Finite State Transducer based Morphology analysis for Malayalam Language

Santhosh Thottingal
Swathanthra Malayalam Computing(smc.org.in)
Kerala, India
santhosh.thottingal@gmail.com

Abstract

This paper presents a finite state transducer approach to morphology analyser and generator for Malayalam language, an agglutinative, inflectional Dravidian language spoken by 38 million people, mainly by people from Kerala, India. This system, named as Mlmorph, is implemented using Stuttgart Finite State Transducer(SFST) formalism and uses Helsinki Finite-State Technology(HFST) as Toolkit. Evaluations show that it is fast and effective to address the morphological and phonological nature of Malayalam. Applications like spellchecker, named entity recognition, number spell out parser and generator are also built on top of Mlmorph.

1 Introduction

Malayalam is a language spoken in India, predominantly in the state of Kerala with about 38 million speakers. This Dravidian language has a rich morphology characterized by inflection and agglutination. Addressing this morphology is a prerequisite for any progress in language computing. Even though there were several efforts on this front, there is no functional morphology analyser for Malayalam. The Malayalam morphology analyser, named as **Mlmorph** presented in this paper tries to solve this. This paper first explains the nature of Malayalam morphology briefly. Then the methodology of morphology analyser and implementation is explained in detail. Evaluation method and results are discussed at the end. A brief description of some of the applications already built on top of this analyser is a lso provided.

2 Malayalam

Malayalam is a heavily agglutinated and inflected language(Asher, 2013). The words are formed by the morphological processes involving (a) *Inflection* where a word in a lexical category undergoes inflection by attaching suffixes to it, generating a new word in the same category (b) *Derivation* where a word belonging to a category becomes another category by attaching a suffix, (c) *Compounding* where a new word is formed by combining two or more nouns, noun and adjective, adjective and noun, verb and noun, or adverb and verb.

2.1 Morphology of nouns

Nouns get inflected due to gender, number or cases(nominative, accusative, dative, sociative, instrumental, genitive and locative). Some examples:

1. പൂച്ചകൾ → പൂച്ച<n><pl>
   cats → *cat<n><pl>*
2. ഇലയിൽ → ഇല<n><locative>
   on leaf → *leaf<n><locative>*

Adjectives can be derived from nouns, changing the root word suffix. An adjective can get agglutinated with a noun as given below.

പണപ്പെട്ടിയിൽ→ പണം<n><adj> + പെട്ടി <n><locative>

in money box → money<n><adj> + box <n><locative>

The above example also illustrate that derivation, agglutination, inflection - all these three can happen in a single word. Also, they can happen more than once in a single word. Example:

പഞ്ചസാരമണൽത്തരികളിലുമാണ് → പഞ്ചസാര<n><adj> + മണൽ<n><adj> + തരി<n><pl><locative> + ഉം<cnj> + ആണ്<aff> *it is in the sugar-white grainy sands too → sugar<n><adj> + sand<n><adj> + grain<n><pl><locative> + too<cnj> + is<aff>*

2.2 Morphology of verbs

Verbs in Malayalam get inflected based on tense, mood, voice and aspect(Varma, 2006). Verbs are inflected for present, past and future tenses. Perfect, habitual and iterative aspects are very common. Iterative aspect has tense and emphatic variations. Verbs get inflected with causative and passive voices as well. A variety of mood forms such as abilitative, imperative, compulsive, promissive, optative, purposive, permissive, precative, irrealis, monitory, conditional, satisfactive exist. All of these forms are supported by Mlmorph. Some examples are given below.

ചിരിച്ചു → ചിരിക്കുക<v><past>
laughed → laugh<v><past>

ചിരിച്ചുകൊണ്ടിരുന്ന → ചിരിക്കുക<v> <iterative-past-aspect><adv-clause-rp-past>
the one who kept laughing → laugh <v><iterative-past-aspect><adv-clause-rp-past>

Nouns can be derived from verbs and then it can undergo all nominal inflections or agglutinations.

പാടിക്കൊണ്ടിരിക്കൽ → പാടുക<v><iterative-aspect><n><deriv>
continuous singing → sing<v><iterative-aspect><n><deriv>

3 Methodology

Mlmorph is based on Finite State Transducer technology. A finite state transducer (FST) maps strings from one regular language (surface language) onto strings from another regular language (analysis language). This process is reversible too. The same transducer can be used to generate (i) analyses for a surface form (in analysis mode) and (ii) surface forms for an analysis (in generation mode). The number of generated or analysed output strings is usually one, but can be more than one while handling morphology variations specific to a language.

