Morphology-Aware Reversible Semantic Tokenization and Hierarchical Word Composition for Tamil Language Models

Anand Murugan
anand.murugan@gmail.com
2026-07-31

Abstract

Statistical subword tokenizers efficiently split any input text into tokens that a language model can process, but their units need not align with lexical or grammatical structure. This is particularly consequential for Tamil, where a written word can combine stem changes, case, number, tense, agreement, voice, clitics and multiple linked verbs. We present a **Tamil morphology system** built by extending the open-source Thamizhi-Morph analyzer and generator, a bounded byte-exact semantic tokenizer and a learned hierarchical word composer. **12 Finite-State Transducers (FSTs)** analyze Tamil words into lemmas and grammatical features. The tokenizer can also reconstruct the original text exactly, using Tamil character and byte fallbacks when necessary.

The flat tokenizer (morphology-flat) represents each Tamil word with lemmas and grammatical features as separate input tokens, exposing useful semantic structure but producing significantly more tokens per word. Our word-composer keeps the lemmas, but summarizes each word's grammatical features into a single summary feature, and later uses sentence context to recover the most relevant grammatical details for the decoder.

We compare **morphology-flat**, the signal-preserving **word-composer** and tokenizer arms based on **Sarvam-1, AI4Bharat IndicBERTv2 and BrahmicTokenizer-131K**. All systems use the same **69,591 Tamil–English training pairs**, **18.97-million-parameter encoder–decoder**, **40,000 updates**, English target tokenizer, optimizer, rotary positions, numeric-copy policy and generation settings.

On the one-time **3,539-row** protected IN22/FLORES+ evaluation, **morphology-flat achieves the best pooled scores: 10.63 BLEU, 35.26 chrF++ and 0.6276 COMETKiwi**. Compared with AI4Bharat—the strongest external-tokenizer arm—these represent **relative score improvements of 7.2%, 3.2% and 2.6%**, respectively. The word-composer scores **10.30, 34.88 and 0.6241, improving on AI4Bharat by 3.8%, 2.1% and 2.0%**, while also outperforming the Sarvam and Brahmic arms.

Compared to morphology-flat, the composer reduces mean global source states from **71.48 to 29.08** (by **59.3%**). An operation-count estimate gives the composer **9–21% fewer inference FLOPs**, depending on decoder caching. Its small quality deficit is concentrated in longer formal FLORES+ sentences; the two IN22 partitions are paired ties.

> Later development-only tests show that post-encoder retrieval of the original grammar factors is important, while earlier retrieval and tested two-summary designs do not improve the quality-efficiency tradeoff. These results show both the benefit and the cost of explicit morphology. Lemmas and grammatical features improve translation under the same small-model budget. The word-composer reduces global attention states significantly and is estimated to require fewer operations (FLOPs) overall, although it retains a small translation-quality gap.

1. Introduction

Subword tokenization made open-vocabulary neural machine translation practical by representing rare words as reusable pieces rather than fixed word IDs (Sennrich et al., 2016). Language-independent implementations such as SentencePiece can learn directly from raw text and avoid a language-specific preprocessing pipeline (Kudo and Richardson, 2018). These are strong engineering properties. They do not, however, imply that a learned piece corresponds to a lemma, case marker, tense, auxiliary relation or other linguistic category.

Tamil exposes this distinction clearly. The surface word மரங்களை can be analyzed as மரம் + noun + plural + accusative; படித்தான் as படி + past + third-person singular masculine; and படித்துக்கொடுத்தான் as two lexical verbs connected by a typed nonfinite relation. Stem changes prevent this structure from being recovered by suffix removal alone: மரம் → மரத்தை, ஆறு → ஆற்றை, காடு → காட்டை and பூ → பூவை follow different lexical classes.

This work asks two questions. First, can Tamil morphology be exposed to a neural model in a deterministic, semantically explicit, auditable and exactly reversible representation? Second, can a model exploit that typically verbose representation without paying the full cost of flat attention over every factor?

We separate the system into three layers:

1. an FST-based morphology system, supported by lemma lexicons, that can analyze Tamil words into lemmas and grammatical features or generate words from those analyses;
2. a semantic tokenizer that produces a semantic token stream. It contains lemmas and grammatical features, along with any sandhi, spelling-variant, ambiguity-marker, structural, grapheme or byte tokens needed to reconstruct the original input exactly;
3. a neural interface that presents the same tokenized information to a language model either as a flat sequence (flat-morphology) or as a compact representation organized by word (word-composer).

This separation keeps the tokenizer output easy to inspect while allowing the model to reorganize the same information for efficiency. Lemmas, grammatical features and reconstruction controls have different purposes, but they remain in one exactly reversible token stream. This design also gives us two clear comparisons. Comparing flat-morphology with conventional statistical tokenizers tests the effect of changing the source representation. Comparing the word-composer with flat-morphology tests the model architecture while keeping the tokenizer output exactly the same.

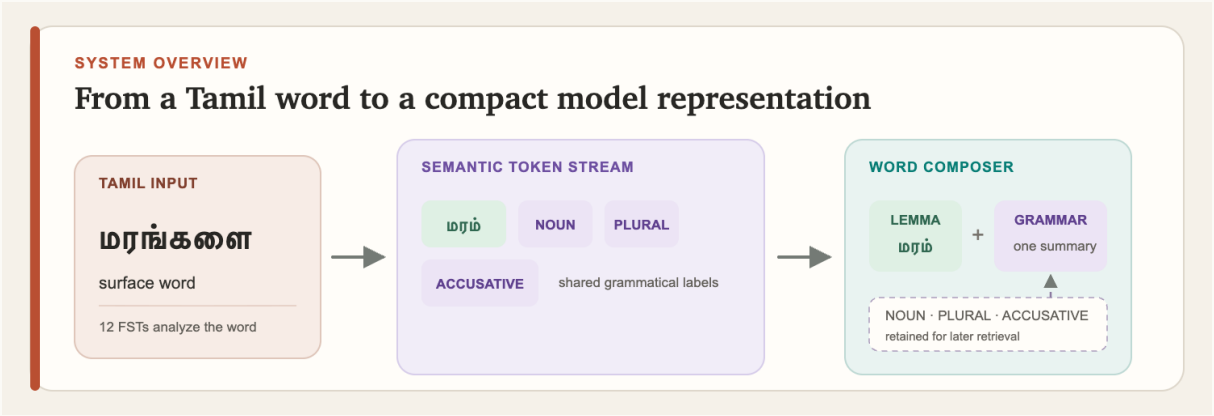

Figure 1. The tokenizer separates a Tamil word into its lemma and grammatical features. The word-composer keeps the lemma and combines the grammatical features into one summary for the model. It also keeps the original features so the model can consult them later.

Morphology-flat produces the best protected-data translation quality in one matched-parameter, one-seed small-model experiment. The signal-preserving composer is second-best at protected-data translation quality while reducing global token count by **59.3%**. Both morphology-flat and composer beat other external tokenizers. Later development experiments explain several design choices: retrieving grammar after sentence encoding helps; moving retrieval earlier hurts; and neither a hand-written nor a learned two-summary design makes enough use of its second state to justify the added positions.

Our contributions are:

- a Tamil morphology system built from 12 FST models, with clearly defined word classes, inflectional rules, compound forms, preservation of valid alternative analyses, and tests in both directions;
- a semantic tokenizer with a fixed vocabulary that can reconstruct the original text exactly;
- a fallback method that breaks unfamiliar words into Tamil character units before using bytes, so they do not collapse into an unknown token;
- a controlled comparison between the flat-morphology token sequence and a word-composer adapter that shortens the model input without discarding information
- tokenizer evaluations by training Tamil-to-English translation language models with comparison using the same data, model parameter count, training process, number handling and text-generation settings;
- a reproducible release that records exact versions and file checksums for every important artifact.

2. Tamil Morphological System

2.1 Construction and lexical sources

The FST lineage begins with **ThamizhiMorph**, the open-source Tamil morphological analyzer and generator implemented with Foma by Sarveswaran, Dias and Butt (2021). Its meta-morph rule design was presented earlier by the same authors (2019). The present work builds directly on the released ThamizhiMorph FST models and lexicons, while substantially expanding and revising lexical coverage, class structure, productive rules, ambiguity handling, generation behavior and regression tests. In the original sources, nouns were organized into 16 inflection classes and base verbs into 20 class paths, counting classes 6.1, 6.2 and 6.3 separately. Our system has **23 noun class or subclass paths** and **24 base or irregular verb paths**, in addition to **70 productive auxiliary continuations**. The corresponding explicit inventories grew from **26,343 distinct noun roots to 127,311**, and from **2,850 distinct base-verb lexical strings to 11,399**. The released tokenizer contains **139,899 lemma entries** across all lexical categories. These extensions should not obscure the origin of the finite-state implementation.

The expanded lexical inventory draws on three headword sources: the University of Madras *Tamil Lexicon*, digitized and hosted by the University of Chicago's Digital South Asia Library; official Tamil Wiktionary title and page dumps; and the supplemental Vuizur Wiktionary-Dictionaries Tamil export. After normalization, the audited snapshots contain **106,486 Tamil Lexicon lemmas**, **98,104 Tamil Wiktionary lemmas** and **5,509 Vuizur lemmas**. Because the sources overlap, their deduplicated union is **129,396 lemmas**, with **76,518 attested in at least two sources**.

These word lists were used to expand the lexicon, define and refine grammatical classes, and develop rules for generating inflected forms. The release records the exact versions of these lexical resources, verifies that the files have not changed, and documents their licenses and redistribution restrictions.

2.2 What the system contains and generates

The system contains **12 FST models** for nouns, verbs, chained verbs, adjectives, adverbs, pronouns, entities and smaller word classes. The released model has **139,895 root lemmas**. Its two largest groups are **127,311 noun roots** and **11,399 verb roots or stems**. Some spellings occur in more than one group because the same word can have both noun and verb readings.

Nouns are divided into **23 classes and subclasses** according to how their stems change when endings are added. Verbs are divided into **24 classes and subclasses**. The three largest parts of the system have the following scale:

Part of the system	Lexical and rule inventory	Grammatical analyses	Written forms
nouns	**127,311 roots** in **23 classes and subclasses**	**70,014,090**	**66,340,679**
single verbs	**11,399 roots or stems** in **24 classes and subclasses**	**164,917,397**	**152,589,463**
chained verbs	**20 following verbs** and **70 allowed ways to continue a chain**	**2,456,063,902**	**2,186,215,210**

The chained-verb figures are counts from that model by itself. They include overlap with other models and intermediate forms used to build longer words, so they cannot be added directly to the noun and single-verb rows.

