Using Meta-Morph Rules to develop Morphological Analysers: A case study concerning Tamil

K Sarveswaran
University of Moratuwa
Sri Lanka
sarvesk@uom.lk

Gihan Dias
University of Moratuwa
Sri Lanka
gihan@uom.lk

Miriam Butt
University of Konstanz
Germany
miriam.butt@uni-konstanz.de

Abstract

This paper describes a new and larger coverage Finite-State Morphological Analyser (FSM) and Generator for the Dravidian language Tamil. The FSM has been developed in the context of computational grammar engineering, adhering to the standards of the ParGram effort. Tamil is a morphologically rich language and the interaction between linguistic analysis and formal implementation is complex, resulting in a challenging task. In order to allow the development of the FSM to focus more on the linguistic analysis and less on the formal details, we have developed a system of meta-morph(ology) rules along with a script which translates these rules into FSM processable representations. The introduction of meta-morph rules makes it possible for computationally naive linguists to interact with the system and to expand it in future work. We found that the meta-morph rules help to express linguistic generalisations and reduce the manual effort of writing lexical classes for morphological analysis. Our Tamil FSM currently handles mainly the inflectional morphology of 3,300 verb roots and their 260 forms. Further, it also has a lexicon of approximately 100,000 nouns along with a guesser to handle out-of-vocabulary items. Although the Tamil FSM was primarily developed to be part of a computational grammar, it can also be used as a web or stand-alone application for other NLP tasks, as per general ParGram practice.

1 Introduction

A morphological analyser and generator is a crucial tool for Natural Language Processing (NLP), especially for processing morphologically rich languages like Tamil, in which morphemes are used to mark various types of information like tense, aspect, mood, person, number and, gender, *etc.* Our use of Finite-State Morphology (FSM) is based on the two-level approach to morphology in which there are two layers, namely surface and lexical (Karttunen and Beesley, 2001). The surface layer represents the actual word, and the lexical layer has a string, also called a lexical string, which shows the morphological analysis. For a language like Tamil, this analysis string is generally complex and may be long.

Designing and writing out a large number of lexical strings is not only tedious but also complicated for a morphologically rich language like Tamil. On the other hand, Tamil is morphologically well structured, in other words, the order of morphemes is generally rather templatic (Lehmann, 1993), though there are a few exceptions. For instance, simple indicative verbs consist of a root that is then followed by a tense marker and finally the person-number-gender (PNG) marker. Because of the complex, yet templatic nature of the morphological system, we decided to aid and speed up the development of the Tamil FSM via the innovation of a set of meta-morph(ology) rules. We further found that our meta-morph approach can also be extended to other structured languages by performing some initial experimentation with Sinhala, an Indo-Aryan language.

Our FSM is being developed in the context of the construction of a computational grammar for Tamil. For this, we are using the Xerox Linguistic Environment (XLE) using Lexical Functional Grammar (LFG), adhering to standards and methods set within the international ParGram effort (Butt et al., 1999).[1]

[1]https://typo.uni-konstanz.de/redmine/projects/pargram/wiki

2 Background

2.1 The Tamil language

Tamil is a Dravidian language, specifically a Southern Dravidian language that is spoken natively by more than 80 million people across the world. It has been recognised as a classical language by the government of India since it has more than 2000 years of a continuous and unbroken literary tradition (George, 2000). It is an official language of Sri Lanka and Singapore, and has regional official status in Tamil Nadu and Pondichchery, India. It has also been recognised as a minority or indigenous language in several countries including Malaysia, Mauritius and South Africa and is taught as a second language in several other countries, including Canada, Australia and United Kingdom.

Tamil is an agglutinative language where a set of morphemes are generally suffixed to a lemma. However, there are a few exceptions where morphemes are prefixed to lemmas. Words in Tamil take both inflectional and derivational suffixes, and engage in compounding. Nouns in Tamil are primarily marked for case and number. Verbs, on the other hand, display complex morphological paradigms that express a range of information relevant for syntactic and semantic analysis.

2.2 Morphology of Tamil

2.2.1 Verb morphology

Verbs in Tamil realise a range of information including tense, mood, aspect, negation, interrogation, information about emphasis, speaker perspective, sentience or rationality, and conditional and causal relations (Annamalai et al., 2014). Entities in Tamil are fundamentally classified into rational vs. irrational. Entities are termed rational if they are perceived as being able to think on their own. The rest are termed irrational. Further, it has been claimed that a weak vs. strong distinction found exists in the verbal paradigms that can be used to determine transitivity, ergativity, volitativity and affectedness (Paramasivam, 2011).