3.1 Implementation

The Mlmorph is written in the SFST transducer specification language which is based on extended regular expressions with variables and operators for concatenation, conjunction, disjunction, repetition, composition, negation, context-dependent replacement, and more(Schmid, 2005). A compiler translates the transducer specifications to an optimal automata. The generator automata can be reversed to create the analyser automata using the same compiler. For Mlmorph, the SFST based morphology model is compiled using Helsinki Finite-State Technology(HFST) toolkit (Lindén et al., 2011). HFST provides programming language interfaces such as python binding, and several tools to work with the compiled automata. SFST is one of the backends HFST supports.

Mlmorph has the following top level composition.

```
$analysis-filter$ || $morph$ || $phon$
    || $delete-pos$
```

This composition results in a morphology generator. Reversing the order of composition, we get analyser. First, the analysis symbols are passed through a filter `$analysis-filter$`. It accepts only the known characters and tags for Mlmorph. It then goes through the morphology rules defined by `$morph$`. The results of morphology generation is applied with the phonological rules defined by `$phon$`. Finally, all POS tags and intermediate tags used internally are removed using `$delete-pos$`. The output is the generated word.

In the transducer `$morph$`, we define what is a word model for Malayalam - `$word$`. We define it is a union of nouns(`$nouns$`), verbs(`$verbs$`), adjectives, adverbs, interjection, quantifiers etc. Kleene's plus(+) and star(*) operators has their usual meaning to denote number of occurrences.

```
$word$ = $punctuations$? ( $nouns$ \
```

```
| $verb$ | $noun_verb_compounds$ \
| $adjective$+ | $adverb$+ \
...
)? $punctuations$*
```

For the sake of brevity, we will explain noun and verb transducer here. A noun is formed by a union of singular nouns or plural nouns. A demonstrative(`$dem$`) or adjective(`$adjective$`) can precede it. The nominal inflectional forms `[#ninfl#]`, postpositions, conjuctions, polarity forms, quantifiers etc. can be suffixes. The whole word then goes through the nominal inflection rules defined by `$ninfl$` transducer.

```
$suffixes$ = $postpositions$ |
    $conjunction$ | $polarity$ |
    $quantifiers$
$noun$ = $dem$ | ( $dem$? $adjective$? (
     $singular_noun$ | $plural_noun$ )
[#ninfl#]? $suffixes$? ) || $ninfl$
```

The verb model is defined as a verb stem from the defined lexicon going through a composition of union of tense, mood, aspect and adverb forms of verbal inflection.

```
$verb$ = $vstem$ || ( $verb-tenses$ |
  $verb-moods$ | $verb-aspects$ |
  $verb-adverbs$ )
```

The phonological transducer `$phon$` applies the composition of phonological rules on the results of previous steps. The changes are mainly based on the last letter of first joining morpheme and first letter of second morpheme.

A python library is implemented on top of the automata generated[1], that abstracts analysis and generation, making Mlmorph easy to use in other applications. A web interface is also available to try out[2]. It also provides web APIs for other applications to consume.

3.2 Lexicon

The lexicon of Mlmorph contains the root words, classified and tagged into the following categories: nouns, person names, place names, postpositions, pronouns, quantifiers, abbreviations, adjectives, verbs, adverb, affirmatives, conjunctions, demonstratives, English borrowed nouns, Sanskrit rooted nouns, interjections and language names.

This lexicon is sourced from Malayalam Wiktionary, Wikipedia(based on categories there), CLDR(for language, place names) etc. The collected words are manually proofread and cleaned up. This task is tedious, but is very critical to the quality of analysis. It is also observed that the coverage ratio of Mlmorph largely depends on lexicon size.

3.3 POS tags

There is no agreement or standard on the POS tagging schema to be used for Malayalam. The general disagreement is about naming, which is very trivial to fix using a tag name mapper. The other issue is classification of features, which I found that there no elaborate schema that can cover Malayalam. So Mlmorph uses its own POS tagging schema and wherever possible the POS tags from Universal dependencies[3] (McDonald et al., 2013) are used. So far Mlmorph has defined 87 POS tags.

4 Applications

A morphology analyser and generator is the foundation for many language computing applications. As we have a functional morphology analyser now, a few applications were built on top of it, mainly to showcase some use cases.

4.1 Malayalam number spell out

In Malayalam, the spell out of numbers forms a single word. For example, a number 108 is നൂറ്റെട്ട് – a single word. This word is formed by adjective form of നൂറ്(100) with എട്ട്(8). While these two words are glued, Malayalam phonological rules are also applied, resulting compounded word നൂറ്റെട്ട്. Parsing the number നൂറ്റെട്ട് and interpreting it as 108 or converting 108 to നൂറ്റെട്ട് is an interesting problem in Malayalam computing.(Thottingal, 2017).