The noun vocabulary was also studied through **38 word-ending families**. Long, specific endings were highly predictable: **99.9% of 2,969 nouns ending in -னம்** and **99.8% of 1,743 nouns ending in -ியம்** followed the expected pattern. All **522 nouns ending in -ிப்பு**, all **218 ending in -ைப்பு** and all **83 ending in -வியல்** also followed their expected patterns. Shorter endings were less reliable: only about **75%** of words ending in இ and **73%** of words ending in short உ followed the expected noun pattern. This shows why a word's ending is useful evidence but cannot by itself determine the correct class.

Tamil can also combine several verbs inside one written word. The system uses **20 common auxiliary or light verbs** and **70 continuation patterns**. These patterns are stored in **47 groups**, allowing verbs that support the same combinations to share rules. Before tense and person endings are added, the rules allow **4,913,950 combinations of a starting verb, a connector and a following verb**.

After overlap and intermediate-only forms are removed across the complete system, chained verbs still account for more than **1.9 billion** forms, while nouns, single verbs and the other basic models together account for about **205.5 million**.

The smaller models are also substantial:

Model	Base words	Grammatical analyses	Written forms
standalone adjectives	**662**	**3,081**	**3,080**
adverbs	**932**	**4,592**	**4,584**
pronouns	**44**	**1,723**	**1,594**
particles and other function words	**414**	**2,568**	**2,471**

The adjective and noun models divide their work to avoid duplication. The adjective model handles words that are directly classified as adjectives and forms built from adjective bases. Adjective-like forms built from nouns are handled by the noun model because it already knows how each noun stem changes. **17 singular and 20 plural noun patterns** generate forms ending in -ஆன and -அற்ற, together with related noun forms. These add about **1,011,524 valid modifier analyses** outside the standalone adjective model. Removing the same noun-based rules from the adjective model eliminated duplicate analyses without losing any generated forms.

The adverb model is similarly careful. It contains **771 words with direct adverb readings** and **932 base words** when words that can also act as nouns, adjectives, pronouns or verbs are included. It does not classify every word ending in -ஆக as an adverb. Some such words describe a role or purpose, while others belong to verb constructions. Preserving these differences avoids incorrect analyses, even if more unsupported words must use the fallback system.

The **129,020-entry source lemma dictionary** is a separate coverage checklist, not the complete model vocabulary. The model vocabulary also contains reviewed roots from its

grammatical-class inventories and lemmas needed to represent compound words as known parts.

Source-dictionary coverage	Lemmas
directly recognized by at least one FST	**102,726**
not directly recognized by any FST	**26,294**
total source-dictionary checklist	**129,020**

The unrecognized group of **26,294 lemmas** is mostly historical or rare material, names, noisy dictionary entries and words whose grammatical class remains uncertain. It may also contain genuine coverage gaps. We do not automatically place every unmatched word into a noun or verb class, because one unsupported class assignment could generate a large family of incorrect forms. Such input is still preserved exactly through the tokenizer's grapheme and byte fallback, but it does not receive an invented morphological analysis.

After duplicate spellings across the models are counted only once, the complete system accepts **2,130,878,180 distinct written forms** and **2,107,907,215 grammatical analyses**. This count excludes forms that exist only as intermediate steps for joining another ending. It does not mean that Tamil has **2.13 billion** ordinary dictionary words. It shows how fewer than **140,000 root lemmas** can produce billions of inflected and multi-verb forms.

The same spelling can have more than one analysis, and one analysis can sometimes allow more than one spelling:

- **One spelling, two analyses:** மரத்தால் can be analyzed as the noun மரம் with the meaning “by or with the tree,” or as a conditional verb form of மர, depending on the sentence.
- **One analysis, two spellings:** the analysis சுவர் + noun + accusative can be written as either சுவரை or சுவற்றை.

When the lemma is ignored, the system uses **36,058 grammatical patterns** after final Sandhi (ஒற்றெழுத்து) sound-linking markers are removed, or **46,846** when those markers are retained. Counting the same pattern separately in each model gives **54,263 model-specific patterns**. Exact reconstruction requires **94,569 spelling-sensitive patterns**. These compact pattern inventories allow the tokenizer to represent the full system without assigning a separate vocabulary entry to every generated word.

2.3 Ambiguity and validation

The analyzer preserves distinct valid readings. A deterministic context-free ranker supplies a default where needed, but candidate analyses remain available. We treat analysis recall, invalid-analysis rate and best-reading accuracy as separate quantities.

Validation includes FST build regressions, forward and inverse probes, paradigm generation, exact-pattern witnesses and source-lemma audits. The current release reports **574,379 regenerated paradigm probes with zero failures** and **94,569 exact realization witnesses with zero round-trip failures**. These are structural tests, not a claim of complete linguistic accuracy.

2.4 Development corpus audits

Several corpora were also inspected while developing and auditing morphology coverage. The general-text audit used the full Mozhi Tamil corpus, two pinned samples of the verified Tamil portion of Sangraha (Khan et al., 2024), the training splits of Dravidian CodeMix and TamilTech-QA. The train and development splits of UD Tamil-TTB (Ramasamy and Žabokrtský, 2012) supported a separate lemma, part-of-speech and feature diagnostic, while the Tamil training split of Naamapadam (Mhaske et al., 2023) was used to stress-test entity handling. These resources exposed coverage gaps and supplied regression candidates. Because they were inspected during development, none is treated as held-out evidence of tokenizer or translation quality.

3. Reversible Semantic Tokenization

BPE-based tokenizers represent rare words through subword units learned from frequency statistics (Sennrich et al., 2016). SentencePiece supports BPE and unigram tokenization directly from raw text (Kudo and Richardson, 2018). These approaches are general, fast and widely supported.

Morphology-aware and factored tokenizers instead supply a model with information such as lemma, part of speech and inflectional features. Our representation differs from a lossy morphological segmenter in two respects. It explicitly identifies lexical and grammatical features across different word classes, rather than only marking morphological boundaries, and its public codec can reconstruct the exact original UTF-8 input.

3.1 Semantic stream

An analysis is mapped deterministically to lemma and feature tokens. For example:

மரங்களை
மரம் <POS_NOUN> <NUM_PL> <CASE_ACC>

படித்துக்கொடுத்தான்
படி <LINK_VPART> கொடு <TENSE_PAST> <PERSON_3SG_MASC>

The same grammatical tokens are shared across words even when Tamil applies different spelling rules. The following nouns all have accusative case, but their stems change in different ways when the ending is added:

Written word	Semantic token output	Visible spelling change
பூவை	பூ <POS_NOUN> <CASE_ACC>	வ் is inserted before the ending
காட்டை	காடு <POS_NOUN> <CASE_ACC>	the final உ disappears and ட doubles
வண்டை	வண்டு <POS_NOUN> <CASE_ACC>	the final உ disappears without doubling
ஆற்றை	ஆறு <POS_NOUN> <CASE_ACC>	final று changes to ற்று
மரத்தை	மரம் <POS_NOUN> <STEM_OBLIQUE> <CASE_ACC>	final ம் changes to த்து

The same principle applies to verbs. These words have different written past stems, but they share the tokens for past tense and third-person singular masculine agreement:

படித்தான் படி <TENSE_PAST> <PERSON_3SG_MASC>
விட்டான் விடு <TENSE_PAST> <PERSON_3SG_MASC>
பெற்றான் பெறு <TENSE_PAST> <PERSON_3SG_MASC>
சென்றான் செல் <TENSE_PAST> <PERSON_3SG_MASC>
கொண்டான் கொள் <TENSE_PAST> <PERSON_3SG_MASC>

The lemma identifies the word, while the shared tokens state its grammatical meaning. The FST knows the spelling rules for each lemma's class, so the semantic stream does not need a different past-tense or accusative token for every written stem change. The exact codec described below records any extra choice needed when the same analysis permits more than one valid spelling.

The fixed vocabulary contains **140,922 tokens**, including **222 grammatical and semantic labels**, **139,957 lexical tokens**, **378 Tamil grapheme fallback tokens** and **256 UTF-8 byte tokens**. Among the lexical tokens, **58** are used as secondary lemmas in current multi-lemma and auxiliary analyses. They remain lexical states rather than forming a special semantic token class. Typed links distinguish verbal-participle, infinitive and other compound relations instead of replacing them with a generic auxiliary marker.

3.2 Exact realization

The tokenizer first chooses a morphological analysis for the input word. The FST can usually turn that analysis back into the expected spelling. When the same analysis allows more than one valid spelling, the tokenizer records a small marker that tells the decoder which spelling appeared in the original text. Spaces, punctuation and text that the FST does not recognize are also recorded directly. The goal is always

$$\text{decode}(\text{encode}(x)) = x$$

for any UTF-8 text accepted by the tokenizer. The decoder must be able to restore the text from the tokens alone, without looking at the original input.

The fallback hierarchy is:

1. direct FST analysis;
2. reviewed entity analysis;
3. constrained semantic paths where enabled;
4. fixed Tamil grapheme units;
5. UTF-8 bytes.

Fallback and unknown IDs are not equivalent. A grapheme or byte sequence can retain the complete input even when no morphology analysis exists.

3.3 How morphology coverage affects the token stream

The tokenizer can provide lemma and grammatical tokens only when an FST recognizes the word. When it does not, the fallback system still preserves the original spelling exactly, but it usually needs more token positions and provides less direct grammatical information. The opening line of the Thirukkural gives a compact example of this difference.

An early release recognized only **four of its seven words**. It represented the line with **95 tokens**, including **75 byte tokens** used mainly to preserve the unrecognized text. A wider coverage review found missing words and rules that also affected other Tamil text. After those problems were fixed, the FSTs recognized **all seven words**. The exact stream fell to **41 tokens**, and its **six remaining byte tokens** represented spaces rather than unrecognized Tamil words.

This example shows why FST coverage affects both the information in the stream and its length. It also shows a limit of the approach: even after repair, the semantic stream was longer than the **nine tokens** produced by a compact BPE tokenizer. One sentence cannot establish overall efficiency, so the later experiments measure sequence length across the complete evaluation data.

3.4 Public and model-facing views

The project distinguishes:

- human-readable semantic tokens;
- the exact reversible codec;
- diagnostic analysis records;
- the translation language model adapter;
- compact model-internal composer states.

Model-only compaction removes standalone `WORD_START` key/value positions and moves reconstruction-only `SURFACE_BYTES` markers into typed metadata. It elides a generic feature only when a more specific feature provably implies it. POS and semantically informative deictic information remain explicit. These optimizations do not alter the public readable representation.

4. Hierarchical word composition

The Transformer performs context-dependent attention over a sequence (Vaswani et al., 2017), and successive layers can learn hierarchical patterns.

