Amharic Normalization and Canonical Representation (v0)

Simachew Mengiste

December 12, 2025

1 Overview

This document summarizes the current state of the Amharic Normalization project (v0). The goal of the project is to enable reliable, deterministic processing of Amharic text across digital systems, including web applications, translation pipelines, and AI-assisted writing tools.

The system establishes a clear architectural separation between:

- a **canonical representation layer** (CAR v0),
- a **normalization layer** for real-world input (AN-v0),
- a **punctuation harmonization layer**,
- a **UI lexicon resolution layer**,
- and a **stable API contract** for downstream systems.

Each layer has a well-defined responsibility and enforces explicit guarantees.

2 Motivation

Existing Amharic input and translation systems suffer from:

- inconsistent transliteration conventions,
- ambiguous mappings from Latin input to Ethiopic script,
- silent guessing that produces incorrect or non-existent words,
- inconsistent punctuation handling,
- and unstable UI terminology.

This project addresses these issues by enforcing explicit rules, surfacing ambiguity, and guaranteeing correctness once input reaches the canonical layer.

3 Stage 1: Canonical Representation – CAR v0

CAR v0 defines a strict, bijective encoding of Ethiopic characters.

3.1 Design Principles

- Every Ethiopic character maps to exactly one CAR token.
- Every CAR token maps back to exactly one Ethiopic character.
- The mapping is table-driven and context-free.
- No ambiguity, no guessing, no heuristics.

3.2 CAR Token Structure

Each token has the structure:

`<base><variant><order>`

- `base`: consonant family (e.g. `l`, `sh`, `t'`)
- `variant`: optional digit for family disambiguation
- `order`: Ethiopic order (1–8)

3.3 Examples

CAR	Ethiopic
`y1t6`	**የት**
`t4r6f4l1h6`	**ታርፋለህ**
`m6d6r6s6sh6`	**መድረስሽ**

3.4 Formal Guarantees

CAR v0 guarantees:

- Bijectivity: encode(decode(x)) = x
- Determinism: identical input produces identical output
- Table completeness for supported characters
- Version stability within a release

Any change to canonical tables requires a CAR version increment.
CAR forms the immutable ground truth of the system.

4 Stage 2: Amharic Normalizer – AN-v0

AN-v0 converts real-world input into canonical CAR.

4.1 Supported Inputs

- Ethiopic Unicode text
- Latin transliteration (informal or standardized)
- Mixed Ethiopic and Latin text

4.2 Pipeline Structure

AN-v0 operates in ordered stages:

1. Script segmentation
2. Deterministic Ethiopic normalization
3. Latin decoding (ambiguity-aware)
4. Span merging
5. Canonical encoding (CAR v0)
6. Ambiguity analysis and confidence scoring

Each stage operates independently and passes structured output to the next.

4.3 Auto vs. Strict Latin Modes

Auto Mode

- Designed for real-world input.
- Applies bounded heuristics informed by linguistic practice.
- Produces ranked alternatives when ambiguity exists.
- Never invents tokens outside CAR tables.

Strict Mode (Latin-Std)

- Fully deterministic and reversible.
- Requires explicit base and order specification.
- Intended for developers and evaluation workflows.

5 Stage 2B: Punctuation Harmonization

Punctuation normalization is implemented as a table-driven layer that operates independently of letter-level transliteration logic.

5.1 Design Principles

- Only table-defined punctuation is transformed.
- Characters outside the table pass through unchanged.
- Letter-level transliteration is never altered by punctuation rules.
- Directional consistency is preserved.

5.2 Standard Mapping Table

Ethiopic	Name	ASCII Equivalent
፡	Word Separator	(space)
።	Full Stop	.
፣	Comma	,
፤	Semicolon	;
፥	Colon	:
፦	Preface Colon	::

5.3 Smart Period Rule

The ASCII period serves multiple roles (sentence end, decimal, abbreviation).

Conversion to ። occurs only when the period functions as a sentence terminator. Periods used in:

- decimal numbers,
- common abbreviations,
- initialisms,

are preserved as ASCII.

5.4 Digit Handling Policy

ASCII digits (0–9) are currently passed through unchanged in both Latin-to-Ethiopic and Ethiopic-to-Latin directions.

Ethiopic numerals are not automatically substituted.

Future versions may introduce optional digit conversion modes.

6 Key Standardization Decisions

6.1 Order and Vowel Mapping

- `e` → order 1
- `ei` → order 5
- No vowel → order 6

6.2 Latin-Std Carrier Letters

Uppercase carriers represent standalone vowel bases:

`A`	አ
`U`	ኡ
`I`	ኢ
`AA`	ኣ
`EE`	ኤ
`E`	እ
`O`	ኦ

These never function as vowel modifiers of a preceding consonant.

6.3 Explicit Base Selectors

`K`	ኽ
`N`	ኝ
`Z`	ዠ
`X`	ጸ
`P`	ጰ
`T`	ጠ
`C`	ጨ
`c`	ቸ

These are honored in both Auto and Strict modes.

7 Stage 3: API Contract

7.1 Endpoint

```
POST /normalize
POST /resolve-ui
GET  /ui-lexicon
```

7.2 Response Structure (Normalize)

- `text_am`
- `car`
- `confidence`
- `alternatives`
- `meta`

The API never silently collapses ambiguity.

8 Stage 4: UI Lexicon Resolution (v1)

A pinned UI lexicon provides deterministic resolution of common interface strings (e.g. login, logout, submit).

8.1 Resolution Strategy

1. Alias canonicalization
2. Direct Amharic match
3. Unique alternative match
4. CAR-level match

8.2 Properties

- Lexicon entries are CAR-pinned.
- Resolution never invents new keys.
- Ambiguous matches fail explicitly.

9 Versioning Policy

- CAR version increments when canonical tables change.
- AN version increments when normalization heuristics change.
- API version increments when request/response contracts change.

10 Implementation Status

- CAR encoder/decoder: bijective
- AN-v0 normalization logic: implemented
- Latin-Std rendering: implemented
- Punctuation harmonization: implemented
- UI lexicon resolution: implemented
- FastAPI backend: operational
- Frontend interface: operational
- Validation tests: passing

11 Conclusion

The project establishes a principled and deterministic foundation for Amharic text processing. Ambiguity is surfaced rather than hidden, correctness is enforced at the canonical layer, and real-world input is handled transparently.

The system provides a reliable infrastructure layer for Amharic language technology, including localization systems, translation pipelines, and AI-assisted writing tools.