Machine Translation
https://doi.org/10.1007/s10590-021-09261-5

*Thamizhi*Morph: A morphological parser for the Tamil language

Kengatharaiyer Sarveswaran[1] · **Gihan Dias**[1] · **Miriam Butt**[2]

Received: 26 February 2020 / Accepted: 3 March 2021

Abstract
This paper presents an open source and extendable Morphological Analyser cum Generator (MAG) for Tamil named *Thamizhi*Morph. Tamil is a low-resource language in terms of NLP processing tools and applications. In addition, most of the available tools are neither open nor extendable. A morphological analyser is a key resource for the storage and retrieval of morphophonological and morphosyntactic information, especially for morphologically rich languages, and is also useful for developing applications within Machine Translation. This paper describes how *Thamizhi*Morph is designed using a Finite-State Transducer (FST) and implemented using Foma. We discuss our design decisions based on the peculiarities of Tamil and its nominal and verbal paradigms. We specify a high-level meta-language to efficiently characterise the language's inflectional morphology. We evaluate *Thamizhi*-Morph using text from a Tamil textbook and the Tamil Universal Dependency treebank version 2.5. The evaluation and error analysis attest a very high performance level, with the identified errors being mostly due to out-of-vocabulary items, which are easily fixable. In order to foster further development, we have made our scripts, the FST models, lexicons, Meta-Morphological rules, lists of generated verbs and nouns, and test data sets freely available for others to use and extend upon.

Keywords Morphological analyser · Morphological generator · Finite-State transducer · Tamil language · Low-resource language · Morphologically rich language

✉ Kengatharaiyer Sarveswaran
sarvesk@uom.lk

Gihan Dias
gihan@uom.lk

Miriam Butt
miriam.butt@uni-konstanz.de

[1] University of Moratuwa, Moratuwa, Sri Lanka

[2] University of Konstanz, Konstanz, Germany

Published online: 23 April 2021

1 Introduction

The Web contains a large and rapidly-growing textual volume of Tamil, a Southern Dravidian language of South Asia.[1] Several organisations and individuals are working on Tamil language computing. However, in comparison to some major European languages as well as Chinese, Japanese, and Korean (CJK), not many natural language processing (NLP) tools such as Part of Speech (POS) taggers, morphological analysers or syntactic parsers are available for Tamil. In addition, many existing tools are either not open, or not extendable due to technical or license restrictions. There are also commercial products such as Google Translate and Google Optical Character Recognition. However, more work needs to be done to improve their quality before those can be put to general use. Beyond that, as claimed by Bhattacharyya et al. (2019), South Asian languages in general lack sufficient language resources in terms of gold standards or benchmark data needed for supervised machine learning.

A Morphological Analyser and Generator (MAG) is a crucial fundamental tool for NLP. This is especially so for the processing of Morphologically Rich Languages (MRL) such as Tamil, in which morphemes are used to mark various types of information, including tense, aspect, mood, person, number, and gender. On the other hand, while there have been several efforts to develop morphological analysers for Tamil (see Sect. 4.3), no functioning Tamil MAGs of sufficient quality have been available for public use and extension. In this paper, we describe how we have developed an openly available morphological analyser cum generator called *Thamizhi*-Morph (pronounced t̪amiɻi morph or Tami*li* morph)[2] and list the resources we have made available for others to use and extend.

A MAG is a useful resource for language application development and language learning. It can be used to effectively do word-level translation, especially for MRLs. Named entity translation is also a task where a MAG can be useful. For instance, in *Moses*, a statistical machine translation system, morphological analysers are used in its factored translation models to improve the results over non-factored models (Koehn et al. 2007). Further, Passban et al. (2018) found that integrating morphology into a neural machine translation pipeline is useful to help overcome out-of-vocabulary problems, especially in MRLs. Machine translation is a pressing problem in Sri Lanka, where two major languages—Sinhala and Tamil—are in everyday use, including their use in official government communications.

[1] meta.wikimedia.org/wiki/Growing_Local_Language_Content_on_Wikipedia_(Project_Tiger_2.0)/Writing_Contest.

[2] *Thamizhi* is an ancient writing system that was used to write the Tamil language.

We have used *Thamizhi*Morph to aid our grammar development within the ParGram project[3] (Butt and King 2002) to allow for the possibility to work with stems in the syntactic parser and generator, rather than full-form lexical entries.