Our flat morphology stream expands each source word into a lemma and multiple grammatical tokens. This lengthens the model-facing input and makes sentence-level global attention more expensive. Our word-composer compensates for this verbosity by retaining lexical states directly while summarizing grammatical information. After the encoder has processed the sentence, the composer uses the contextualized summary to retrieve relevant information from the original fine-grained grammatical factors before passing the resulting representation to the decoder.

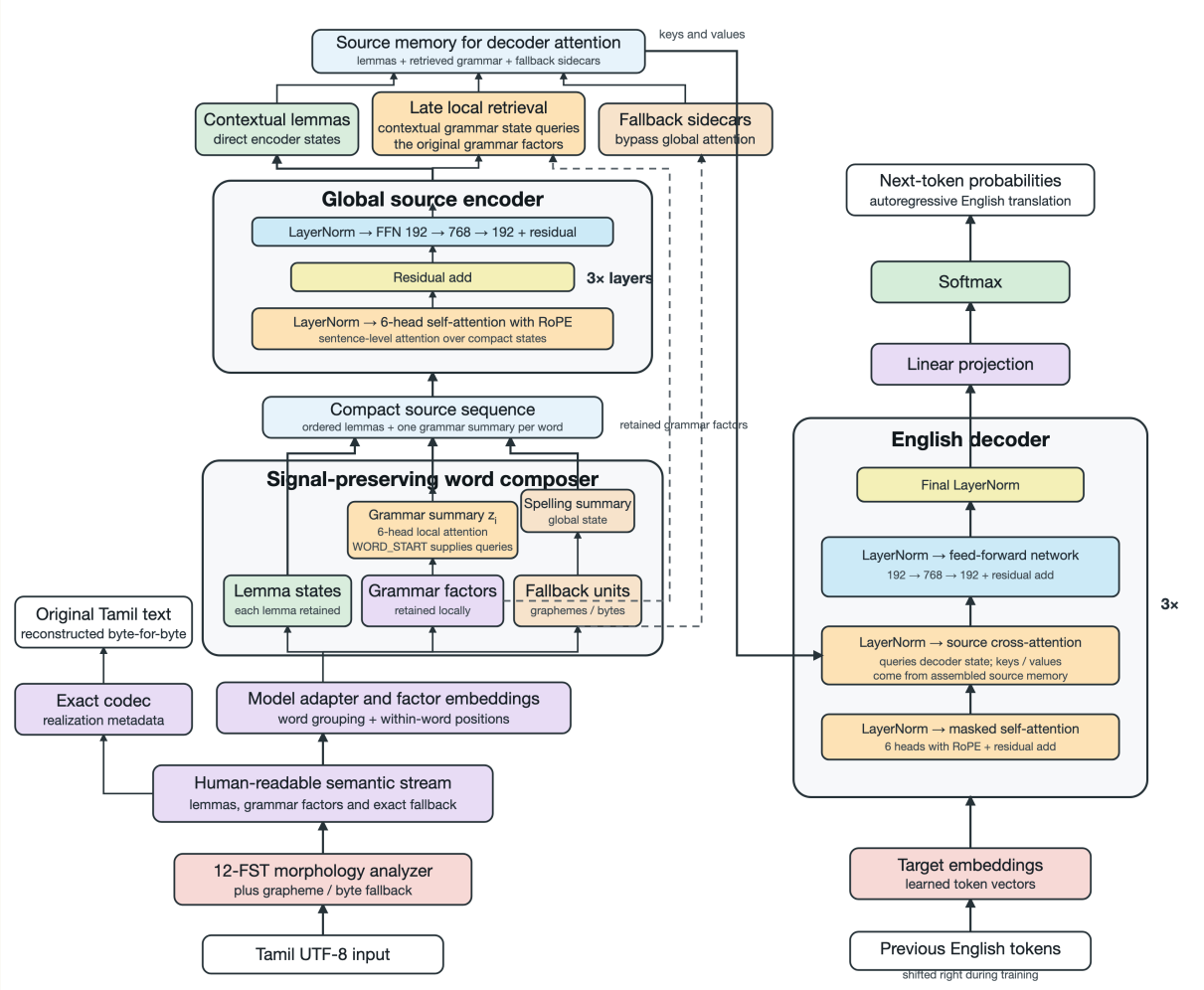

Figure 2. The tokenizer keeps a public, human-readable stream that can reconstruct the Tamil input exactly. The model adapter reorganizes the same information for translation: lemmas remain direct sentence-level states, grammar is summarized and later retrieved from the original factors, and fallback spelling units remain directly available to the decoder.

4.1 Signal-preserving composer

For an analyzed word i, let x_{ij} be the vector for its jth tokenizer factor and let F_i be the number of factors in that word. The flat model passes every factor through sentence-level attention, so the sentence has

$$N = \sum_i F_i$$

source positions, ignoring the beginning and end markers for simplicity.

The composer divides the factor indices into two sets. L_i contains the lemma positions, and G_i contains the grammatical-factor positions. Every lemma vector x_{ij} for which $j \in L_i$ is kept as a separate sentence-level state. A chained-verb analysis can therefore retain several lemmas in their original order.

The grammatical factors are replaced at the sentence level by one summary vector. No gram-

mar summary exists yet when this initial local attention is performed. To create it, the model uses the learned <WORD_START> embedding, together with its within-word position information, as the seed vector s_i. Each of the six attention heads turns this seed into one query. The grammar factors of the same word provide the keys and values:

$$
\begin{aligned}
q_i^h &= W_Q^h s_i, \\
k_{ij}^h &= W_K^h x_{ij}, \quad j \in G_i, \\
v_{ij}^h &= W_V^h x_{ij}, \quad j \in G_i, \\
a_{ij}^h &= \mathrm{softmax}_j\left(\frac{q_i^h \cdot k_{ij}^h}{\sqrt{d_h}}\right), \\
z_{i,\mathrm{gram}}^h &= \sum_{j \in G_i} a_{ij}^h v_{ij}^h.
\end{aligned}
$$

Here h identifies the attention head; q_i^h is that head's query for word i; and k_{ij}^h and v_{ij}^h are the key and value made from grammar factor j. The weight a_{ij}^h measures how much that factor contributes to the head's summary $z_{i,\mathrm{gram}}^h$. The model computes these scores only between the seed-derived query and grammatical factors from the same word. It does not compare the factors with one another or with factors from other words at this stage.

The six head summaries are joined and passed through an output projection, residual connection, normalization and feed-forward network to produce one grammar state z_i. With $d_{\text{model}} = 192$, each of the six heads has $d_h = 32$ channels. The compact representation of an analyzed word is therefore

$$C_i = \big(x_{ij} : j \in L_i\big) \,\|\, z_i,$$

where $\|$ means that the retained lemma states and grammar summary are placed next to one another in the output sequence. More precisely, the word-composer places all lemma states first, in their original order, and then places one grammar summary for the entire word. It does not alternate each lemma with a separate summary.

For example, the semantic stream for படித்துக்கொடுத்தான்,

படி <LINK_VPART> கொடு <TENSE_PAST> <PERSON_3SG_MASC>

becomes two direct lemma states followed by one grammar state:

[படி] [கொடு] [z_i]

Here z_i summarizes <LINK_VPART>, <TENSE_PAST> and <PERSON_3SG_MASC>. The original within-word position embeddings remain part of the lemma and grammatical-factor vectors, allowing the local attention to distinguish their original locations.

All three global encoder layers then process the compact sentence. The lemma states exchange information with the rest of the sentence without first being compressed into the grammar summary. The encoder also adds sentence context to the grammar-summary position. Only at this later stage does the contextualized grammar summary become a query. It queries the original fine-grained grammar factors of its own word, replacing the earlier summary with a context-sensitive one for the decoder. This late retrieval allows sentence context to affect which

original factors receive the most attention, without restoring every factor as a global source position.

The initial summary and late retrieval use the same local attention and feed-forward parameters, which are also shared with the first global encoder layer. Local dropout is zero. The matched global encoder and decoder dropout remain 0.1. This is not additive feature packing: the original grammar factors remain separate vectors during both local attention operations.

Fallback words follow a separate path. One spelling summary participates in global sentence attention, while the original grapheme or byte vectors remain available directly to decoder cross-attention. They are not treated as grammatical factors or forced through the grammar summary.

If P is the number of retained lemma states, grammar summaries, fallback summaries and sentence markers, then $P < N$ for the compact sequence. The leading attention cost changes from $O\left(\left(\sum_i F_i\right)^2\right)$ for flat sentence attention to $O\left(\sum_i F_i\right) + O(P^2)$ for local composition and retrieval plus global attention. This simplified comparison leaves out projections, feed-forward networks, padding, decoder cross-attention and decoding.

4.2 Estimated computation

Wall-clock measurements on a shared laptop are sensitive to background load, thermal throttling and other activity. We therefore use an approximate floating-point operation count to compare the two architectures. The estimate uses unpadded examples at the recorded mean training lengths: **80.77 flat source positions**, **32.44 composer global states**, **52.74 word-composer decoder-memory states** and **29.55 target tokens**. It includes the word-composer's initial word-local attention and late retrieval. One multiplication and one addition are counted as two floating-point operations.

Estimated work per average example	Morphology-flat	Composer	Composer reduction
source encoder, forward pass	0.229 GFLOPs	0.130 GFLOPs	**43%**
complete training forward pass	0.546 GFLOPs	0.432 GFLOPs	**21%**
training forward and backward, approximate	1.64 GFLOPs	1.30 GFLOPs	**21%**
30-token generation with the current uncached decoder	5.71 GFLOPs	5.21 GFLOPs	**9%**
30-token generation with decoder key/value caching	0.55 GFLOPs	0.44 GFLOPs	**21%**

The estimate suggests that the composer requires less arithmetic overall, despite performing two word-local attention operations. Its largest saving is in the source encoder, where fewer states participate in sentence-level attention.

Fewer operations do not automatically produce shorter wall-clock time. The composer currently uses many small variable-length operations, indexing and memory copies, while the flat model uses larger regular matrix multiplications that hardware libraries execute efficiently. The present decoder also recomputes the complete English prefix at every generation step. The FLOP counts therefore describe the architecture's computational potential, not a measured

latency guarantee. A controlled benchmark and a fused local composer implementation are needed before making claims about end-to-end speed.

4.3 What the architecture tests taught us

The final composer and model architecture was not chosen from intuition alone. We tested several ways of shortening or enriching the word representation.