All of these properties are realised via suffixation. For instance, *he has been coming* can be translated by the Tamil verb form வந்துகொண்டிருந்திருக்கிறான் (vantukoṇṭiruntirukkirāṉ). This word consists of the following morphs: வா (vā) (lemma: 'come') + கொண்டு (koṇṭu) (continuous) + இருந்து (iruntu) (has) + இரு (iru) (be) + கிறு(kiru) (present tense)+ ஆன் (āṉ) (3rd person + singular + masculine + rational). In Tamil, PNG and rationality are expressed via a single portmanteau form (Nuhman, 1999; Sarveswaran et al., 2018). For instance, in the above example it is the morph ஆன் (āṉ) that marks all of these features.

Tamil verbs can be classified on the basis of criteria that can be either morphological, syntactic or semantic (Paramasivam, 2011; Agesthialingom, 1971). Many scholars, including Lisker (1951); Graul (1855); Arden (1962) have classified verbs based on their morphology, specifically, based on how morphemes conjugate. Graul (1855) has provided an early classification on which other scholars have built their proposals (e.g., Irākavaiyaṅkār 1958; Sithiraputhiran 2004). His classification was also adapted for the Tamil lexicon project (Rajaram, 1986). Graul's classification of Tamil verbal lemmas includes 12 categories, and is based on the future tense markers in the verbs. His basic classification is also adopted in our work. In addition, Tamil also contains a set of auxiliaries, derived verbs and compound verbs.

2.3 Noun morphology

Nouns in Tamil display the morphological processes of inflection, derivation and compounding. Nouns are inflected for number and cases (Rajendran, 2012; Nuhman, 1999). In our FSM, we have so far tackled the inflectional morphology. We have also implemented a guesser which handles the inflections of out-of-lexicon nouns, including compound nouns. Compound nouns are handled as a single unit in our current system.

Rajendran (2012) has proposed a paradigm for noun morphology with 26 classes based on the morphophonological changes, also called *canti* (Sandhi). Among these 26 classes, 9 classes are used to capture the morphophonological rules pertaining to pronouns. Pronouns take different forms (different from their nominative forms) when inflecting for a case suffix. Currently, we are handling a subset of all of the possible pronoun categories.

In our noun paradigm, we have identified 38 classes for pronouns, including personal, possessive, and interrogative pronouns. We found that though many pronouns are subject to the same morphophonological rules, these result in different analyses or lexical strings. Therefore, these have been sorted into different classes. Overall, we followed the same classification as proposed by Rajendran (2012) for other noun classes.

2.4 Finite-State Morphology

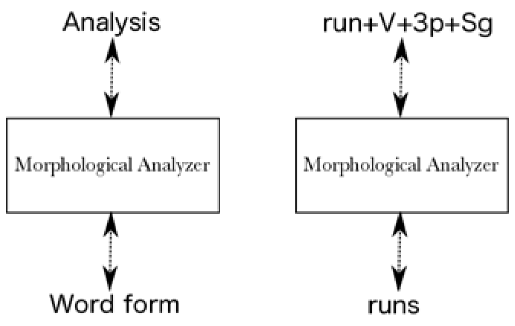


Figure 1: Word form (or Surface form) and Analysis form (or Lexical form)
Source : https://fomafst.github.io/morphtut.html

The theory of two-level morphology saw successful early applications for morphologically rich languages like Finnish, Russian and Sanskrit (Koskenniemi, 1983; Karttunen and Beesley, 2001). Subsequently, it has been taken up by researchers developing morphological analysers for other languages, including the South Asian languages Urdu, Sindhi and Nepali (Bögel et al., 2007; Prasain, 2011; Rahman, 2016) (see also Seiss 2012 for the morphologically extremely complex Australian language Murrinh-patha). In the two-level morphological analysis, a word is represented at two levels, namely the lexical level or lexical form, and the surface level or surface form, as shown in Figure 1.

Several tools have been developed to model FSM. Proprietary tools like the Xerox Finite-State Transducer (XFST) (Beesley and Karttunen, 2003) and the FSM Library from AT&T (now in OpenFST) have been widely used in the past. Open source solutions like OpenFST (Allauzen et al., 2007), HFST (Lindén et al., 2009) and Foma (Hulden, 2009) are also employed. XFST has been used widely as an aid to grammar engineering in the LFG/XLE context (Beesley and Karttunen, 2003; Butt et al., 1999; Rahman, 2016) as part of the ParGram effort. However, we found that in addition to licensing issues, XFST also has issues in rendering the scripts of South Asian languages, including Tamil, Sinhala and Devanagari. Among the available open source solutions, Foma complies with XFST standards, and has built-in support for the Unicode processing and proper rendering of South Asian scripts. We therefore decided to work with this software.