4.2 Spelling checker

The productive morphology of Malayalam causes practically infinite vocabulary. So a word list based approach for spellcheck won't work. A morphology analyser based spellchecker was developed and found to be effective in addressing this long pending need for the language(Thottingal, 2018). A word is correctly spelled if the word can be analysed using

[1] https://pypi.org/project/mlmorph/
[2] https://morph.smc.org.in/
[3] http://universaldependencies.org/

morphology analyser. If not, a variety of correction strategies were applied to find spelling correction candidates.

4.3 Named entity recognition

Identifying the named entities in words that are inflected or agglutinated was a challenge, but with the morphology analyser it is now possible to easily identify them by analysing each words. It is also possible to get better search results for named entities like people names, person names or places in large text for content using the same approach. (Thottingal, 2019).

5 Evaluation and Results

The evaluation of the system is done in three levels:

1. The Mlmorph is developed in a test driven approach. A manually prepared test case containing a word and its morphology analysis is first added to the collection of test cases. Then the analyser is enhanced to pass this test while not breaking any other tests that are already existing. About 450 such tests are present in the system. The same tests are used for morphology generator. So every analysis in the test case should generate the word by passing through the morphology generator. The test success rate is 100%

2. A set of 50,000 Malayalam words from the Malayalam corpus project of SMC[4] were used for the coverage analysis. The content in this set is already proofread and cleaned up to avoid words from other languages. It was found that 84% of words are analysed by Mlmorph. For the words that are not able to parse by the analyser, a frequency analysis is done, and top items in that list are considered for the tests in first level.

3. 1.4 million words collection, from all articles of Malayalam Wikipedia as of January 1, 2019 is used for testing. It can have spelling mistakes, words and acronyms from other languages like English. Also it can have many named entities that are not present in the lexicon of Mlmorph compared to the processed corpus in previous step of evaluation. The current coverage for this corpus is 45%.

[4]https://gitlab.com/smc/corpus

5.1 Performance

The Mlmorph implementation is reasonably fast as illustrated in Table 1. There is a constant time delay to load the whole automata of 12MB to memory and get ready for processing words. The time given in Table 1 includes that. The performance of the applications are very similar to this.

Number of words	Time taken for analysis
800	2 seconds
40,000	5 seconds
1400000	90 seconds

Table 1: Performance of Mlmorph. System: 64bit 4×i7-7600U CPU @ 2.80GHz, 15GB RAM

6 Future work

From the evaluation, it was observed that most of the words that the analyser failed to analyse are less frequent, but valid named entities such as person names, place names, brand names, language names etc. Adding more such words to lexicon is a never ending task, but having a better coverage based on large corpus evaluation, the effectiveness of Mlmorph can be enhanced. A few automated lexicon enhancements and possible sources for such words are being actively explored.

7 Concluding remarks

This paper presented the first functional system for Malayalam morphology. The transducer has addressed the agglutinational and inflectional properties of Malayalam. The system has the potential to be of an important component of language computing tools such as spell checker, search, named entity recognition and machine translation. The code is available under a free/open-source license(MIT license)[5]. The analyser, generator, named entity recognition applications are available at project website[6].

[5]https://gitlab.com/smc/mlmorph
[6]https://morph.smc.org.in

References

Asher, Ronald E. 2013. *Malayalam*. Routledge.

Lindén, Krister, Erik Axelson, Sam Hardwick, Tommi A Pirinen, and Miikka Silfverberg. 2011. Hfst—framework for compiling and applying morphologies. In *International Workshop on Systems and Frameworks for Computational Morphology*, pages 67–85. Springer.

McDonald, Ryan, Joakim Nivre, Yvonne Quirmbach-Brundage, Yoav Goldberg, Dipanjan Das, Kuzman Ganchev, Keith Hall, Slav Petrov, Hao Zhang, Oscar Täckström, et al. 2013. Universal dependency annotation for multilingual parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 92–97.

Schmid, Helmut. 2005. A programming language for finite state transducers. In *FSMNLP*, volume 4002, pages 308–309.

Thottingal, Santhosh. 2017. Number spellout and generation in malayalam using morphology analyser. http://thottingal.in/blog/2017/12/10/number-spellout-and-generation-in-malayalam-using-morphology-analyser/. Accessed: 2019-05-15.

Thottingal, Santhosh. 2018. Malayalam spellchecker – a morphology analyser based approach. https://thottingal.in/blog/2018/09/08/malayalam-spellchecker-a-morphology-analyser-based-approach/. Accessed: 2019-05-15.

Thottingal, Santhosh. 2019. Malayalam named entity recognition using morphology analyser. https://thottingal.in/blog/2019/03/10/malayalam-named-entity-recognition-using-morphology-analyser/. Accessed: 2019-05-15.

Varma, AR Rajaraja. 2006. *Keralapanineeyam*. DC books, 8 edition.