An earlier prototype compressed every word into one content-independent summary and replaced one global encoder layer. It trained faster, but on the **69,591-pair run** it lost **1.03 BLEU, 1.56 chrF++ and 0.0152 COMETKiwi** relative to flat morphology. This failure motivated composer's direct lemma path, separate grammar state, late retrieval and full encoder depth.

We then trained matched versions of the composer with and without late retrieval. Removing retrieval increased development cross-entropy by **1.93%** and reduced chrF++ by **1.58%**. Directly bypassing retrieval in trained models raised loss by about **17-22%**, depending on the diagnostic and checkpoint. The model therefore uses the second look at the original grammar factors; it is not an unused extra operation.

Moving retrieval earlier, between encoder layers two and three, did not help. At **20,000 updates**, this model increased cross-entropy from **2.6793 to 2.7180** and reduced chrF++ from about **45.55 to 45.20**. The final encoder layer appears to mix away some of the newly retrieved detail, so our current composer instead retrieves immediately before decoder access.

We also tested two grammar summaries. We used a hand-written split between factor groups. It preserved the assigned information, but the second summary was sparse and weakly used: only **8.92% of source words** received it, **44.91%** of those summaries contained one factor, and removing the second state raised causal loss by only **0.57%**. Its small **40,000-update** development-score lead did not come with a demonstrated subgroup benefit, and mean global states rose from **32.44 to 33.80**.

A second experiment let two learned queries inspect every grammar factor instead of imposing a hand-written split. The two states became increasingly similar. On a measure where 1 means that two vectors point in the same direction, their mean similarity reached **0.929** after late retrieval. Removing the second state raised loss by only **0.54%**, while using two states increased the global source sequence by **37.8%** on the diagnostic set. This design was stopped at **20,000 updates**.

These negative results narrow the claim. One summary is not guaranteed to be perfect, but the tested second summaries added positions without showing enough distinct, useful work. Our proposed composer is therefore the best supported compact architecture in this study.

5. Data and experimental design

5.1 Training data

We first tested our data-cleaning and review process on fixed samples of **1,000 and 100,000** Tamil–English sentence pairs from **Samanantar** (Ramesh et al., 2022). This testing revealed

inconsistent text formatting, repeated English translations, number mismatches and incorrectly aligned sentence pairs. We therefore did not include any Samanantar pairs in the final **69,591-pair** training dataset.

The final training union contains **69,591 Tamil–English pairs**. Its **22,955 authentic pairs** comprise **18,000 BPCC-Human Wiki pairs**, **4,000 BPCC-Human Daily pairs** (Gala et al., 2023), and **955 sentence pairs** aligned from corresponding Tamil and English Press Information Bureau releases. The remaining **46,636 pairs** begin with known-original English sentences from two non-overlapping tranches of WikiText-103 (Merity et al., 2017). The pinned **IndicTrans2 English-to-Tamil model** (Gala et al., 2023) generates their Tamil sources. The first tranche contributed **19,409 reviewed pairs** to the earlier **42,364-pair treatment**. For the second tranche, bidirectional COMETKiwi and structural checks filtered **35,000 candidates to 27,234**.

A deterministic blinded Codex review sampled **200 rows**: **194 were rated good** and six partial. The six sampled partials and one exact prior-source collision were removed. The sample estimates the gate's selectivity; it neither proves every unreviewed row correct nor constitutes bilingual human evaluation. Synthetic Tamil may reproduce the teacher's constructions and biases.

The authentic **2,000-row BPCC-Human Wiki development set** is excluded from training. Exact source, target and pair overlap checks are zero.

5.2 Protected evaluation

The final test contains **1,024 IN22-Gen**, **1,503 IN22-Conv** and **1,012 FLORES+ devtest rows**. IN22-Gen and IN22-Conv were introduced with IndicTrans2 and BPCC (Gala et al., 2023). FLORES+ derives from the multilingual FLORES evaluation lineage; FLORES-style sets use professionally translated parallel sentences at fixed split sizes (NLLB Team et al., 2022). The exact dataset revisions and files are checksum-pinned.

The **3,539 rows** were opened once after checkpoints, tokenizers, decoder settings and analysis procedures were frozen. Dataset fingerprints, row order, checkpoint hashes, canonical tokenizer/FST provenance, zero truncation and bit-exact checkpoint reloads were verified. FLORES+ `dev` remained unopened.

5.3 Arms and controls

The five primary systems are **morphology-flat**, the signal-preserving **word-composer**, **Sarvam-1, AI4Bharat IndicBERTv2 and BrahmicTokenizer-131K**. The frozen evaluation also retains the earlier single-summary composer as a historical architecture ablation.

The external comparison set uses the tokenizer artifacts from Sarvam-1, AI4Bharat IndicBERTv2 and BrahmicTokenizer-131K. These artifacts are used only as source tokenizers inside our randomly initialized translation model. Our results do not evaluate the organizations' complete pretrained systems. IndicBERTv2 is described with the IndicCorp and IndicXTREME resources by Doddapaneni et al. (2023).

Every system uses **18,967,296 parameters**, **width 192**, **6 attention heads**, **FFN width 768**, **3 decoder layers**, a **16,000-token English target tokenizer**, AdamW at **learning rate 0.0007**,

batch size 24, **40,000 updates** and seed 20260731. **RoPE** is applied to encoder and decoder self-attention; cross-attention remains unrotated. Training rows, order, target IDs, optimizer, schedule, numeric channel and greedy decoding are shared.

The final models use rotary position embeddings. RoPE applies position-dependent rotations to queries and keys so attention contains relative-position information (Su et al., 2021). We selected it through a matched ablation rather than assuming it would transfer unchanged to a morphology-factor sequence.

Morphology-flat and the external arms use three global encoder layers. The signal-preserving composer retains three global encoder layers and adds its declared composition paths while preserving the matched total parameter budget of 18.97-million. The historical ablation used a composer layer plus two global layers.

The shared numeric channel replaces aligned source–target number occurrences with ordered slots and restores exact strings after generation. The headline decoder requires every source-present slot before EOS. This prevents basic number copying from becoming the primary tokenizer advantage, although forced coverage can extend an otherwise degenerate output.

5.4 Metrics

We report SacreBLEU, chrF++, exact match and teacher-forced target cross-entropy. chrF++ includes character and word n-gram overlap (Popović, 2017). COMETKiwi is a reference-free quality estimator based on source and hypothesis rather than the gold target (Rei et al., 2022).

We use **fertility** in its conventional word-normalized sense. If a corpus contains W orthographic words, a subword tokenizer with T model tokens has token fertility T/W. For morphology-flat, the corresponding quantity is **serialized factor fertility**, N/W, because the model-facing positions are lemmas and grammatical factors rather than ordinary subword tokens. For the composer, **global-state fertility**, P/W, counts the states passed to sentence-level attention. We report **positions per grapheme** when normalizing by Tamil grapheme clusters and **graphemes per position** for its inverse. “Tokens per character” is avoided because Unicode code points, UTF-8 bytes and visible Tamil graphemes are not equivalent.

Preservation measures include number recall/precision, required-slot recall, a conservative capitalized-reference entity-token proxy, Tamil leakage and repeated trigrams. Efficiency measures include serialized and global source positions, updates per second, wall time, peak RSS and decoding throughput. Paired intervals use 10,000 row-level bootstrap resamples. They measure test-row uncertainty for one trained seed, not training-seed variation.

6. Results

6.1 Development evidence

On the **69,591-pair development comparison**, composer and flat were tied:

Arm	BLEU	chrF++	COMETKiwi	Dev. CE
composer	21.85	**48.76**	0.7050	2.4458
morphology-flat	**22.69**	48.68	**0.7063**	**2.4249**
AI4Bharat	21.47	47.52	0.6951	2.5807
Sarvam	20.80	46.51	0.6842	2.5527
Brahmic	20.47	46.12	0.6782	2.5343

Composer-minus-flat sentence-chrF++ was **+0.185 with a 95% interval of [-0.262, +0.634]**; COMETKiwi was **-0.00123** with **[-0.00483, +0.00235]**. This justified carrying both systems to the protected test. Composer reduced mean global semantic states from **80.77 to 32.44**. The historical single-summary ablation scored **21.65 BLEU, 47.12 chrF++ and 0.6911 COMETKiwi**; its role is architectural diagnosis rather than a proposed system.

A separate 12,000-update positional ablation compared learned absolute, scaled fixed sinusoidal and rotary positions. RoPE improved development loss, BLEU, chrF++, COMETKiwi and repetition behavior for morphology-flat, the morphology composer and AI4Bharat. We therefore used one positional system, RoPE, for the final study.

After the protected comparison was complete, we ran development-only architecture tests to understand composer rather than to revise the protected result. They produced four main findings:

- models trained without late retrieval were worse for both one- and two-summary composers;
- moving retrieval before the final encoder layer was worse than retrieving after all encoder layers;
- a hand-written two-summary split preserved information, but the decoder made little use of the second state;
- two learned all-factor summaries also became largely redundant and required **37.8% more global states** than one summary on the diagnostic set.

6.2 Protected quality

Arm	Loss	BLEU	chrF++	COMETKiwi
morphology-flat	3.8550	**10.63**	**35.26**	**0.6276**
composer	3.8519	10.30	34.88	0.6241
AI4Bharat	4.0137	9.92	34.16	0.6119
Brahmic	3.8871	9.45	33.55	0.5923
Sarvam	3.8794	9.24	33.40	0.5985

Morphology-flat leads all three generation metrics. The omitted historical ablation had the lowest teacher-forced loss, **3.8396**, but only **10.01 BLEU, 34.35 chrF++ and 0.6156 COMETKiwi**. This illustrates why target cross-entropy is diagnostic rather than the headline outcome.

Composer-minus-flat has a pooled sentence-chrF++ difference of **-0.316 [-0.591, -0.037]** and COMETKiwi difference **-0.00343 [-0.00636, -0.00062]**. Composer nevertheless exceeds AI4Bharat by **+0.743** sentence-chrF++ and **+0.01226** COMETKiwi, Sarvam by **+1.492** and **+0.02559**, and Brahmic by **+1.649 and +0.03184**. All corresponding paired intervals exclude zero. Against the historical single-summary ablation, composer gains **+0.461** sentence-chrF++ and **+0.00856** COMETKiwi.

The domain breakdown explains the pooled result:

Arm	IN22-Gen chrF++ / COMETKiwi	IN22-Conv	FLORES+ devtest
morphology-flat	**36.07 / 0.6229**	**27.21 / 0.6137**	**39.08 / 0.6530**
composer-v2	35.88 / **0.6237**	26.95 / 0.6106	38.41 / 0.6446

The paired intervals include zero on both IN22 partitions. FLORES+ carries the measurable deficit: composer-minus-flat sentence-chrF++ is -0.754 [-1.258, -0.248], and COMETKiwi is -0.00836 [-0.01285, -0.00372].