3 Related work

3.1 Meta-model / Meta-grammar development

Fokkens and Bender (2013) argue strongly that humans are better suited to the task of developing linguistic analyses than machines. We also believe that it is specifically better for the analysis of a language like Tamil, which is computationally under-resourced and which displays complex and interacting patterns of linguistic structure that need to be made transparent for down-stream NLP applications.

Bender et al. (2011) and Butt and King (2003) point out that regression testing is important for grammar engineering to be able to manage complex models when extended. This is also true for the development of a morphological analyser for a morphologically complex language like Tamil, where a continuous development is required, and where each time the system should be checked for possible errors. In order to facilitate the regression testing, Fokkens and Bender (2013) proposed a meta-grammar layer for grammar development, which places the customisation of source code under the control of grammar engineers while other users are then encouraged to do whatever changes may be necessary for their language specific needs. Otherwise, engineers need to engage with in-depth linguistic knowledge, and in turn, linguists need to engage with engineering issues.

The concept of meta-modelling has also been used in domains such as information mining. For instance, Ruiz et al. (2016) have proposed meta-association rules to compile new information from data extracted from multiple data sets, in order to provide a summarised representation. Similarly, the meta-rules proposed here also provide a summary of how words are formed.

3.2 Morphological analysers

A number of studies have been done on FSMs for South Asian languages. One of the earliest was Bögel et al. (2007) for Urdu that includes a transliteration component so that the morphological analyzer and generator can also be used for the structurally almost identical language Hindi. In addition to inflectional and derivational morphology, it also tackles complex problems such as reduplication and compounding. Prasain (2011) has developed an FSM for Nepali using the two-level morphology approach and XFST tool. Rahman (2016) has developed an analyser and generator for Sindhi as a part of his work on grammar development for Sindhi. He also used XFST, which he then integrated within his grammar.

Antony and Soman (2012) have carried out a survey on the state of affairs of computational morphology of Indian languages, and have documented 17 efforts of morphological analysers and/or generators for Tamil. 12 of them were carried out before 2007 and the relevant papers, data sets and/or software are not retrievable. The remainder have been carried out since 2010. Among those five efforts (Anand Kumar et al., 2010b,a; Menaka et al., 2010) are available for download in binary form yet without any data sets.

Menaka et al. (2010) and Anand Kumar et al. (2010b) have used rule-based approaches which only perform morphological generations. Anand Kumar et al. (2010a) have, on the other hand, applied a machine learning approach for the morphological analysis and generation of Tamil. Anand Kumar et al. (2010a) claim that the system was tested using 40,000 verbs and 30,000 nouns, and that the machine learning system was trained using 130,000 verbs and 70,000 nouns from their own corpus. However, data sets, sources or any detailed documentation are not available except for a sample corpus with 270,000 tokens. The extendability of this work to aid grammar development is also questionable, and would yet need to be researched. An email exchange with the authors has established that they do not work on this domain anymore.

Parameshwari (2011) has implemented a morphological analyser and generator for Tamil using a rule-based approach which covers verbs, nouns, adjectives, pronouns, numerals and non-standard Tamil words, with the use of the Apertium tool. The author claims that the system shows an accuracy of 84%. There are no associated data sets or rules available and the authors are also not contactable.

Lushanthan et al. (2014) have proposed a morphological analyser and generator for Tamil and have implemented it using XFST. The authors have used transliteration to handle the Tamil script given that XFST has issues in rendering, although it supports Unicode internally. The authors have considered 2,000 noun and 96 verb stems for the analysis and generation. They have tested the proposed system using their own data set consisting of 3,500 nouns and 500 verbs with a success rate of 78%. However, the data sets and XFST rules have not been made available.

Anna University, India has developed a morphological analyser in 2001 called *Atcharam* that has recently been added to the GitHub repository.[2] It is developed for TAB (TAmil Bilingual) encoded text as a stand-alone application using Java. Further, there is no detailed technical documentation or rule set. Some data in the form of a list of words are available in the repository. However, those are encoded using TAB, and an attempt to convert them to Unicode was also not successful.

There are also some morphological tools available in the GitHub code repository without corresponding academic publications. Pranavan and *et al.*[3] have provided work on a basic morphological analyser developed as a stand-alone application using Java. However, as also claimed by the developers, it is a basic analyser which handles only 20 words with 28 conjugation forms.

