A New Model for Morphological Parsing: A Case Study on Swahili Verbs

Alessio Tosolini

Department of Linguistics, University of Washington

LING 581: Morphology

Dr. Sharon Hargus

March 17, 2023

Background

Swahili is Bantu language spoken primarily in the countries of Tanzania and Kenya by over 18 million people as an L1 (Ethnologue 2022). It is a polysynthetic language with verbs following a position-class template[1], by which optional and mandatory morphemes fill slots to denote inflectional information when conjugating verbs (Loogman 1965: 114). Due to the small amounts of available online data on the language (making Swahili a low-resource language) and the vast grammatical differences between Swahili and English, existing natural language processing models struggle to accurately parse Swahili verbs (Casper 2020). I present a procedure to the task of glossing Swahili verbs and explain how it works based on the assumptions it makes about morphology, and assess the benefits and limitations of this approach to morphological parsing.

In the context of my parsing algorithm, Nondeterministic Finite State Transducers (NDFSTs) models are models which take a series of strings (sequences of characters) representing the various surface forms of morphemes, and a sequence of slots those morphemes are allowed to be in, including information about whether a slot in the position-class template is mandatory. When given a word to parse, the NDFST finds all possible combinations of the surface forms of morphemes that fill all mandatory slots.

Existing FST[2] Models' Limitations and the New Model's Goals

[1] By some analyses, Swahili verbs are considered to have up to 11 slots (Muthee 2022). In this paper, I show how the algorithm works by considering positive Swahili verbs in the indicative mood, excluding the irregular verbs *kuwa* 'to be', *kuwa na* 'to have'. Valency changing operations marked on the verb and one type of irregular verbs (monosyllabic root verbs) will be described in the Limitations section of this paper. The simplified verb template thus is as follows: SBJ-TAM-OBJ-ROOT-FV. The OBJ slot is optional, and all others are required. FV = Final Vowel (Aikhenvald 2017), REL = Relative Pronoun (Comrie 2008), ROOT = Verb Root (Comrie 2008).

[2] Finite State Transducers (FSTs) are the model which NDFSTs are built on, and the main model used in existing morphological parsing algorithms. The primary difference between FSTs and NDFSTs is that FSTs are only able to output **one** parse given an input, while NDFSTs don't have this limitation.

Current Swahili verb parsing algorithms like SwaRegex do not consider the allomorphy that may exist with certain morphemes present in the Swahili verb template (Muthee 2022). Existing language independent parsing models such as SFST-PL do not account for linguistic classes, including those covertly marked, or specific types of non-concatenative morphology, such as circumfixes (A Programming Language for Finite State Transducers 2005). Additionally, they fail to present multiple valid interpretations for an input string when there is morphological ambiguity, an issue which rarely appears in English but which is more prevalent in more morphologically rich languages (Attia 2006). The goal of the NDFST algorithm presented here is thus to create an algorithm able to (i) correctly handle agreement between **non-consecutive** morphemes and **covertly marked agreement** such as gender, and (ii) output **all** valid parses for a word in situations of morphological ambiguity.

