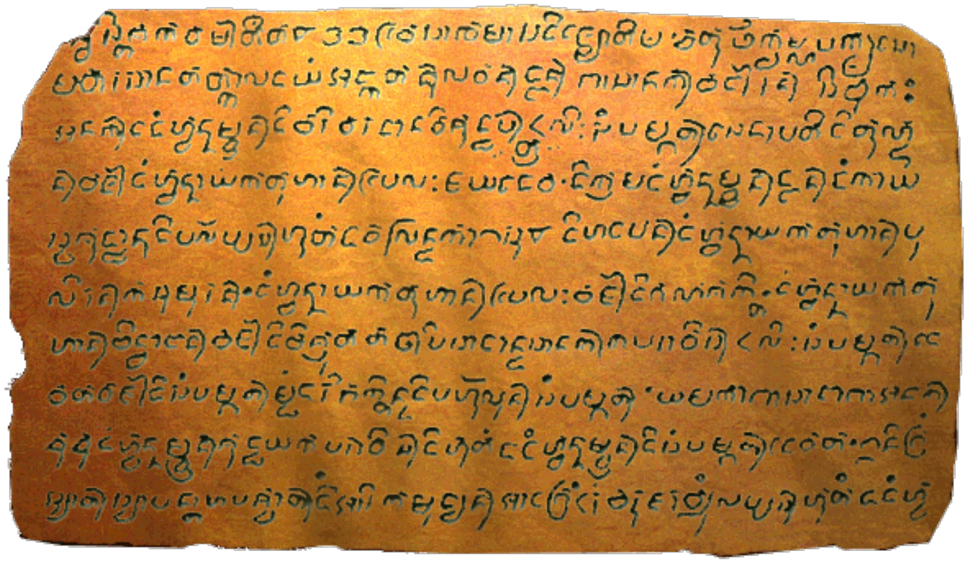
[illegible]

[illegible]

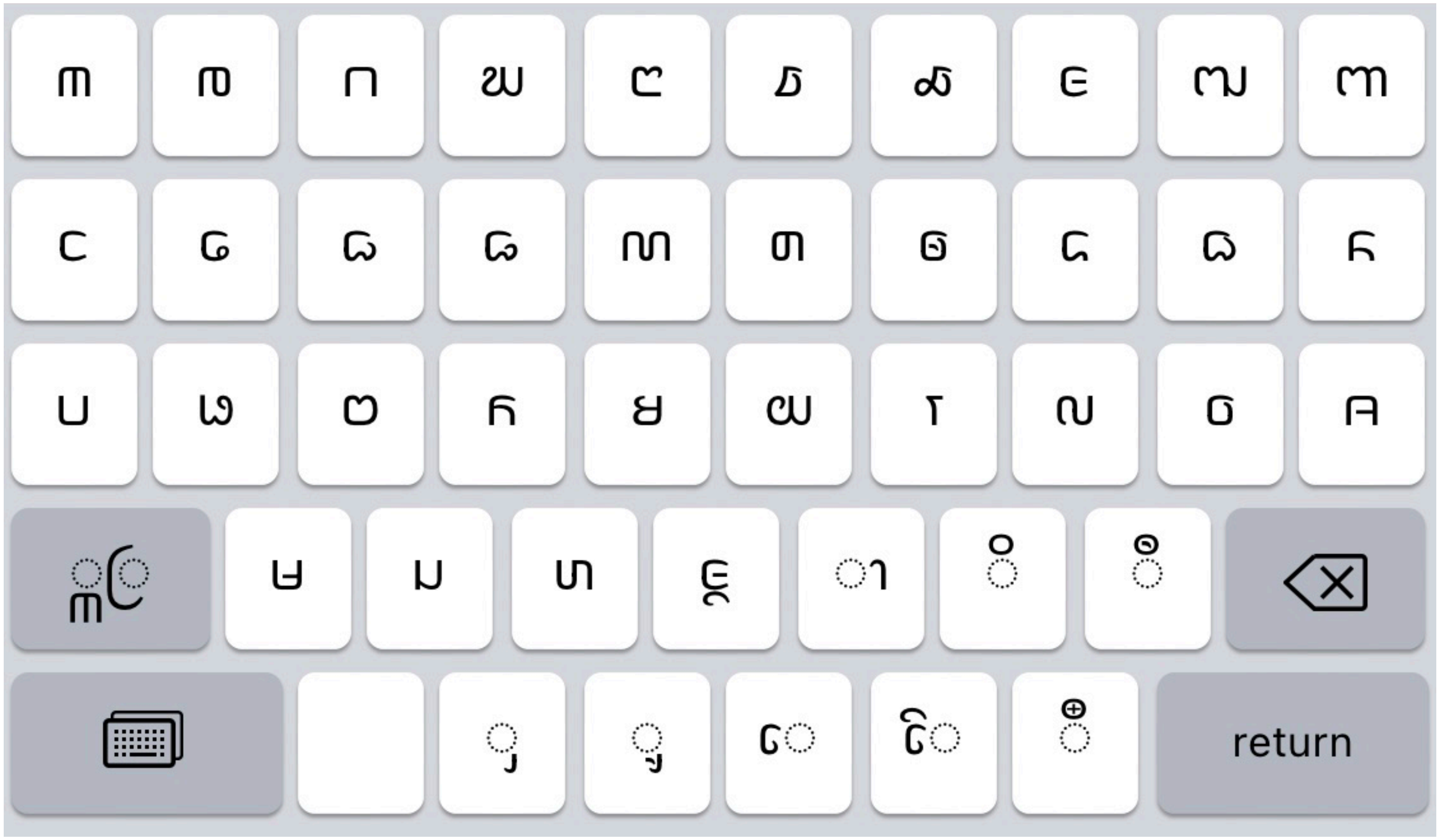
[illegible]