Yet another code repository is that by tacola-aucse.[4] This is also developed as a stand-alone application using Java, covering the analysis of verbs and nouns. However, no

[2]https://github.com/tacola-auceg/morpha_ta

[3]https://github.com/Pranavan135/Tamil_Morphological_Analyzer

[4]https://github.com/tacola-aucse/Morphological-Analyzer-For-Tamil

information about the data set or the rules developed, were found. We managed to run the tool with an older version of Java, but, irregular verbs like செத்தான் (cettāṉ) ‘he died’ do not give any analysis. In some cases, the given analysis is very confusing, especially when an out-of-vocabulary word is fed in. For instance, the analysis of சர்வேஸ்வரன் (carvēśvaraṉ), a proper noun, showed that it has the root of சர்வே (carvē) and a future tense marker வ் (v) and the past tense marker ன் (n). That is, it not only mistakes a proper noun for a verb, it also provides a completely wrong analysis with two tense markers. Furthermore, if the text is not Unicode normalised, then the tool produces unexpected results. Finally, when there are multiple analyses for a word, only one is provided. In comparison to the other tools available, however, this tool works well, but the extendability of the stand-alone Java tool is not very straightforward, unless a complete documentation can be found.

4 Development of Tamil FSM

4.1 Need for a Tamil FSM

Our research conducted on existing Tamil morphological analysers has demonstrated that none of the analysers developed in the past are complete or maintained anymore and that most of the existing applications do not support Unicode encoding. Our target task of constructing a computational grammar requires a morphological analyser with good accuracy with a specific type of interface to the grammar. None of the existing efforts fulfill these requirements.

On the other hand, we found that the open source software Foma fulfils our requirements; while rendering our scripts correctly, it also complies with XFST and can be easily integrated to the grammar we develop.

4.2 Methodology

Lexicons of verbs and nouns were compiled from various sources and classified on the basis of their inflected classes. Thereafter, a set of labels was identified and a parser to parse the meta-morph rules was developed. Next, orthographic rules were written for the identified classes. The FSM was then evaluated. In order to evaluate, a data set was also compiled, since there were no existing benchmark data sets found. All the inputs were preprocessed before being analysed. The Tamil FSM has been developed as a web-based system (parsers.projects.uom.lk) so that anyone can check or use it, where a word can be fed in, and an analysis produced as an output.

4.2.1 Pre-processing

Due to the nature of the Tamil Unicode encoding and input methods, all the inputs needed to be Unicode normalised before being fed to the web interface for analysis or generation. In Tamil, the same character can be formed by multiple code sequences if it is not controlled or handled by the keyboard input driver. For instance, the letter கொ can be entered by users using the following sequences: க + ொ or க + ெ + ா . However, it is only the first sequence is acceptable and logical. Because, in Tamil, a composite character like கொ is formed by adding a vowel to a consonant. In Unicode, vowels are denoted by vowel modifiers. Therefore, a consonant cannot be followed by two vowel modifiers (actually two vowels). However, in the case of க + ெ + ா , there are two vowel modifiers are followed by a consonant. This is impossible in Tamil. Therefore, it is important to convert all the unacceptable formations to acceptable formations; the process of converting other forms to Unicode normalised form is called Unicode normalisation. Otherwise, this would lead to issues when passing through the FST. We therefore developed a script that enables the Unicode normalisation of Tamil text.

4.2.2 Compiling lexicons

A lexicon of 3,300 lemmata of Tamil verbs was compiled from the following two verified sources:

- Annamalai et al. (2014) identified 369 of the most frequently used verbs in Modern Tamil. Their analysis is based on a corpus of 7 million tokens compiled from the web and took into account expert advice on linguistic matters. Their list has been included in the contemporary Tamil dictionary Cre-A (Ramakrishnana, 2014).

- Irākavaiyaṅkār (1958) surveyed Tamil literature up until 1958, where he identified 3,124 lemmas and categorised these into 12 classes as per the classification proposed by Graul (1855) (Sithiraputhiran, 2004). However, some of these forms are not used in the contemporary language. Nevertheless, since the analysis of those verbs is necessary for processing the historical Tamil text, the entire list has been considered for the development of our FSM.

In Tamil, complex verbs are formed on the basis of infinitival forms, verbal participles and verbal nouns (Boologarambai, 1986). Therefore, in addition to the verbal lemmata collected, auxiliary classes were also constructed manually, using the infinitival and verbal participle forms of the lemmata.

Tamil nouns were collected from various databases online, glossaries and corpora. An initial level of cleaning was additionally conducted in order to ensure that the list has only lemmata.

4.2.3 Verb Paradigm

Instead of handling words individually, a paradigmatic approach is used to reduce the volume of the problem. Anand Kumar et al. (2010b) have proposed a paradigm with 32 classes in their data-driven morphological analyser, while Menaka et al. (2010) identified a verb paradigm with 34 classes in their Tamil morphological generator study. However, we have here chose to use the widely accepted 12 verb paradigm proposed by Graul (1855). In addition to these 12 categories, each of the 7 irregular verbs defined in (Annamalai et al., 2014) is considered as a separate category. Further, 15 auxiliaries, identified from the literature, were also implemented as 15 separate classes. Altogether, a taxonomy of 34 classes has been used to develop the FSM for verbal forms.