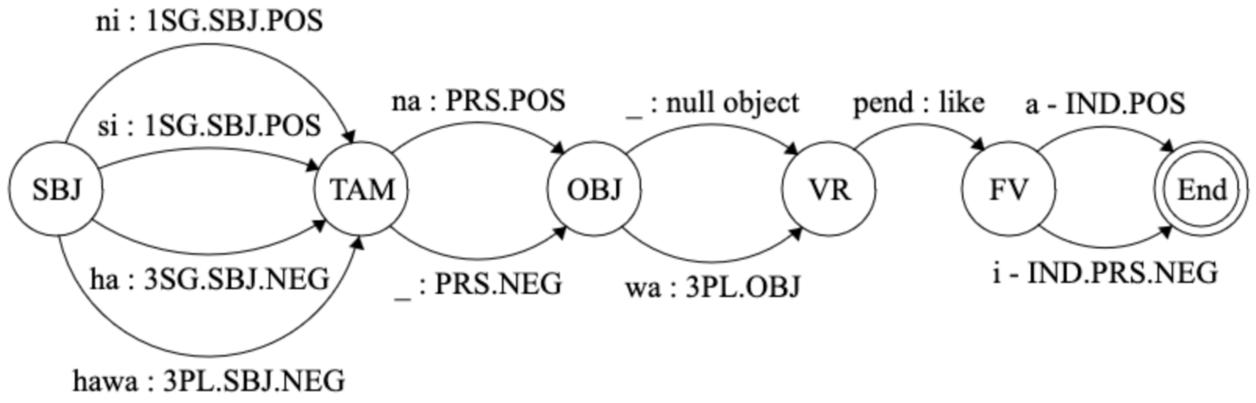


Image 1: A simplified NDFST visualization of Swahili verbs. Each circle represents a slot in the simplified verbal position-class template: SBJ-TAM-OBJ-VR-FV. For example, to parse sentence (1a) *ninapenda* 'I like', we start at the SBJ circle. Then, we move to the TAM circle through *ni-* '1SG.SBJ.POS'. The TAM prefix is *na-* 'PRS.POS', and from there we move to the OBJ circle. The key factor to note here is that as we transition from one verb slot to another, we output a gloss. We continue this until we reach the End circle. This visualization is applicable to examples (1a) - (2b).

(1a) *ni-na-pend-a*

1SG.SBJ.POS[3]-PRS.POS-like-IND

[3] POS = Positive (Aikhenvald 2017).

‘I like’

(1b) *si-∅[4]-pend-i*

1SG.SBJ.NEG-PRS.NEG-like-IND.PRS.NEG

‘I don’t like’

(1c) **si-∅-pend-a*

1SG.SBJ.NEG-PRS.NEG-like-IND[POS]

‘I (NEG) like (POS)’

We can see the issues surrounding agreement between non-adjacent morphemes with traditional FSTs by considering the simplified example above. When we consider sentences (1a) and (1b) in the context of the model depicted in Image 1, we can see that both of these sentences would correctly be parsed with their respective glosses. However, traditional models would be unable to mark (1c) as grammatically incorrect since they aren’t able to consider the fact that the *si-* ‘1SG.SBJ.NEG’ SBJ morpheme doesn’t agree with the *-a* ‘IND[POS]’ FV morpheme, as they are not next to each other. Our new algorithm’s first goal should be to resolve this issue by validating that any given parse’s morphemes agree with each other, avoiding situations as in (1c).

(2a) hawa-∅-pend-i

3PL.SBJ.NEG-PRS.NEG-like-IND.PRS.NEG

‘they don’t like’

(2b) ha-∅-wa-pend-i

[4] I do not argue for the morphological existence of a zero morpheme ∅ ‘PRS.NEG’, as it is not the agreed upon analysis of the negative present tense TAM morpheme (Ashton 1944: 70, Loogman 1965: 200). Rather, they represent the way the NDFST model parses the input word. In order to ensure that every mandatory slot is filled, the algorithm passes in an empty morpheme with a zero phonological representation but non-zero gloss for such morphemes in mandatory slots. By contrast, it does not do so when a morpheme is not passed into an optional slot.

3SG.SBJ.NEG-PRS.NEG-3PL.OBJ-like-IND.PRS.NEG

‘he doesn’t like them’

Existing models additionally struggle with morphological ambiguity since they don’t allow for multiple valid parses to be output. The Swahili word *hawapendi* has two valid parses (2a) (2b), but existing models would have to select where the morpheme boundary lies as they parse the word, allowing for only one parse per input word. Our model’s second goal will be to resolve this issue by using NDFSTs as opposed to FSTs[2].

NDFST Properties-Agreement Algorithm: Overview

```
1.   define parse(word):
2.        # Find all valid morpheme sequences with filled slots
3.        # non-empty using NDFST
4.        morpheme_sequences = parse_with_NDFST(word)
5.        # Ensure that each morpheme sequence is valid
6.        valid_morpheme_sequences = []
7.        for sequence in morpheme_sequences:

8.             if valid_agreements(sequence):
9.                  valid_morpheme_sequences.add(sequence)
10.       return valid_morpheme_sequences
```

Image 2: NDFST Properties-Agreement Algorithm overview, written in pseudocode style for clarity. Lines starting with “#” and written in orange are comments for readability. A summary of the algorithm in prose is as follows: given a word we want to parse (line 1), first parse it using NDFSTs and find all the morpheme sequences whose sum of surface forms equal the input word (line 4). Then, for every morpheme sequence we found (line 7), check to see if the morphemes in the morpheme sequence agree with each other (line 8). The output of the algorithm is thus all such morpheme sequences with internally valid agreements (line 10).

In Image 2, we can see the steps the NDFST Properties-Agreement model takes and where it differs from traditional morphological parsing algorithms, in blue. The procedure described above is a simplified way to handle concatenative morphology which meets our goals, but it is still naive in that it is unable to handle all types of morphologies. Here we see our first key assumption, that the morphological decomposition of a word is equivalent to the sum of certain predefined morphemes. We will be comparing the model above by the procedure above with the SwaRegex algorithm and the SFST-PL algorithms and identify key differences (A Programming Language for Finite State Transducers 2005, Muthee 2022).