A MAG is also useful for language learning purposes so that a learner can analyse and search for morphs of an inflected word. Our morphological analyser is designed to give the morphs in addition to the morphemes. This is important for languages with complex morphology as well as irregular words, where it is difficult for a learner to identify root words (Seiss 2012).

2 Background

2.1 The Tamil language

Tamil 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 continuous and unbroken literary tradition (Hart 2000). It is one of the official languages of Sri Lanka and Singapore, and has regional official status in Tamil Nadu and Pondicherry, India. It has also been recognised as a minority or indigenous language in several countries including Malaysia, Mauritius, and South Africa, and is taught there as a second language. The spoken forms of Tamil vary depending on the region, mainly due to language contact, and/or politics (Schiffman 2008). Within Sri Lanka, there are several varieties of Tamil that are spoken, and at times, one speaker may not understand the other. Further, there is no common agreement among the different governments on the choice of terminologies. Consequently, terminological variation also exists.

The Tamil language has its own script, which is referred to as Tamil script now, and follows the Abugida or Alphasyllabary writing system where a pure consonant and a vowel are written together as one unit or syllable. For instance, a consonant த் (*t*) together with a vowel உ (*u*) will form a single unit, a composite character, து (*tu*). The alphabet has 12 vowel, 18 pure consonant, and 216 composite characters which are formed when pure consonants and vowels combine together in a single unit. Tamil also has a special character ஃ (*ah*) which is categorised neither as a vowel nor as a consonant. In addition to these a total of 247 Tamil letters, some letters from the Grantha alphabet, which is used to write Sanskrit in South India, are also widely used with the Tamil alphabet.

[3] The Parallel Grammar (ParGram) Project (Butt et al. 1999; Butt and King 2002) aims to develop and implement large and wide coverage grammars for languages of different families. These parallel grammars are written collaboratively within the linguistic framework of Lexical Funcational Grammar (LFG) and with an agreed set of grammatical features by the project group members. The Xerox Linguistic Environment (XLE) (Crouch et al. 2017), which is a parsing and generation implementation of LFG developed at PARC, is used as a grammar development platform. In addition to putting effort into feature standardisation, the project also promotes similar analyses for similar phenomena across languages (Butt and King 2002), a property that is useful for cross-lingual language applications such as machine translation and information retrieval.

Apart from the information about the alphabet, understanding how these letters are encoded using Unicode[4] is important in order to develop language processing tools, and to write morphological and orthographical rules. Each syllable in Tamil has a single code point. For instance, all the vowels and pure consonants in the Tamil alphabet are each encoded with a single code point. However, all the composite characters are encoded with two Unicode points; one to represent the consonant letter, and the other to represent the vowel modifier which is applied to form the composite. For instance, து (*tu*) will have two Unicode points,[5] one for த (*t*) and the other for the ◌ு (*u*)- vowel modifier. Similarly, தொ (tou) also has two Unicode points corresponding to த (*t*) and ◌ொ (*ou*). Typing in these vowel modifiers and handling them as part of computer programming is not a very straightforward task.

2.2 Finite-State morphology

In the conception of two-level morphology, a word is represented at two levels, namely the lexical level, or lexical form, and the surface level, or surface form. This concept can be modelled computationally using Finite-State Transducers (FST) (Beesley and Karttunen 2003) and is referred to as Finite-State Morphology (FSM). A FSM approach has been widely used to develop successful early applications for morphologically rich languages such as Finnish and Russian (Koskenniemi 1983; Karttunen and Beesley 2001). Subsequently, it has been taken up by researchers developing morphological analysers for other languages, including South Asian languages such as Urdu (Bögel et al. 2007), Sindhi (Rahman 2016) and Nepali (Prasain 2011), and also for the morphologically extremely complex Australian language Murrinh-patha (Seiss 2012).

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.

[4] https://home.unicode.org/.

[5] https://unicode.org/charts/PDF/U0B80.pdf.

Abbreviations used in examples: VPART=Verbal Participle; INF=Infinitive; 3SN=3rd Person Singular Neuter; 1S = 1st Person, Singular; 3SMR=3rd Person, Singular, Masculine and Rational; PASS=Passive; SAN=Sandhi; RP= Relative Participle; IMP=Imperative; CAUS=Causative; 3SE= 3rd person, singular and epicene, 3SEH=3rd person, singular, epicene and honorific; PL=Plural; OBL=Oblique; EUPH=Euphonic; NOM=Nominative; DAT=Dative; ACC=Accusative; INST=Instrumental.