[illegible]

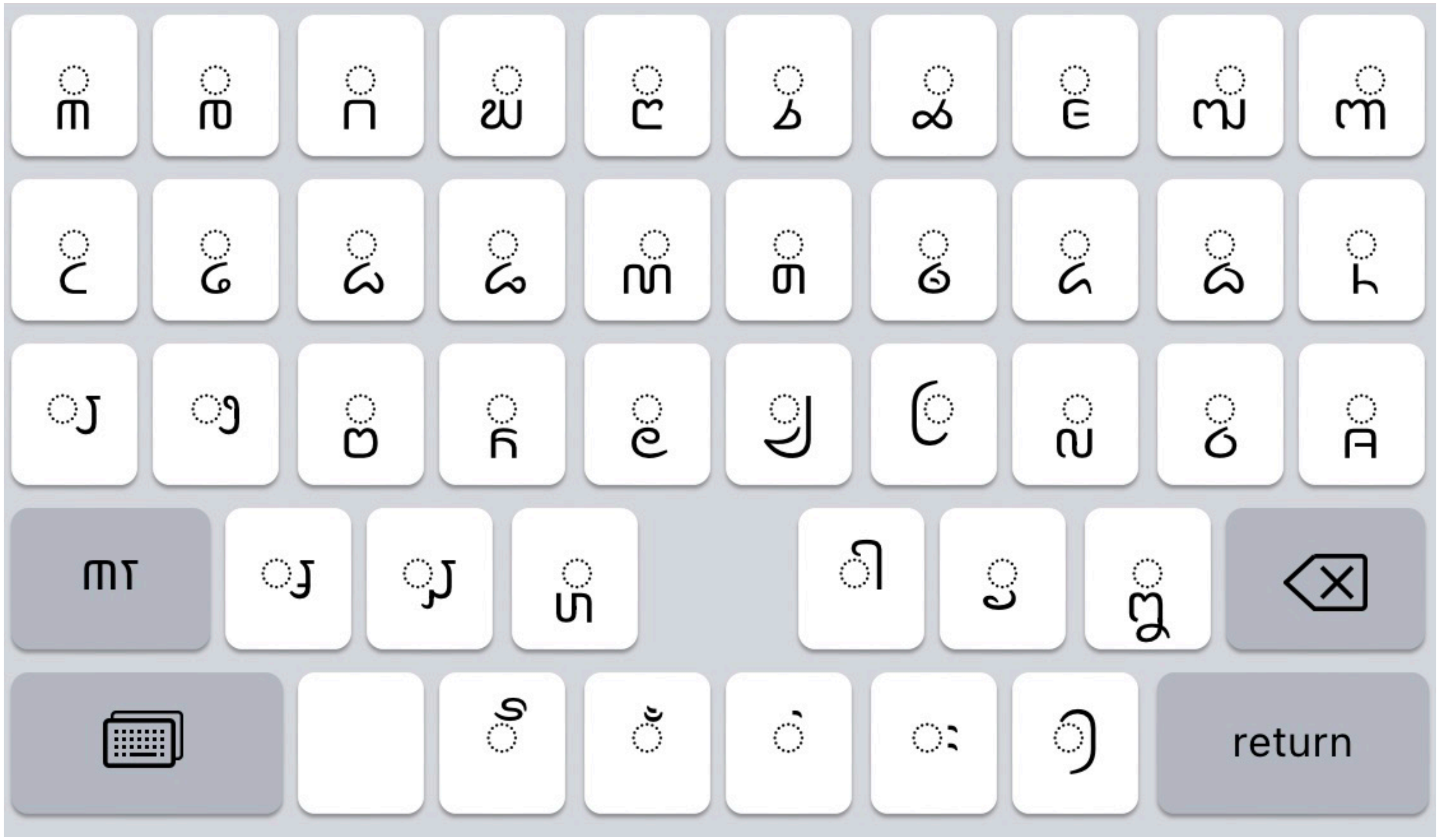
[illegible]



Keyboard



Keyboard



Summary

No automated process

- Scripts have different features
    - Repha vs. final -r
- Script users have different requirements
    - Scholars vs. online communities
- Looking at reasons for Kawi choices can help encode future scripts

Advice

- Take advantage of changes in technical environment
    - Standard rendering with default cluster model: Universal Shaping Engine
    - Flexible input technology, e.g. Keyman

Advice

- Avoid mistakes made in encoding 1..63
  - ISCII/Devanagari influence
  - Magic characters
  - Custom encoding of repha
- Define and validate cluster structure
- Create test implementation

References

- Aditya Bayu Perdana, Ilham Nurwansah: Proposal to encode Kawi. L2/20-284R
  unicode.org/L2/L2020/20284r-kawi.pdf
- Norbert Lindenberg: Repha representation for Kawi. L2/20-283
  unicode.org/L2/L2020/20283-kawi-repha.pdf
- Lindenberg Software: The Aksara Kawi app
  lindenbergsoftware.com/en/keyboards/kawi/support.html

Fonts used

- Tantular Kawi

  Design by Aditya Bayu Perdana. Engineering by Norbert Lindenberg.

- Myriad Pro

  Design by Robert Slimbach and Carol Twombly at Adobe Systems Inc.