When considering the SFST-PL algorithm, we noted its inability to handle morphological ambiguity. This is easily resolved by line 4’s use of an NDFST algorithm as opposed to a FST algorithm, where the use of a related model allows us to find more than one morpheme sequence given a word. This modeling of language also better reflects how humans understand language, and our ability to understand all valid parses given a morphologically ambiguous word. Note that “morpheme sequence” is used when describing a set of morphemes whose surface forms combine to form the word, and “gloss” is used when describing a grammatical morpheme sequence.

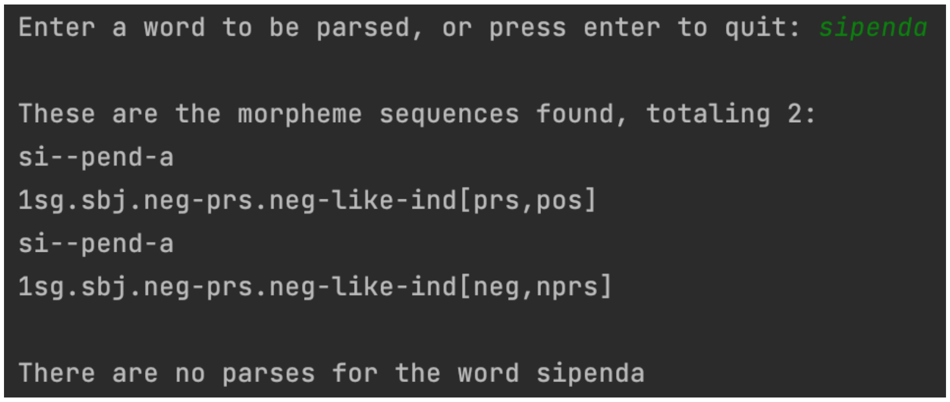
```
1.   {
2.       form: “na”,
3.       slot: “tam”,
4.       properties: [“prs”],
5.       agreements: [[“pos”,“sbj”]],
6.       gloss: “PRS.POS”
7.   }
```

Image 3: Example morpheme data for *na-* ‘PRS.POS’. `form` represents a surface realization of the morpheme. `slot` represents the slot that it is able to fill. `properties` represent the series of properties that this morpheme has, which other morphemes will check for when seeing if they agree correctly. `agreements` is the series of agreements that this morpheme has with other morphemes in the morpheme sequence. In this case, the morpheme must agree with the “pos” property of the morpheme in the SBJ slot. Finally, `gloss` is the morpheme’s gloss, in this case ‘PRS.POS’.

Additionally, we identified SwaRegex’s inability to properly handle allomorphy or agreement between two morphemes that is not overtly marked. We resolve this in lines 6 to 9 of image 2, where we run through all the morpheme sequences to ensure that they are valid. We define a morpheme sequence as being valid if all agreements defined for each morpheme (Image 3, Line 5) are present in the properties slot (Image

3, Line 4) of the morphemes it must agree with, and this is our second key assumption, which states that a parse's validity is defined based solely on whether the morphemes in a word agree with properties of other morphemes in a given slot.

To understand the steps this procedure, let us consider the ungrammatical Swahili word **sipenda* 'I (NEG) like (POS)' and see the steps our model takes to mark it as ungrammatical. Below are parses generated by following the steps in Image 2:



```
Enter a word to be parsed, or press enter to quit: sipenda

These are the morpheme sequences found, totaling 2:
si--pend-a
1sg.sbj.neg-prs.neg-like-ind[prs,pos]
si--pend-a
1sg.sbj.neg-prs.neg-like-ind[neg,nprs]

There are no parses for the word sipenda
```

Image 4: Output of the NDFST Algorithm on *sipenda*. The algorithm outputs two tentative parses for this sentence, neither of which are grammatical. The first is ungrammatical because the SBJ *si-* '1SG.SBJ.NEG' doesn't agree with the FV. The second is ungrammatical because the TAM ∅- 'PRS.POS' doesn't agree with the FV. The algorithm finally concludes there are no valid parses. NPRS = non-present (Comrie 2008).