6.3 Preservation and degeneration

Arm	Number recall	Number precision	Entity proxy	Repeated-trigram rows
morphology-flat	**93.97%**	83.60%	**39.53%**	**250**
composer	**93.97%**	**88.33%**	39.38%	274
AI4Bharat	92.90%	84.33%	37.79%	280
Sarvam	93.26%	86.13%	35.58%	314
Brahmic	93.62%	82.49%	35.74%	313

Every arm achieved **100% recall of required numeric slots**. Surface-number scores remain lower because unmatched literals and additional generated numbers are possible. No arm emitted Tamil script in its English output. The entity measure is a conservative exact-token proxy, not full entity evaluation.

6.4 Representation length and recorded runtime

Arm	Serialized source positions/row	Global encoder states/row	Recorded rows/s	Peak RSS
morphology-flat	71.48	71.48	4.12	6,544 MiB
composer	71.48	**29.08**	3.15	7,254 MiB
AI4Bharat	**25.45**	25.45	4.50	6,754 MiB
Sarvam	32.88	32.88	3.27	**6,135 MiB**
Brahmic	45.52	45.52	**6.14**	6,228 MiB

Composer reduces **252,979 protected raw factors to 102,897 global states** (by **59.3%**). Our training and inference runs were not repeated under controlled system load, so they do not establish comparable wall-clock speeds. The operation-count estimate in Section 4.2 instead predicts fewer FLOPs for composer.

6.5 Evaluation-only vocabulary IDs

No tokenizer emitted its unknown ID. Each frozen dense map nevertheless encountered native IDs absent from training:

Arm	Distinct IDs	Occurrences	Rows affected
morphology	298	423	344 (9.72%)
AI4Bharat	366	627	527 (14.89%)
Sarvam	63	230	164 (4.63%)
Brahmic	82	112	78 (2.20%)

These IDs use initialized but untrained embedding rows. This is preferable to collapsing the surface to <unk>, but it remains a generalization weakness. Future systems should initialize unseen lemma or grapheme rows from spelling, features or lexical priors.

6.6 Qualitative pattern

The frozen AI review found that composer often handles short conversational constructions and common paraphrases well. Flat more often preserves a specific lexical item, proper name, organization or long formal clause. Composer can turn compressed factors into a plausible but incorrect lexical choice. Difficult long inputs expose omissions and occasional repetition in both small systems. The composer's one grammar summary can still be a bottleneck despite its direct lexical path. However, the tested two-summary replacements did not solve that problem: they added weakly used or redundant states.

7. Discussion

7.1 What the comparison supports

Morphology-flat versus each external flat arm is the cleanest tokenizer contrast. Under the shared model and budget, the complete morphology representation provides useful translation signal. The experiment does not attribute the gain to case, tense, lemma reuse or any single feature. A future causal ablation should compare full morphology, lemma-only, lemma-plus-POS, shuffled feature labels and a surface-only word composer.

Composer versus morphology-flat is an architecture contrast. The main result shows that direct lexical, grammar and fallback paths preserve most flat-model quality while substantially shortening global attention. The historical single-summary ablation confirms that efficiency can be purchased by discarding information.

The later architecture tests clarify what did and did not repair the compact model. Late

retrieval is useful, and placing it after all encoder layers works better than placing another encoder layer after it. Simply adding a second summary is not enough. The hand-written split produced a sparse secondary channel, while two learned all-factor summaries converged toward similar states. A future design would need a clearer source of complementary information—such as a conditional residual path for unusually complex words—and would still need to beat the one-summary composer control under matched quality and runtime tests.

7.2 Efficiency is multidimensional

Serialized factor fertility, global-state fertility, training throughput, memory and decoding speed are related but not interchangeable. Flat morphology has the best quality but the highest factor fertility—that is, the most serialized model positions per source word—and the longest global sequence. Composer substantially lowers global-state fertility and requires less estimated arithmetic. AI4Bharat uses the shortest mean source sequence, while Brahmic had the highest recorded protected row throughput. A claim of compute superiority would require repeated controlled benchmarks and matched token, wall-time or FLOP budgets rather than matched examples and updates alone.

7.3 Coverage is not one number

Exact reversibility, direct FST coverage, entity coverage, grapheme fallback, byte fallback, unknown IDs and learned embeddings are different properties. The protected evaluation demonstrates the last distinction particularly clearly: a tokenizer can preserve a valid unseen ID without giving the model a trained meaning for it.

The morphology resource should therefore continue to publish multiple measurements rather than a synthetic “tokenizer score.” Classical Tamil, names, code mixing, rare words and noisy text remain important coverage areas. Context-sensitive analysis selection also remains open.

7.4 Domain knowledge and the Bitter Lesson

Sutton's Bitter Lesson argues that, over long periods, general methods that can make use of increasing computation tend to outperform systems built around human knowledge of a particular domain. Our approach is deliberately in tension with that lesson. The FSTs, lexical classes, semantic factors and word-local attention structure all place Tamil-specific knowledge in front of the learned model.

Our results do not overturn the Bitter Lesson. They show that this knowledge helped under one deliberately small and fixed budget: **69,591 training pairs**, an **18.97-million-parameter model** and **40,000 updates**. We did not measure a scaling curve. A much larger general model with much more Tamil data might learn the same regularities from use, and the advantage of the explicit representation might shrink or reverse. The external arms also test tokenizer interfaces inside our small model, not the complete pretrained systems from which those tokenizers came.

There are nevertheless reasons to test linguistic structure rather than assume that scale will always be available. Tamil is relatively data-limited; the flat result suggests that explicit factors can improve sample efficiency in that setting. Exact reconstruction, readable analyses,

controlled fallback and auditable grammar are also system properties, not merely shortcuts to a benchmark score. The word-composer asks a more scale-compatible question: whether those properties can be retained while reducing the computation spent on the expanded factor stream.

The decisive experiment is therefore not domain knowledge versus scale in the abstract. It is a scaling study across several data, parameter and FLOP budgets. Such a study should test whether morphology's quality advantage persists, narrows or reverses, and whether the composer converts any remaining advantage into lower total computation. Until then, the appropriate claim is limited: Tamil-specific structure is useful in the low-resource, small-model regime measured here, but it has not been shown to dominate general methods at scale.

8. Limitations

The study has one primary language, one translation direction, small randomly initialized models and one training seed. Bootstrap intervals measure variation over rows, not seed stability. The final test was protected from model selection, but it does not replace multi-seed replication.

The later ablations were run after the six-arm protected comparison and used development data only. They help explain the architecture, but they do not have new protected-test scores and should not be presented as protected comparisons.

The training union includes synthetic Tamil. Automatic filtering and AI review reduce obvious failures but do not replace independent bilingual evaluation. COMETKiwi is reference-free and imperfect. The entity metric is only a proxy. The required-number decoder changes generation behavior and can extend weak outputs.

The external arms test selected tokenizer artifacts inside our architecture, not complete Sarvam, AI4Bharat or Brahmic systems. Matched parameters, examples and updates do not match processed source tokens, FLOPs or wall time. Local CPU behavior may not predict optimized GPU kernels.

Finally, the FSTs are broad but incomplete. They have poor coverage on Classical Tamil. Deterministic context-free ranking can select the wrong valid reading. The exact public codec is designed for auditability and reversibility, not asserted to be the most compact possible model interface.

9. Ethics, provenance and licensing

The release must retain attribution to the upstream ThamizhiMorph models and their authors, Kengatharaiyer Sarveswaran, Gihan Dias and Miriam Butt, as well as every lexical, corpus and model source. Component licenses differ: Apache-2.0 code and tokenizer artifacts must not be conflated with CC BY-SA lexical or documentary material. Private or license-restricted word lists and audit inventories are excluded from public archives. Dataset text is released only where the corresponding license permits it; otherwise the project publishes identifiers, hashes and reconstruction scripts.

Synthetic sources are labeled as synthetic and retain their teacher model, revision and generation provenance. They should not be presented as native human-authored Tamil.

OpenAI Codex, primarily using the GPT-5.6 Sol model, served as an AI research and coding assistant. It helped implement and test software, audit data, monitor experiments, prepare figures and tables, inspect qualitative outputs and review prose. This work is credited as AI assistance, not as bilingual human annotation or paper authorship. Anand Murugan selected the research questions, approved the experimental and release decisions, reviewed and edited the article, and accepts responsibility for the published claims.

All reported training and evaluation ran locally with zero incremental paid external compute cost. Hardware and approximate wall time should accompany the release so that the resource cost remains visible. The small research translation checkpoints are not suitable for authoritative deployment. Omissions, altered relations and plausible hallucinations remain common enough to create harm if outputs are treated as reliable translations.

10. Artifacts, provenance and release

The project is divided by responsibility:

Resource	Public location
original ThamizhiMorph FST models	Open
tokenizer, codec, runtime FSTs, tests	Open
versioned morphology/FST release	Open
experiment code, configs and analyses	Open
interactive tokenizer web interface	Open
searchable grammatical and semantic label vocabulary	Open
machine-readable grammatical and semantic label vocabulary	Open

Tokenizer artifacts are frozen at `789570741ed50321753911e6d7233dd2114bcece`. The manifest calls this `0.1.0-rc10`; stale Python metadata calls it `0.1.0rc8`. Release `0.1.0-rc11` corrects metadata only; vocabulary and IDs are unchanged. Morphology is frozen at `c9fe57cca09b0c6f51178386261232bc699dcc9a` (`0.1.0-rc8`). External tokenizer revisions are:

Artifact	Frozen revision
`sarvamai/sarvam-1`	`e9607337286ddf496d4a2562b194e489dcf3feea`
`ai4bharat/IndicBERTv2-MLM-Sam-TLM`	`bb783337859e3d7957de5ba82766ddea51e8fc3e`
`theschoolofai/BrahmicTokenizer-131K`	`93df154cbc9dbf038a222c010d9b43906a8a72c3`

Protected dataset revisions are `e042ab3d3063110b1a85efa0a59bdbf8553bb928` for IN22-Gen, `18cd45870ff0a9e65df9b80dbbcc615eec0e4899` for IN22-Conv and `5fec6c13f9e5a4db2f745d4ec0d7c9721ddc4f0` for FLORES+. File-level hashes remain in the machine-readable manifest rather than being duplicated in the main text.