4.2.4 Conjugation forms

Annamalai et al. (2014) have identified 254 forms for each Tamil verb after a rigorous analysis of their corpus of contemporary texts. Some verbs may not take all of the 254 forms. Further, Rajaram (1986) has also identified 21 forms for each verb from a pedagogical perspective. On the other hand, Anand Kumar et al. (2010a) claim that a Tamil verb lemma can take up to 8,000 forms though not all are listed or found in the literature. In our FSM 260 inflectional forms are considered. These forms are the set common to Annamalai et al. (2014) and Rajaram (1986). For each lemma, these 260 forms are generated and analysed. However, more forms can easily be added to the system without the need of any additional programming.

4.2.5 Morpheme labels

There are different sets of labels used to mark the morphemes in the morphological analysers of Anand Kumar et al. (2010a); Menaka et al. (2010). Kirov et al. (2016) attempt to unify the morphological labels under the brand of Unimorph to facilitate the cross-lingual morphological transfer.

However, in our Tamil FSM, we have developed a set of our own morpheme labels. Because PNG and rationality are marked by a single morph in Tamil, it is more efficient from a grammar engineering perspective to handle them together, thus reducing the number of lexical rules in the grammar (Butt et al., 1999). While we have decided to develop and use our own labelling, we plan to implement an interface that will facilitate exporting information in the Unimorph format (Kirov et al., 2016).

5 Meta-morph rules

From the review of Tamil morphological analysers, it is evident that most of the efforts in defining morphological structure or morphotactics are deeply coupled with the programming logic. In some other efforts, people have spent a considerable amount of time writing rules.

Snippet 1 Snippet of meta-morph rules

```
1.classes=C1,C2,C3,C4,C5,C17,C18,C19
2.commonLabels=+fin+sim+ind
3.v-ind=root+tense+png
4.v-euph=root+past+euph+pngeuph
```

On the other hand, arguments have been presented that a meta-representation of the formal implenentational details will allow and encourage computationally naive linguists to contribute to the development of linguistic resources for language processing.

The meta-morph rules outlined in this paper successfully hide the programming details of the morphological analysis and help to focus only on the analysis of the language. Further, this also automates the generation of lexical entries, which when done manually is not only a tedious and time-consuming task, but also one where people can easily make mistakes. This is particularly true for a language like Tamil in which each verb may undergo several hundred inflections. Therefore, even if a paradigm approach is used, it is challenging to write rules, maintain them and perform a regression testing without the aid of a meta-grammar.

To achieve this we have developed meta-morph rules. Consider Snippet-1 of the meta-morph rules example. Line number 3 shows how finite, simple and indicative verbs are formed for the classes listed in line number 1. The order of conjugation also matters, where it shows that with a verb root, first, it is a tense marker that is coined, and finally a PNG marker follows. These rules can also be applied in other studies to see how words are conjugated in Tamil. Further, the rules can be defined at different levels. For instance, line number 3 shows that all the finite, simple and indicative verbs are formed by conjugating a tense and a PNG, where tense stands for the realisation of one of the three tenses available. However, line number 4 shows that verbs which consist of euphonic markers (material used to fulfill phonological phrasing requirements) are constructed only with past tense verbs, and with a specific PNG marker. Snippet-1 thus exemplifies the type of meta-morph rules that we have devised.[5]

The corresponding values for the labels in the meta-morph rules are stored in JSON[6] files. Data are stored in JSON as key-value pairs which are also human readable. The above rules and JSON entries can be written in a plain text file. For instance, Snippet-2 shows how tense labels are defined and stored in a JSON file. As shown here, there can be different past tense markers for different classes of verbs. For general cases, the tense marking can be done as shown in line 3. However, if required, a particular tense marker can also be used, as shown in line number 4 of the above Snippet-1. In addition to labels, values corresponding to each morph can also be stored in the JSON file, as shown in Snippet-2. For instance, in the above example for “past1”, the label is *past* and the morph த் (which marks the verb as past) is also included as a part of the label. This information becomes part of the lexical analysis. Further, as shown in the Snippet-2, the proposed data structure provides the flexibility for defining different tense markers and labels for different classes, and these data can be referred at different levels when writing the meta-morph rules. For instance, it can be either referred as “past” or “past1” rule-base.