Our procedure can consider more than one morpheme sequence for a given string to parse, something that the SFST-PL algorithm is unable to do and completing our goal relating to morphological ambiguity. At this point, the SwaRegex algorithm would have returned either one of the two morpheme sequences we found and claimed it as a valid parse, but our model also checks that each morpheme in the sequence agrees with every other morpheme in the sequence. When we check each morpheme sequence to see if its agreements are validated, we can see that it fails, as it should. Let us look again at one specific generated morpheme sequence from **sipenda*:

(1c) **si-∅-pend-a*

1SG.SBJ.NEG-PRS.NEG-like-IND[POS]

‘I (NEG) like (POS)’

We now iterate over each morpheme in the sequence to ensure that it agrees with all of the others. Let’s consider the final morpheme, -*a* ‘IND[POS]’, which must agree with the morpheme in the SBJ slot and TAM in being positive. We notice that it does not agree with the morpheme *si*- ‘1SG.SBJ.NEG’ in the SBJ slot since -*a* ‘IND[POS]’ is positive, encoded as having the property “pos”. Since the morpheme in the subject slot *ni*- ‘1SG.SBJ.POS’ has “neg” as a property and not the “pos” required by the FV , the algorithm cannot validate this morpheme sequence, and it is not considered a valid parse. From this example, we can see that it successfully satisfies our two goals.

From this worked example, we can see the benefits of approaching morphological parsing from this simplistic perspective. Firstly, by not limiting ourselves to a single valid parse given an input word, we are able to better handle situations of morphological ambiguity. Secondly, by assigning certain properties to each morpheme and validating that each morpheme’s agreements are met, we can accurately parse sentences with complex agreements or agreements between non-consecutive morphemes.

Limitations

Firstly, it must be noted that the NDFST Algorithm does not accurately reflect human morphological processing, which in reality is much more varied in its capabilities, but rather a simple way of parsing morphologically rich words to fit our goals (Oseki 2020). Because of this, it is important to consider the limitations of simplifying language to fit a model in the context of the wide range of morphologies that exist in the real world, and the tradeoffs between creating a model to describe language in high theoretical fidelity and as it is and one that is computationally accurate.

(3a)	*fungu-**liw**-a*	(3b)	*fungu-lish-a*	(3c)	*fungu-lish-**w**-a*
	fasten-**PASS**-FV		fasten-CAUS-FV		fasten-CAUS-**PASS**-FV
	“be fastened”		“cause to fasten”		“cause to be fastened”

As mentioned in the previous section, one of the assumptions this parsing procedure makes is that a morpheme agrees with other morphemes in fixed slots relative to itself, but there exist many situations where allomorphy or agreement is dependant not on a morpheme in another fixed slot, but one directly preceding or following the morpheme in question. In examples (3a) and (3c), we can see that the morpheme -*w* ‘PASS’ has C-V allomorphy, same as examples (3a) and (3b), depending on the morpheme that precedes it (Loogman 1965: 112). Our model, by assuming that allophony is dependent on morphemes in a fixed slot, is unable to handle in a simple way the type of allomorphy described in the examples above. However, it would still be able to accurately parse examples such as (3c) by considering -*lish* ‘CAUS’ and -*w* ‘PASS’ as a single morpheme -*lishw* ‘CAUS-PASS’, sacrificing faithful representations of linguistic features for a simpler model. This shows that although viewing allomorphs as agreeing with the properties of morphemes in fixed slots generally allows us to accurately parse words, it begins to break down when the allomorphy is dependent on neighboring morphemes, a common situation in some polysynthetic languages.

(4a)	**ni-ˈna-l-a*	(4b)	*ni-na-ˈku-l-a*
	1SG.SBJ.POS-PRS.POS-eat-IND		1SG.SBJ.POS-PRS.POS-AUG-eat-IND
	‘I eat’		‘I eat’

(4c)	**ni-na-ku-ˈl-ish-a*	(4d)	*ni-na-ˈl-ish-a*
	1SG.SBJ.POS-PRS.POS-AUG-eat-CAUS-IND		1SG.SBJ.POS-PRS.POS-AUG-eat-CAUS-IND
	*‘I cause (someone) to eat’		‘I cause (someone) to eat’

A second issue with our model for Swahili verbal morphology is its inability to handle suprasegmental features, including stress. In Swahili, stress always falls on the penultimate syllable, but positive TAM affixes are unable to take stress. This means that in situations where these prefixes would be stressed (4a), an augmentative prefix *ku-* ‘AUG’ must be inserted before the verb root to resolve the prosodic constraints of the language (4b) (Loogman 1965: 192). On its own, one could claim that *ku-* ‘AUG’ is simply a morpheme which must agree with monosyllabic roots like *l* ‘eat’, however the morphophonological interaction of Swahili’s prosodic rules with the morpheme *-ish* ‘CAUS’ (4d) are difficult to capture with our model due to the rule ordering of inserting morphemes (4c). One possible work-around would be to say that the morpheme *ku-* ‘AUG’ must agree with (i) a monosyllabic root, **and** (ii) a null morpheme in the slot for the causative. The addition of a null morpheme once again represents the trade off between linguistically informed models and simple parsing algorithms, as claiming that *ku-* ‘AUG’ agrees with a null morpheme is not linguistically justified, but necessary to ensure grammatical parsing of example (4d) under our model’s assumptions (Alekseeva 2022).