The public repositories contain machine-readable manifests that identify the released source commits and record checksums for the FSTs, vocabulary and token mappings. The experiment manifest also records the configuration, software environment, hardware, commands and cost of the comparison, together with checksums for the **69,591-pair training set** and all **six trained checkpoints**. The published material supports three levels of reproduction: testing the tokenizer and its exact reconstruction, recomputing the reported analysis from redistributable artifacts and aggregate evidence, or repeating the complete training procedure with the released code and configuration.

The protected IN22 and FLORES+ source sentences, reference translations and row-level predictions are not redistributed in this release. This respects the datasets' individual license conditions and preserves the evaluation boundary used in the study. The release instead records the exact dataset revisions and checksums, provides the permitted preparation and evaluation code, and publishes the aggregate results reported in this paper.

11. Conclusion

Tamil morphology can be exposed through a deterministic, auditable and byte-exact tokenizer. In the controlled experiment, the flat morphology representation produces the best protected Tamil–English translation quality, outperforming the selected external-tokenizer arms at the cost of higher serialized factor fertility: more model positions per source word and, consequently, a longer mean source sequence.

The hierarchical results refine that finding. The signal-preserving composer keeps lexical identity, grammar and fallback on separate paths, exceeds all external arms and reduces global source states by **59.3%**. It does not fully match flat on long formal sentences. The FLOP estimates predict less arithmetic though the wall-time comparison was not controlled. A historical aggressive-compression ablation confirms that removing direct lexical information loses quality. Matched controls show that late retrieval is useful. Earlier retrieval and both tested two-summary designs fail to improve the overall tradeoff, making our presented composer the best supported compact model rather than merely the first one tried.

The resulting claim is not that linguistic tokenization always wins. It is that explicit Tamil morphology provides useful reusable signal, and that tokenization and attention topology should be designed together. A hierarchy is valuable only when it preserves the distinctions it was introduced to organize.

Appendix A. Complete grammatical and semantic label vocabulary

The released vocabulary contains **222 fixed grammatical and semantic labels**. They cover grammar, relations and named-entity types. Ordinary lemmas are a separate part of the unified model vocabulary. This includes **58 words** used as secondary lemmas in multi-lemma and auxiliary analyses; they remain lexical states rather than becoming a special semantic-token class.

Token type is recorded explicitly rather than inferred from numerical position. For compatibility with the trained checkpoints, secondary lemmas retain IDs 969–1026. The nearby IDs 965–968 are **4 ordinary noun lemmas** left there by an older incremental refresh. None of these lexical entries is included in the **222-label** count or in the table below.

The table also excludes codec structure, spelling-reconstruction markers, grapheme or byte fallback, and the ordinary lemma vocabulary. The token IDs below are those of the frozen `0.1.0-rc10` stream and remain unchanged in `0.1.0-rc11`. A searchable copy is also published with the interactive tokenizer.

ID	Token	Family	Meaning
743	`<ABBREVIATION>`	grammatical or semantic feature	Marks an abbreviation.
744	`<ACTION_NOMINAL>`	grammatical or semantic feature	Marks a verb-derived noun that names an action or event.
745	`<ADJECTIVAL_PARTICIPLE>`	participle	Marks a verb form used to modify a noun.
746	`<ASPECT_PERFECT>`	aspect	Marks a completed action or a resulting state.
747	`<ASPECT_PROSPECTIVE>`	aspect	Marks an action viewed as expected or about to happen.
748	`<AUX_ATTITUDINAL>`	grammatical or semantic feature	Broad inherited FST label for an auxiliary construction that expresses the speaker's stance.
749	`<CASE_ABL>`	case	Ablative case: from, out of, or away from.
750	`<CASE_ACC>`	case	Accusative case: usually the direct object.
751	`<CASE_DAT>`	case	Dative case: usually to or for.
752	`<CASE_GEN>`	case	Genitive case: possession or an of-relation.
753	`<CASE_INST>`	case	Instrumental case: by, with, or using.
754	`<CASE_LOC>`	case	Locative case: in, at, or on.
755	`<CASE_MARKER>`	case	Broad inherited FST label indicating that a case marker is present.
756	`<CASE_NOM>`	case	Nominative or unmarked base case, often used for the subject.
757	`<CASE_SOC>`	case	Sociative case: with or together with.
758	`<CASE_TRANS>`	case	Translative or adverbial case-like form: as, becoming, or in a stated manner.
759	`<CASE_VOC>`	case	Vocative case used for direct address.
760	`<CLITIC_ADD>`	clitic	Additive clitic: also, too, or and.
761	`<CLITIC_FOCUS>`	clitic	Focus or emphatic clitic, often corresponding to தான்.

ID	Token	Family	Meaning
762	<COMPARATIVE>	grammatical or semantic feature	Marks a comparison such as than, more, or less.
763	<COMPLEMENTIZER>	grammatical or semantic feature	Introduces a quoted, reported, or embedded clause.
764	<COMPOUND_MODIFIER>	grammatical or semantic feature	Marks a noun used attributively before another word in a compound.
765	<COPULA>	grammatical or semantic feature	Marks a copular expression that links a subject with a description or identity.
766	<COP_BECOME>	grammatical or semantic feature	Marks a change into a state: become.
767	<DEGREE>	grammatical or semantic feature	Marks an amount or degree expression.
768	<DEICTIC>	deixis	General demonstrative or pointing meaning.
769	<DEICTIC_DIST>	deixis	Distal demonstrative: that, there, or then.
770	<DEICTIC_INTERROGATIVE>	deixis	Interrogative demonstrative: which, where, or when.
771	<DEICTIC_MED>	deixis	Medial demonstrative: an intermediate distance.
772	<DEICTIC_PROX>	deixis	Proximal demonstrative: this, here, or now.
773	<DEICTIC_SAME>	deixis	Marks identity or sameness: the same.
774	<DEICTIC_SITUATION>	deixis	Points to a situation or context.
775	<DEICTIC_TIME>	deixis	Points to a time.
776	<DEICTIC_TYPE>	deixis	Points to a kind or type.
777	<DERIV_AATTAM>	grammatical or semantic feature	Marks the ஆட்டம்-derived manner or likeness construction.
778	<DETERMINER>	grammatical or semantic feature	Marks a word that specifies or limits a noun.
779	<DISTRIBUTIVE>	grammatical or semantic feature	Distributive meaning: each, respective, or one by one.
780	<ENTITY_BRAND>	named entity	Named entity: brand.
781	<ENTITY_CITY>	named entity	Named entity: city.
782	<ENTITY_COUNTRY>	named entity	Named entity: country.
783	<ENTITY_ORG>	named entity	Named entity: organization.
784	<ENTITY_OTHER>	named entity	Named entity: other reviewed entity type.
785	<ENTITY_PERSON>	named entity	Named entity: person.
786	<ENTITY_PLACE>	named entity	Named entity: place.
787	<ENTITY_REGION>	named entity	Named entity: region.
788	<ENTITY_WORK>	named entity	Named entity: named creative work.
789	<EUPHONIC_AUGMENT>	grammatical or semantic feature	Marks an inserted sound used to join morphemes smoothly.
790	<EVIDENTIAL_REPORTATIVE>	grammatical or semantic feature	Marks information presented as reported rather than directly witnessed.
791	<EXISTENTIAL>	grammatical or semantic feature	Marks existence or availability.
792	<FUTURE_ADJECTIVAL_PARTICIPLE>	participle	Marks a future-oriented verb form used to modify a noun.
793	<INDEFINITE>	grammatical or semantic feature	Marks an indefinite meaning such as some or any.
794	<LETTER_NAME>	grammatical or semantic feature	Marks a spoken or written letter name.
795	<MANNER_PURPOSE>	grammatical or semantic feature	Marks a directed manner or intended outcome, often in -உமாறு.

ID	Token	Family	Meaning
796	<MEASUREMENT_UNIT>	grammatical or semantic feature	Marks a unit of measurement.
797	<MODAL>	modality	Broad modal meaning such as ability, necessity, or possibility.
798	<MODAL_MUST>	modality	Necessity or obligation: must, should, or need to.
799	<MODAL_WORTHY>	modality	Marks suitability or worthiness.
800	<MOOD_CONDITIONAL>	mood	Conditional mood: if or under a condition.
801	<MOOD_OPTATIVE>	mood	Optative mood: a wish, hope, or blessing.
802	<MOOD_PARTICLE>	mood	Marks a particle that contributes mood.
803	<MOOD_PROHIBITIVE>	mood	Negative command: do not.
804	<MOOD_QUESTION>	mood	Marks a question.
805	<MORPH_AFFIRM>	legacy FST label	Legacy FST label for affirmative meaning; retained as a fixed readable factor rather than created dynamically.
806	<MORPH_ALT>	legacy FST label	Legacy FST label for alternative form or reading; retained as a fixed readable factor rather than created dynamically.
807	<MORPH_BEN>	legacy FST label	Legacy FST label for benefactive meaning; retained as a fixed readable factor rather than created dynamically.
808	<MORPH_CMPR>	legacy FST label	Legacy FST label for comparative meaning; retained as a fixed readable factor rather than created dynamically.
809	<MORPH_CONJUNCTION>	legacy FST label	Legacy FST label for conjunction; retained as a fixed readable factor rather than created dynamically.
810	<MORPH_DEICTIC>	legacy FST label	Legacy FST label for deictic or demonstrative meaning; retained as a fixed readable factor rather than created dynamically.
811	<MORPH_EXCLAM>	legacy FST label	Legacy FST label for exclamation; retained as a fixed readable factor rather than created dynamically.
812	<MORPH_INT>	legacy FST label	Legacy FST label for intensifying or interrogative legacy label; retained as a fixed readable factor rather than created dynamically.
813	<MORPH_INTERJECTION>	legacy FST label	Legacy FST label for interjection; retained as a fixed readable factor rather than created dynamically.
814	<MORPH_INTERROGATIVE>	legacy FST label	Legacy FST label for interrogative meaning; retained as a fixed readable factor rather than created dynamically.
815	<MORPH_LIMIT>	legacy FST label	Legacy FST label for limit or restriction; retained as a fixed readable factor rather than created dynamically.
816	<MORPH_LOAN>	legacy FST label	Legacy FST label for loanword; retained as a fixed readable factor rather than created dynamically.
817	<MORPH_NEUT>	legacy FST label	Legacy FST label for neuter agreement or class; retained as a fixed readable factor rather than created dynamically.
818	<MORPH_N_PATHIL>	legacy FST label	Legacy FST label for noun-based பதில் relational construction; retained as a fixed readable factor rather than created dynamically.
819	<MORPH_OTHER>	legacy FST label	Legacy FST label for other inherited FST category; retained as a fixed readable factor rather than created dynamically.