2.3 Foma

Foma is a C library and a compiler that is used to develop FSTs for various purposes, including the development of language applications such as a MAG. In this section, we briefly discuss how Foma can be used to model inflectional morphology.

Developing an FST-based morphological analyser generally requires two components: (1) list of morphs (morphotactics and lexicon); (2) alteration rules (morphophonological rules and orthographical rules) (Beesley and Karttunen 2003; Hulden 2009). Foma also has these two components. A lexicon component shows the ordering restrictions of the root and its morphs, and maps them to an intermediate or final form. For instance, (1) (a) shows how a plural morpheme can be mapped to intermediate and final forms respectively, where the morph *s* is used to mark the morphemes *+Noun* and *+Pl*. However, as shown in (1)(b), it does not yield the final results for all constructions. This is where the second component, alteration rules, comes in. In this component, we define the morphophonological or orthographic changes which take place. In other words, as a first step, we consider the generic scenario, where root forms take *s* to mark plural, and then write alternation rules for the exceptional cases, as in (2). In Foma we can define such a rule, as in (2), where we define, for example, an eInsertion rule for plural constructions and add a letter *e* in front of plural morph *s* whenever a noun ends with the letters *ch*. In this way we can generate the final plural form for *watch*, as *watches*. We can define any number of such rules and can then concatenate them and apply them to the output of the lexicon component to get the final forms.

(1) (a). cat +Noun +Pl:cat∧s
(b). watch +Noun +Pl: watch∧s

(2) define eInsertion [..] -> e || c h _ ∧ s ;

The words of a language can be divided into different classes based on their morphophonological behavior. When we develop a morphological analyser we therefore need to define different lexicon components for each of the identifiable word classes. Each of these lexicon components is associated with different sets of alteration rules. This feature is useful for a MRL such as Tamil, where the alteration rules are complex. In Foma we can define these classes as different *lexc* files that can then be concatenated as necessary integrating alteration rules.

3 Tamil morphology

In this section, we provide some basic information about Tamil morphology and then discuss the paradigms we have used in order to develop the analyser.

The Tamil grammar tradition classifies words in Tamil as either divisible or indivisible. A divisible word can have up to six parts, namely: root, suffix, *idainilai* (medial particle), *chariyai*, *Sandhi*, and *vikaram* (alteration) (Nuhman 1999; Senavaraiyar 1938). The medial particles can be tense markers or negation markers in divisible verbs as shown in (3). *chariyai* is a phonological modifier which can be further divided into a euphonic marker and an oblique marker based on the function it expresses (Lehmann 1993). The term *Sandhi* refers to a cluster of morphophonological phenomena, the alteration is also a morphophonological process of assimilation which has orthographic consequences.

(3)

(a)	செய்கிறான்	'seikiran'		(b)	செய்யாது	'seiyaatu'	
	செய்	கிற்	ஆன்		செய்	ஆ	அது
	sei	*kir*	*aan*		*sei*	*aa*	*atu*
	'(He) is doing.'				'(It) won't do.'		

The six parts of a Tamil word can be illustrated using the following example in (4), which is taken from Sarveswaran and Butt (2019, p. 274). Here, an internal *Sandhi t* occurs between the root and the tense particle *t*, and then it becomes *n* because of *vikaram* (alternation).

(4)

வந்தனன் (*vantanan*)				
வா	த்(ந்)	த்	அன்	அன்
vaa	*t(n)*	*t*	*an*	*an*
root (வா→வ)	*Sandhi* (த்→ந்)	*medial*	*chariyai*	suffix
'(He) came.'				

Tamil is an agglutinative language where a set of morphs is generally suffixed to the root or base, as shown in (4). However, there are some exceptions, where morphs are prefixed, especially for negation, and demonstratives. For instance, அநியாயம் (*aniyayam*) 'injustice' involves the prefixation of அ (*a*) to express negation. The words with prefixation can also take suffixes, for instance, அநியாயத்தை (*aniyayattai*) 'injustice.ACC' where the ஐ (*ai*) expresses accusative case.