(5a)	**ni-na-ku-w-a*	(5b)	*mimi ni*
	1SG.SBJ.POS-PRS.POS-AUG-be-IND		1SG be.POS.PRS.IND
	‘I am’		‘I am’

Finally, let us consider the Swahili verb *kuwa* ‘to be’, whose present indicative conjugation is highly irregular due to suppletion (5b). As opposed to other verbs, it does not follow a verb template, rather opting for a highly inflectional morpheme *ni* ‘be.POS.PRS.IND’ to be used instead. We may not analyze this as being composed of phonologically null morphemes as with example (1c) due to *ni* ‘be.POS.PRS.IND’ not inflecting for person, a mandatory slot in the Swahili verb template (Ashton 1944: 205). This example of suppletive morphology requires us to modify our assumptions of how morphology can be processed, and speaks to the morphological diversity that a parsing algorithm must handle to accurately represent a language’s morphology. In the case of the ungrammatical (5a) and irregular (5b) conjugations, the best way to handle this is to consider two different templates, one for suppletive verbs and one for all other verbs.

Finally, although reduplication is not productive in Swahili verbal morphology, it is worth mentioning due to our parsing procedure’s complete inability to handle it under our assumptions. Consider the following two words from Warlpiri, which uses total reduplication as a technique for marking the plural on animate nouns (Nash 1986: 130).

(6a)	*kurdu*	(6b)	*kurdu~kurdu*
	child		child~child
	‘child’		‘children’

Although examples of singular nouns (6a) would easily be represented by our model, we can’t account for reduplication examples like (6b) besides manually encoding every example of reduplication, which would defeat the purpose of having a morphological parser in the first place. Although our model based on our assumptions is able to correctly parse a very large range of verbs, these examples of diverse morphological processes show that there is no easy way to computationally model all types and nuances of morphology under this simplified understanding of language.

Works Cited

(2002) Ethnologue, Languages of the World. United States. [Web Archive] Retrieved from the Library of Congress, https://www.loc.gov/item/lcwaN0021868/.

(2005) A Programming Language for Finite State Transducers, *Proceedings of the 5th International Workshop on Finite State Methods in Natural Language Processing (FSMNLP 2005)*, Helsinki, Finland.

Aikhenvald, A. & Dixon, RMW. (2017). *The Cambridge Handbook of Linguistic Typology.*

Alekseeva, M., Myachykov, A., & Shtyrov, Y. (2022). Inflectional zero morphology – Linguistic myth or neurocognitive reality? *Frontiers in Psychology, 13.*

Ashton EO. (1944). *Swahili grammar (including intonation)*. 2nd ed., 13th impression. Longmans, Green and Co..

Attia, Mohammed. (2006). An Ambiguity-Controlled Morphological Analyzer for Modern Standard Arabic Modelling Finite State Networks. *ACL Anthology*.

Brand, D., & Zafiropulo, P. (1983). On communicating finite-state machines. *Journal of the ACM (JACM)*, *30*(2), 323-342.

Casper S. Shikali & Mokhosi R. (2020). Enhancing African low-resource languages: Swahili data for language modelling. *Data in Brief*.

Comrie, B., Haspelmath, M., & Bickel, B. (2008). The Leipzig Glossing Rules: Conventions for interlinear morpheme-by-morpheme glosses. *Department of Linguistics of the Max Planck Institute for Evolutionary Anthropology & the Department of Linguistics of the University of Leipzig.*

Loogman A. (1965). *Swahili grammar and syntax*. Duquesne University Press.

Muthee G., Makau M., Amos O. (2022). SwaRegex: a lexical transducer for the morphological segmentation of swahili verbs. *African Journal of Science, Technology and Social Sciences*.

Nash, D. (1986). Topics in Warlpiri grammar. *Outstanding Dissertations in Linguistics*.

Oseki, Yohei and Marantz, Alec (2020) "Modeling Morphological Processing in Human Magnetoencephalography," *Proceedings of the Society for Computation in Linguistics: Vol. 3 , Article 22*.

Wiemerslage, A., Silfverberg, M., Yang, C., McCarthy, A.D., Nicolai, G., Colunga, E., & Kann, K. (2022). Morphological Processing of Low-Resource Languages: Where We Are and What's Next. *Findings*.