ID	Token	Family	Meaning
820	<MORPH_PRIV>	legacy FST label	Legacy FST label for privative meaning; retained as a fixed readable factor rather than created dynamically.
821	<MORPH_PSP_ALLAAMAL>	legacy postposition	Legacy FST postposition label meaning without. It remains fixed in the released vocabulary pending a narrower named mapping.
822	<MORPH_PSP_APPAAL>	legacy postposition	Legacy FST postposition label meaning beyond or on the other side. It remains fixed in the released vocabulary pending a narrower named mapping.
823	<MORPH_PSP_APPAAL_IRUNTU>	legacy postposition	Legacy FST postposition label meaning from beyond or on the other side. It remains fixed in the released vocabulary pending a narrower named mapping.
824	<MORPH_PSP_APPURAM>	legacy postposition	Legacy FST postposition label meaning after. It remains fixed in the released vocabulary pending a narrower named mapping.
825	<MORPH_PSP_ARUKIL>	legacy postposition	Legacy FST postposition label meaning near. It remains fixed in the released vocabulary pending a narrower named mapping.
826	<MORPH_PSP_ARUKIL_IRUNTU>	legacy postposition	Legacy FST postposition label meaning from near. It remains fixed in the released vocabulary pending a narrower named mapping.
827	<MORPH_PSP_ATIYIL>	legacy postposition	Legacy FST postposition label meaning under or at the foot of. It remains fixed in the released vocabulary pending a narrower named mapping.
828	<MORPH_PSP_ATIYIL_IRUNTU>	legacy postposition	Legacy FST postposition label meaning from under or at the foot of. It remains fixed in the released vocabulary pending a narrower named mapping.
829	<MORPH_PSP_ETHIR>	legacy postposition	Legacy FST postposition label meaning opposite or against. It remains fixed in the released vocabulary pending a narrower named mapping.
830	<MORPH_PSP_ETHIRE>	legacy postposition	Legacy FST postposition label meaning opposite or facing. It remains fixed in the released vocabulary pending a narrower named mapping.
831	<MORPH_PSP_ETHIRE_IRUNTU>	legacy postposition	Legacy FST postposition label meaning from opposite or facing. It remains fixed in the released vocabulary pending a narrower named mapping.
832	<MORPH_PSP_ETHIR_IRUNTU>	legacy postposition	Legacy FST postposition label meaning from opposite or against. It remains fixed in the released vocabulary pending a narrower named mapping.
833	<MORPH_PSP_IDAIYIL>	legacy postposition	Legacy FST postposition label meaning between or among. It remains fixed in the released vocabulary pending a narrower named mapping.
834	<MORPH_PSP_IDAIYL>	legacy postposition	Legacy FST postposition label meaning between or among. It remains fixed in the released vocabulary pending a narrower named mapping.
835	<MORPH_PSP_IDAYIL_IRUNTU>	legacy postposition	Legacy FST postposition label meaning from between or among. It remains fixed in the released vocabulary pending a narrower named mapping.
836	<MORPH_PSP_ILLAAMAL>	legacy postposition	Legacy FST postposition label meaning without. It remains fixed in the released vocabulary pending a narrower named mapping.
837	<MORPH_PSP_KEEL>	legacy postposition	Legacy FST postposition label meaning below. It remains fixed in the released vocabulary pending a narrower named mapping.

ID	Token	Family	Meaning
838	<MORPH_PSP_KEELE>	legacy postposition	Legacy FST postposition label meaning below. It remains fixed in the released vocabulary pending a narrower named mapping.
839	<MORPH_PSP_KEELE_IRUNTU>	legacy postposition	Legacy FST postposition label meaning from below. It remains fixed in the released vocabulary pending a narrower named mapping.
840	<MORPH_PSP_KEEL_IRUNTU>	legacy postposition	Legacy FST postposition label meaning from below. It remains fixed in the released vocabulary pending a narrower named mapping.
841	<MORPH_PSP_KURUKKE>	legacy postposition	Legacy FST postposition label meaning across. It remains fixed in the released vocabulary pending a narrower named mapping.
842	<MORPH_PSP_KURUKKE_IRUNTU>	legacy postposition	Legacy FST postposition label meaning from across. It remains fixed in the released vocabulary pending a narrower named mapping.
843	<MORPH_PSP_MEEL>	legacy postposition	Legacy FST postposition label meaning above or on. It remains fixed in the released vocabulary pending a narrower named mapping.
844	<MORPH_PSP_MEELE>	legacy postposition	Legacy FST postposition label meaning above or on. It remains fixed in the released vocabulary pending a narrower named mapping.
845	<MORPH_PSP_MEELE_IRUNTU>	legacy postposition	Legacy FST postposition label meaning from above or on. It remains fixed in the released vocabulary pending a narrower named mapping.
846	<MORPH_PSP_MEEL_IRUNTU>	legacy postposition	Legacy FST postposition label meaning from above or on. It remains fixed in the released vocabulary pending a narrower named mapping.
847	<MORPH_PSP_MUN>	legacy postposition	Legacy FST postposition label meaning before or in front of. It remains fixed in the released vocabulary pending a narrower named mapping.
848	<MORPH_PSP_MUNNAAL>	legacy postposition	Legacy FST postposition label meaning before. It remains fixed in the released vocabulary pending a narrower named mapping.
849	<MORPH_PSP_MUNNAAL_IRUNTU>	legacy postposition	Legacy FST postposition label meaning from before. It remains fixed in the released vocabulary pending a narrower named mapping.
850	<MORPH_PSP_MUNNE>	legacy postposition	Legacy FST postposition label meaning before or in front. It remains fixed in the released vocabulary pending a narrower named mapping.
851	<MORPH_PSP_MUN_IRUNTU>	legacy postposition	Legacy FST postposition label meaning from before or in front of. It remains fixed in the released vocabulary pending a narrower named mapping.
852	<MORPH_PSP_NADUVIL>	legacy postposition	Legacy FST postposition label meaning in the middle of. It remains fixed in the released vocabulary pending a narrower named mapping.
853	<MORPH_PSP_NADUVIL_IRUNTU>	legacy postposition	Legacy FST postposition label meaning from in the middle of. It remains fixed in the released vocabulary pending a narrower named mapping.
854	<MORPH_PSP_PIN>	legacy postposition	Legacy FST postposition label meaning after or behind. It remains fixed in the released vocabulary pending a narrower named mapping.
855	<MORPH_PSP_PINNAAL>	legacy postposition	Legacy FST postposition label meaning after or behind. It remains fixed in the released vocabulary pending a narrower named mapping.

ID	Token	Family	Meaning
856	<MORPH_PSP_PINNAAL_IRUNTU>	legacy postposition	Legacy FST postposition label meaning from after or behind. It remains fixed in the released vocabulary pending a narrower named mapping.
857	<MORPH_PSP_PINNE>	legacy postposition	Legacy FST postposition label meaning after or behind. It remains fixed in the released vocabulary pending a narrower named mapping.
858	<MORPH_PSP_PINNE_IRUNTU>	legacy postposition	Legacy FST postposition label meaning from after or behind. It remains fixed in the released vocabulary pending a narrower named mapping.
859	<MORPH_PSP_PIN_IRUNTU>	legacy postposition	Legacy FST postposition label meaning from after or behind. It remains fixed in the released vocabulary pending a narrower named mapping.
860	<MORPH_PSP_PIRAKU>	legacy postposition	Legacy FST postposition label meaning after. It remains fixed in the released vocabulary pending a narrower named mapping.
861	<MORPH_PSP_POL>	legacy postposition	Legacy FST postposition label meaning like or as. It remains fixed in the released vocabulary pending a narrower named mapping.
862	<MORPH_PSP_POLA>	legacy postposition	Legacy FST postposition label meaning like or as. It remains fixed in the released vocabulary pending a narrower named mapping.
863	<MORPH_PSP_TAVIRA>	legacy postposition	Legacy FST postposition label meaning except. It remains fixed in the released vocabulary pending a narrower named mapping.
864	<MORPH_PSP_ULE>	legacy postposition	Legacy FST postposition label meaning inside. It remains fixed in the released vocabulary pending a narrower named mapping.
865	<MORPH_PSP_ULE_IRUNTU>	legacy postposition	Legacy FST postposition label meaning from inside. It remains fixed in the released vocabulary pending a narrower named mapping.
866	<MORPH_PSP_ULLE>	legacy postposition	Legacy FST postposition label meaning inside. It remains fixed in the released vocabulary pending a narrower named mapping.
867	<MORPH_PSP_ULLE_IRUNTU>	legacy postposition	Legacy FST postposition label meaning from inside. It remains fixed in the released vocabulary pending a narrower named mapping.
868	<MORPH_PSP_UL_IRUNTU>	legacy postposition	Legacy FST postposition label meaning from inside. It remains fixed in the released vocabulary pending a narrower named mapping.
869	<MORPH_PSP_VALIYAAKA>	legacy postposition	Legacy FST postposition label meaning through or by way of. It remains fixed in the released vocabulary pending a narrower named mapping.
870	<MORPH_PSP_VARAIKKUM>	legacy postposition	Legacy FST postposition label meaning until or up to. It remains fixed in the released vocabulary pending a narrower named mapping.
871	<MORPH_PSP_VARAIYIL>	legacy postposition	Legacy FST postposition label meaning until or within the limit. It remains fixed in the released vocabulary pending a narrower named mapping.
872	<MORPH_PSP_VELIYEE>	legacy postposition	Legacy FST postposition label meaning outside. It remains fixed in the released vocabulary pending a narrower named mapping.
873	<MORPH_PSP_VELIYEE_IRUNTU>	legacy postposition	Legacy FST postposition label meaning from outside. It remains fixed in the released vocabulary pending a narrower named mapping.