Snippet 2 Snippet of data in a JSON file

```
    ”tense”: {
    ”past”: {
            ”past1”: {
                    ”label”:”+past=த்”,
                    ”marker”:”த”,
                    ”classes”:[”C1”,”C15”]
                    },
            ”past2”: {
                    ”label”:”+past=ண்ட்”,
                    ”marker”:”ண்ட”,
                    ”classes”:[”C2”]
            }
    },
    ”pres”:{
            ”pres1”: {
                    ”label”:”+pres=கிற்”,
                    ”marker”:”கிற”,
                    ”classes”:[”C2”,”C3”]
            }
    },
    ”fut”: {
            ”fut1”: {
                    ”label”: ”+fut=வ்”,
                    ”marker”: ”வ”,
                    ”classes”: [”C3”,”C4”]
            }
    }
    }
```

In case a mistake in the labelling, or in the

[5]A reviewer asks about the formal power of the meta-morph rules. Essentially they replace the continuation classes found in LEXC.

[6]https://www.json.org/

value of a marker, is found, it can be easily corrected in the JSON text file without needing to engage with FSM programming.

Once the meta-morph rules are finalised, they can be parsed to produce the actual lexical strings that are then fed to Foma to compile an FST. A parser is developed using Python to parse these morph-rules to generate lexical rules for Foma. A sample of compiled meta-morph rules will look as shown in Snippet-3. We use the pipe "|" symbol to mark morpheme boundaries, as inspired by Beesley and Karttunen (2003), who use "TB" to mark the token boundary. "|" is used in Universal Dependencies to separate features.[7] The % is allowed to escape special characters in the lexical string.

Apart from the generation of these intermediate entries, orthographical rules were written for each class in the paradigm as necessary. If a new class needs to be introduced, then a new set of entries needs to be added to the orthographical file. Otherwise, there is no need to touch the lexical strings or the orthographical files.

Snippet 3 Snippet of lexical string or analysis string

```
%|+fin %|+sim %|+ind %|+strong
%|+past %= த் %|+3sgn %=அது
```

We initially developed a Tamil FSM by entering all of the necessary lexical strings manually, yet found this to be tedious task that took time and energy away from understanding the more generalised overall structure of the language morphology. In evaluating our progress, we found that correcting errors was complicated and time-consuming, since we always had to engage with the details of the Foma specifications. The frustration with these time-consuming tasks led us to experiment with meta-morph rules. We found writing rules in the meta-morph and defining feature-value pairs using JSON to be easy and quick. It also helped us to accelerate the process of developing our FSM for Tamil, where for instance the identification of mistakes could be corrected easily. Adding a lexical string or new conjugational form also became very straightforward. We need to just list the classes which will take those new forms and then define a generalised rule for the formation of that word as shown in Snippet-1.

6 Evaluation and Discussion

There are no benchmark data sets available to evaluate a morphological analyser in Tamil. Therefore, the 500K corpus from AUKBC[8] was used for evaluation. This corpus is compiled from a popular Tamil historical novel written by an Indian author. From the 19,250 unique verbs found in the corpus, only the finite, infinitives, relative participles, verbal participles and conditional verbs were extracted. However, the finite verbs compiled also comprised of compound and derivational verbs, which cannot be separated from the finite list as there are no tags to identify them. In addition 26,000 tokens which were marked as nouns also extracted from the corpus to evaluate the noun morphology. However, the list had a significant number of compound nouns, nominal complex (for instance, noun + conjunction), personal names, and borrowed words from Sanskrit. Table 1 shows the outcome of the evaluations.

Derivational verbs, spelling mistakes, nominal complex words, and personal names, as well as errors in the tagging, were the primary causes for the failure of the Tamil FSM. The guesser provided an analysis for the compound verbs and nouns.

7 Conclusion

We conclude that meta-morph rules are useful for the acceleration of the development of FSMs. At the same time, they allow the statement of linguistic generalisations in a form that is easily human readable and provides an interface for computationally naive linguists to interact with the FSM and to potentially help extend and improve it.

The Tamil FSM outlined in this paper has also been developed as a web-based system: `parsers.projects.uom.lk`. The evaluation in Table 1 shows that the Tamil FSM already provides a high accuracy for the analysis of verbs and a reportable accuracy for the

[7]https://universaldependencies.org/format.html

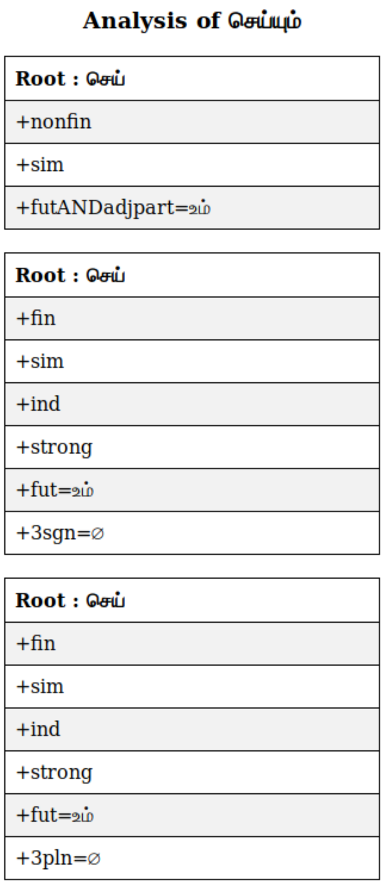
[8]http://www.au-kbc.org/nlp/corpusrelease.html

Word type	Found in the Corpus	Analysed	Percentage
Verbs - Finite	10,269	10,142	98.7
Verbs - Infinitive	1,615	1,540	95.4
Verbs - Relative participle	2,677	2,525	94.3
Verbs - Verbal participle	3,339	3,110	93.1
Nouns	26,000	19,990	76.5

Table 1: Evaluation results

analysis of nouns. Further, the system also shows if there are multiple analyses for a word; for instance, an analysis for a word செய்யும் (*ceyyum*) is shown in the Figure 2.

In future work, we intend to explore whether the meta-morph rule interface can be further generalised and used for other languages, at least for other South Asian Languages. We already have performed an initial work on this for the Indo-Aryan language Sinhala and the results are encouraging. We will also extend the existing Tamil FSM by fully incorporating derivational morphology.


Analysis of செய்யும்

Root : செய்
+nonfin
+sim
+futANDadjpart=உம்

Root : செய்
+fin
+sim
+ind
+strong
+fut=உம்
+3sgn=∅

Root : செய்
+fin
+sim
+ind
+strong
+fut=உம்
+3pln=∅

Figure 2: Screenshot of the analysis from the web interface

Acknowledgement

This research was supported by the Accelerating Higher Education Expansion and Development (AHEAD) Operation of the Ministry of Higher Education, Sri Lanka funded by the World Bank and the DAAD (German Academic Exchange Office). We also thank Lauri Karttunen for his time and provision of several helpful pointers.

References

S Agesthialingom. 1971. A Note on Tamil Verbs. *Anthropological Linguistics*, pages 121–125.

Cyril Allauzen, Michael Riley, Johan Schalkwyk, Wojciech Skut, and Mehryar Mohri. 2007. OpenFst: A general and efficient weighted finite-state transducer library. In *International Conference on Implementation and Application of Automata*, pages 11–23. Springer.

M Anand Kumar, V Dhanalakshmi, KP Soman, and S Rajendran. 2010a. A sequence labeling approach to morphological analyzer for Tamil language. *International Journal on Computer Science and Engineering (IJCSE)*, 2(06):1944–195.

M Anand Kumar, VV Dhanalakshmi, and S Rajendran. 2010b. A novel data driven algorithm for Tamil morphological generator. *International Journal of Computer Applications*, 6:52–56.

E Annamalai, A Dhamotharan, and A Ramakrishnan. 2014. *Akarātiyin̲ putiya patippil tar̲kālat tamil̲ ilakkaṇa viḷakkam*, pages xxxi–xlvii. Crea-A Publishers, India.

PJ Antony and KP Soman. 2012. Computational morphology and natural language parsing for Indian languages: a literature survey. *International Journal of Scientific and Engineering Research*, 3.

Albert Henry Arden. 1962. *A progressive grammar of common Tamil*. Christian Literature Society.

Kenneth R Beesley and Lauri Karttunen. 2003. Finite-state morphology: Xerox tools and techniques. *CSLI Publications, Stanford*.

Emily M Bender, Dan Flickinger, and Stephan Oepen. 2011. Grammar engineering and linguistic hypothesis testing: Computational support for complexity in syntactic analysis. *Language from a cognitive perspective: grammar, usage and processing*, pages 5–29.

Tina Bögel, Miriam Butt, Annette Hautli, and Sebastian Sulger. 2007. Developing a Finite-State Morphological Analyzer for Urdu and Hindi. In *Finite-State Methods and Natural Language Processing*, pages 86–96. Potsdam University Press. Revised Papers of the Sixth International Workshop on Finite-State Methods and Natural Language Processing.

A Boologarambai. 1986. *A Study of Auxiliaries in the Old and the Middle Tamil*. Ph.D. thesis, Centre of Advanced study in linguistics, Annamalai University, India.

Miriam Butt and Tracy H. King. 2003. Grammar writing, testing and evaluation. In Ali Farghaly, editor, *A Handbook for Language Engineers*, pages 129–179. CSLI Publications, Stanford.

Miriam Butt, Tracy Holloway King, Maria-Eugenia Nino, and Frederique Segond. 1999. *A Grammar Writer's Cookbook*. CSLI Publications, Stanford.