ID	Token	Family	Meaning
874	<MORPH_PSP_VELIYIL>	legacy postposition	Legacy FST postposition label meaning outside. It remains fixed in the released vocabulary pending a narrower named mapping.
875	<MORPH_PSP_VELIYIL_IRUNTU>	legacy postposition	Legacy FST postposition label meaning from outside. It remains fixed in the released vocabulary pending a narrower named mapping.
876	<MORPH_REGISTER>	legacy FST label	Legacy FST label for register label; retained as a fixed readable factor rather than created dynamically.
877	<MORPH_SANDHI_C>	legacy FST label	Legacy FST label for legacy c-type sandhi; retained as a fixed readable factor rather than created dynamically.
878	<MORPH_SANDHI_K>	legacy FST label	Legacy FST label for legacy k-type sandhi; retained as a fixed readable factor rather than created dynamically.
879	<MORPH_SANDHI_P>	legacy FST label	Legacy FST label for legacy p-type sandhi; retained as a fixed readable factor rather than created dynamically.
880	<MORPH_SANDHI_T>	legacy FST label	Legacy FST label for legacy t-type sandhi; retained as a fixed readable factor rather than created dynamically.
881	<MORPH_UNTIL>	legacy FST label	Legacy FST label for until or boundary meaning; retained as a fixed readable factor rather than created dynamically.
882	<MORPH_VPARTP_TAANDI>	legacy verbal relation	Legacy participial relation meaning beyond or crossing.
883	<MORPH_VPART_CUTTI>	legacy verbal relation	Legacy verbal-participle relation meaning around or concerning.
884	<MORPH_VPART_KONDU>	legacy verbal relation	Legacy verbal-participle relation meaning with, by, or while doing.
885	<MORPH_VPART_OTTI>	legacy verbal relation	Legacy verbal-participle relation meaning adjoining or in relation to.
886	<MORPH_VPART_TAANDI>	legacy verbal relation	Legacy verbal-participle relation meaning beyond or crossing.
887	<MORPH_VPART_TAVIRTU>	legacy verbal relation	Legacy verbal-participle relation meaning excluding or avoiding.
888	<MORPH_VPART_VAITTU>	legacy verbal relation	Legacy verbal-participle relation meaning using, keeping, or having done.
889	<MORPH_VPART_VIDA>	legacy verbal relation	Legacy verbal-participle relation meaning than or leaving.
890	<NEGATIVE_PARTICIPLE>	participle	Marks a negative non-finite or modifying verb form.
891	<NUM_CARDINAL>	number and quantity	Cardinal number: one, two, three, and so on.
892	<NUM_FRACTION>	number and quantity	Fractional number.
893	<NUM_ORDINAL>	number and quantity	Ordinal number: first, second, and so on.
894	<NUM_PL>	number and quantity	Plural number.
895	<NUM_SG>	number and quantity	Singular number.
896	<PART_ONLY>	grammatical or semantic feature	Restrictive particle: only or just.
897	<PERSON_1PL>	person and agreement	First person plural: we.
898	<PERSON_1SG>	person and agreement	First person singular: I.
899	<PERSON_2PL>	person and agreement	Second person plural: you (plural).
900	<PERSON_2PL_HON>	person and agreement	Second person plural honorific: respectful you.
901	<PERSON_2SG>	person and agreement	Second person singular: you.

ID	Token	Family	Meaning
902	<PERSON_2SG_HON>	person and agreement	Second person singular honorific: respectful you.
903	<PERSON_3PL>	person and agreement	Third person plural: they.
904	<PERSON_3PL_EPICENE>	person and agreement	Third person plural without a masculine/feminine distinction.
905	<PERSON_3PL_NEUT>	person and agreement	Third person plural neuter or non-human.
906	<PERSON_3SG>	person and agreement	Third person singular.
907	<PERSON_3SG_EPICENE>	person and agreement	Third person singular without a masculine/feminine distinction.
908	<PERSON_3SG_FEM>	person and agreement	Third person singular feminine: she.
909	<PERSON_3SG_HON>	person and agreement	Third person singular honorific.
910	<PERSON_3SG_MASC>	person and agreement	Third person singular masculine: he.
911	<PERSON_3SG_NEUT>	person and agreement	Third person singular neuter: it.
912	<POLARITY_NEG>	polarity	Negative polarity.
913	<POLARITY_POS>	polarity	Positive polarity.
914	<POSTPOSITION>	postposition and relation	General postposition or relational function word.
915	<POST_ABOUT>	postposition and relation	Relation meaning about or concerning.
916	<POST_ACCORDING_TO>	postposition and relation	Relation meaning according to or in the manner stated.
917	<POST_AFTER>	postposition and relation	Temporal or spatial relation meaning after or behind.
918	<POST_AMONG>	postposition and relation	Relation meaning among or between.
919	<POST_BEFORE>	postposition and relation	Temporal or spatial relation meaning before or in front of.
920	<POST_LIKE_AS>	postposition and relation	Similarity relation: like or as.
921	<POST_TOWARD>	postposition and relation	Direction relation: toward.
922	<POST_UNTIL>	postposition and relation	Boundary relation: until or up to.
923	<POST_WITHIN_BY>	postposition and relation	Interior or deadline relation: within, inside, or by.
924	<POS_ADJ>	part of speech	Part of speech: adjective.
925	<POS_ADV>	part of speech	Part of speech: adverb.
926	<POS_INTERJECTION>	part of speech	Part of speech: interjection.
927	<POS_NOUN>	part of speech	Part of speech: noun.
928	<POS_PART>	part of speech	Part of speech: particle.
929	<POS_PARTICIPIAL_NOUN>	part of speech	Part of speech: noun formed from a participle.
930	<POS_PRONOUN>	part of speech	Part of speech: pronoun.
931	<POS_QUANTIFIER>	part of speech	Part of speech: quantifier.
932	<POS_VERBAL_NOUN>	part of speech	Part of speech: verb-derived action or event noun.
933	<PRESENTATIVE>	grammatical or semantic feature	Presentative expression used to point out or introduce something.
934	<PRIVATIVE_WITHOUT>	grammatical or semantic feature	Privative meaning: without or lacking.
935	<PRON_EXCLUSIVE>	pronoun	Exclusive first-person plural: we, excluding the addressee.

ID	Token	Family	Meaning
936	<PRON_INCLUSIVE>	pronoun	Inclusive first-person plural: we, including the addressee.
937	<PRON_POSSESSIVE>	pronoun	Possessive pronoun function.
938	<PRON_REFLEXIVE>	pronoun	Reflexive pronoun function: self.
939	<QUANT_ALL>	grammatical or semantic feature	Universal quantity: all or every.
940	<RECIPROCAL>	grammatical or semantic feature	Reciprocal relation: each other.
941	<REDUPLICATION>	grammatical or semantic feature	Marks a repeated form used for distribution, emphasis, or iteration.
942	<REGISTER_COLLOQUIAL>	grammatical or semantic feature	Marks a colloquial form.
943	<REL_ATTACH>	grammatical or semantic feature	Marks an attaching or related-to construction.
944	<SANDHI_C>	sandhi	Marks c-type linking sandhi.
945	<SANDHI_K>	sandhi	Marks k-type linking sandhi.
946	<SANDHI_P>	sandhi	Marks p-type linking sandhi.
947	<SANDHI_T>	sandhi	Marks t-type linking sandhi.
948	<SEM_HUMAN>	grammatical or semantic feature	Marks reference to a human being or group.
949	<SEM_PURPOSE>	grammatical or semantic feature	Marks purpose or intended use.
950	<STEM_OBLIQUE>	grammatical or semantic feature	Marks a changed noun stem used before a case ending.
951	<TEMPORAL_IMMEDIATE>	grammatical or semantic feature	Marks immediate succession: as soon as.
952	<TEMPORAL_WHEN>	grammatical or semantic feature	Marks a time relation: when or while.
953	<TENSE_FUTURE>	tense	Future tense.
954	<TENSE_PAST>	tense	Past tense.
955	<TENSE_PRESENT>	tense	Present tense.
956	<TITLE_HONORIFIC>	grammatical or semantic feature	Marks an honorific title.
957	<VERBAL_PARTICIPLE>	verb form	Marks a non-finite verb that links to a following action.
958	<VERB_COMPLEX>	verb form	Broad inherited FST label for a complex verb construction.
959	<VERB_FINITE>	verb form	Broad inherited FST label for a finite verb.
960	<VERB_IMPERATIVE>	verb form	Imperative verb form: a command or request.
961	<VERB_INFINITIVE>	verb form	Infinitive verb form.
962	<VERB_NONFINITE>	verb form	Broad inherited FST label for a non-finite verb.
963	<VOICE_CAUSATIVE>	voice	Causative voice: causes someone or something to act.
964	<VOICE_PASSIVE>	voice	Passive voice: presents the affected participant rather than the actor.

References

- Doddapaneni, S., et al. 2023. Towards Leaving No Indic Language Behind: Building Monolingual Corpora, Benchmark and Models for Indic Languages. ACL.
- Gala, J., et al. 2023. IndicTrans2: Towards High-Quality and Accessible Machine Translation Models for all 22 Scheduled Indian Languages. *Transactions on Machine Learning Research*.
- Khan, M. S. U. R., et al. 2024. IndicLLMSuite: A Blueprint for Creating Pre-training and Fine-Tuning Datasets for Indian Languages.
- Kudo, T., and Richardson, J. 2018. SentencePiece: A Simple and Language Independent Subword Tokenizer and Detokenizer for Neural Text Processing. EMNLP System Demonstrations.
- Merity, S., Xiong, C., Bradbury, J., and Socher, R. 2017. Pointer Sentinel Mixture Models. ICLR.
- Mhaske, A., et al. 2023. Naamapadam: A Large-Scale Named Entity Annotated Data for Indic Languages. ACL.
- NLLB Team, et al. 2022. No Language Left Behind: Scaling Human-Centered Machine Translation.
- Popović, M. 2017. chrF++: Words Helping Character N-grams. WMT.
- Press Information Bureau, Government of India. Press releases and copyright policy.
- Ramasamy, L., and Žabokrtský, Z. 2012. Prague Dependency Style Treebank for Tamil. LREC.
- Ramesh, G., et al. 2022. Samanantar: The Largest Publicly Available Parallel Corpora Collection for 11 Indic Languages. *Transactions of the Association for Computational Linguistics*, 10, 145–162.
- Rei, R., et al. 2022. CometKiwi: IST-Unbabel 2022 Submission for the Quality Estimation Shared Task. WMT.
- Sarveswaran, K., Dias, G., and Butt, M. 2019. Using Meta-Morph Rules to Develop Morphological Analysers: A Case Study Concerning Tamil. FSMNLP.
- Sarveswaran, K., Dias, G., and Butt, M. 2021. ThamizhiMorph: A Morphological Parser for the Tamil Language. *Machine Translation*, 35(1), 37–70.
- Sennrich, R., Haddow, B., and Birch, A. 2016. Neural Machine Translation of Rare Words with Subword Units. ACL.
- Sutton, R. S. 2019. The Bitter Lesson.
- Su, J., et al. 2021. RoFormer: Enhanced Transformer with Rotary Position Embedding.
- University of Madras. 1924–1936. *Tamil Lexicon*. University of Madras; digital database hosted by the University of Chicago Digital South Asia Library.
- Vaswani, A., et al. 2017. Attention Is All You Need.
- Vuizur. *Wiktionary-Dictionaries*. Supplemental Tamil–English data extracted from Wiktionary.
- Wiktionary contributors. Tamil Wiktionary. Wikimedia Foundation; source snapshots obtained from the official database dumps.