Antske Fokkens and Emily M Bender. 2013. Time travel in grammar engineering: Using a metagrammar to broaden the search space. In *Proceedings of the ESSLLI Workshop on High-Level Methodologies in Grammar Engineering*, pages 105–116.

L. Hart George. 2000. Statement on the Status of Tamil as a Classical Language. Available: https://southasia.berkeley.edu/tamil-classes. Accessed on: 2017-11-12.

Karl Graul. 1855. *Outline of Tamil grammar*. Leipzig University.

Mans Hulden. 2009. Foma: a finite-state compiler and library. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics: Demonstrations Session*, pages 29–32. Association for Computational Linguistics.

M Irākavaiyaṅkār. 1958. *'Viṉaittiripu viḷakkam' (conjugation of Tamil verbs) (in Tamil)*. Eighty year anniversary publication.

Lauri Karttunen and Kenneth R Beesley. 2001. A short history of two-level morphology. *ESSLLI-2001 Special Event titled" Twenty Years of Finite-State Morphology*. Available: http://www.helsinki.fi/esslli/.

Christo Kirov, John Sylak-Glassman, Roger Que, and David Yarowsky. 2016. Very-large scale parsing and normalization of wiktionary morphological paradigms. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Paris, France. European Language Resources Association (ELRA).

Kimmo Koskenniemi. 1983. *Two-level morphology*. Ph.D. thesis, University of Helsinki.

T. Lehmann. 1993. *A grammar of modern Tamil*. Pondicherry Institute of Linguistics and Culture, India.

Krister Lindén, Miikka Silfverberg, and Tommi Pirinen. 2009. HFST tools for morphology–an efficient open-source package for construction of morphological analyzers. In *International Workshop on Systems and Frameworks for Computational Morphology*, pages 28–47. Springer.

Leigh Lisker. 1951. Tamil verb classification. *Journal of the American Oriental Society*, 71(2):111–114.

Sivaneasharajah Lushanthan, AR Weerasinghe, and DL Herath. 2014. Morphological analyzer and generator for Tamil language. In *Advances in ICT for Emerging Regions (ICTer), 2014 International Conference on*, pages 190–196. IEEE.

S Menaka, Vijay Sundar Ram, and Sobha Lalitha Devi. 2010. Morphological generator for Tamil. *Proceedings of the Knowledge Sharing event on Morphological Analysers and Generators (March 22-23, 2010)*, pages 82–96.

M Nuhman. 1999. *Basic Tamil Grammar (In Tamil)*. Readers' Association, Sri Lanka.

K Paramasivam. 2011. *Contemporary Tamil Grammar*. Adaiyaalam.

K Parameshwari. 2011. An implementation of APERTIUM morphological analyzer and generator for Tamil. *Parsing in Indian Languages*, page 41.

Balaram Prasain. 2011. *Computational Analysis of Nepali Morphology: A Model for Natural Language Processing*. Ph.D. thesis, Faculty of Humanities and Social Sciences of Tribhuvan University, Nepal.

Mutee U Rahman. 2016. *Developing a Sindhi Computational Resource Grammar in Lexical Functional Grammar framework*. Ph.D. thesis, Faculty of Engineering Science and Technology, Isra University, Hyderabad.

S Rajaram. 1986. English-Tamil Pedagogical Dictionary. *Thanjavur: Tamil University*.

S Rajendran. 2012. Preliminaries To The Preparation Of A Spell And Grammar Checker For Tamil. *Upoaded in academia. edu and Reseach Gate.*

S Ramakrishnana, editor. 2014. *Cre-A: Dictionary of Contemporary Tamil.* Cre-A publishers, Chennai, India.

M Dolores Ruiz, Juan Gómez-Romero, Miguel Molina-Solana, Jesús R Campaña, and Maria J Martín-Bautista. 2016. Meta-association rules for mining interesting associations in multiple datasets. *Applied Soft Computing*, 49:212–223.

K Sarveswaran, Gihan Dias, and Miriam Butt. 2018. ThamizhiFST: A Morphological Analyser and Generator for Tamil Verbs. In *2018 3rd International Conference on Information Technology Research (ICITR)*, pages 1–6. IEEE.

Melanie Seiss. 2012. A rule-based morphological analyzer for murrinh-patha. In *Proceedings of the 8th International Conference on Language Resources and Evaluation (LREC 2012)*, pages 751–758. Istanbul,Turkey: EuropeanLanguage Resources Association (ELRA).

H Sithiraputhiran. 2004. *Viṉaittiripu viḷakkamum moḻiyiyal kōṭpāṭum.* International Institute of Tamil Studies.