All words in Tamil can be segmented into two parts: (1) the base or root; (2) grammatical formatives. The grammatical formatives can be either inflectional or derivational affixes that are added via six morphophonological operations, namely, affixation, incorporation, compounding, cliticisation, doubling, and stem mutation (Lehmann 1998). Example in (4), for instance, shows the process of suffixation, and in association with it, stem mutation where வா(*vaa*)→வ (*va*).

Word formation through agglutination exhibits fusional tendencies or morphophonemic alterations between the root word and grammatical formatives. Based on these alterations, several different conjugational and declinational patterns or paradigms for verbs and nouns, respectively, can be identified. In Sects. 3.2 and 3.3 we describe verbal and nominal paradigms that have been considered in order to be able to develop the morphological analyser.

3.1 Part of speech

In grammar books written by native grammarians (Thesikar 1957; Senavaraiyar 1938) Tamil words have been primarily divided into four types, namely: nouns, verbs- intensifiers/attributives, and particles. However, more recently there have been more granular Part of Speech (POS) analyses proposed by Sarveswaran and Mahesan (2014); Baskaran et al. (2008); Lehmann (1993). We follow Lehmann (1993) and Sarveswaran and Mahesan (2014) closely. These are relatively less granular when compared to others, but we have found that these allow for the most accurate analysis in our implementation.

3.2 Verbal morphology

The structure of a simple verb in Tamil is <root> + <medial-particle> + (euphonic-particle) + <terminal-suffix>, where the euphonic particle அன் (*an*) is optional—it seems to be used to add a dimension of politeness to the verb in current usage. The medial particle is used to realise the tense (past, present and future) or negation of the verb (Pope 1979; Lehmann 1993; Paramasivam 2011). The terminal-suffix of a finite verb is used to realise multiple types of information such as number, person, gender, and rationality (or status) (Pope 1979; Lehmann 1993). As for other morphosyntactic features, Tamil has singular and plural values for number, 1st/2nd/3rd person values, and three gender values, namely masculine, feminine and neuter. In addition to these three genders, a fourth class called 'epicene' is used to mark the 3rd person plural forms of rational entities (Lehmann 1993). Entities in Tamil are fundamentally classified into rational or irrational. Entities are termed *rational* if they are perceived as being able to think on their own, whereas the rest are termed *irrational*. This is different from splits found otherwise in terms of human vs. non-human, or animacy. For instance, infants are considered to be irrational like other animals or inanimate objects even though infants are human and animate. Further, when a human entity behaves in an insane manner, that entity is morphologically classified as irrational. The above concepts are used in choosing the correct terminal-suffix.

In addition to simple verbs, Tamil also has complex or compound verbs that have more than one verbal root within them, which may express mood, aspect, negation, interrogativity, emphasis, speaker perspective, and conditional and causal relations (Annamalai et al. 2014). Agesthialingom (1971) claims that Tamil can have up to four verbal roots in one verb. For instance, there are four verbal roots in the complex verb in (5): *vaa* (come), *koḷ* (hold), *iru* (be) and *iru* (be). The *koḷ* (hold) and *iru* (be) in the middle together signal continuous aspect. Further, as shown in the example in (5), in verbal conjugation, only the last verb in the sequence takes tenses and person, number and gender (png) marking. All the preceding verbs appear either in a participial form or an infinitival form.

(5) வந்துகொண்டிருந்திருக்கிறான்
vantukoṇṭiruntirukkiraan
vantu-koṇṭiru-iru-kkir-aan
come.VPART-hold_be.VPART-be-PRES-3SMR
'(he) has been coming'

A complex verb in Tamil can be written as two tokens, as in (6), or as a single token as in (7). If a complex verb is written as one word as in (7), then the analyser should provide a proper analysis, including an identification of all the verbal roots in it.

(6) வாங்கச் செய்தான்
vang-a-c sei-t-aan
buy-INF-SANDHI-C do-PAST-3SMR
'(he) made someone buy'

(7) வாங்கச்செய்தான்
vang-a-c-sei-t-aan
buy-INF-SANDHI-C-do-PAST-3SMR
'(he) made someone buy'

In our development of *Thamizhi*Morph, we have so far only handled instances of up to two verbs which are written together. We have analyzed those cases in which the second word in a complex verb functions either as an auxiliary or a light verb. For instance, Sarveswaran and Butt (2019) show how the verb கொடு (*koṭu*) 'give' functions as a light verb when it occurs as the second verb in a complex verb. We note that more studies along these lines need to be done in order to fully analyse the functions and structure of complex verbs in Tamil.

As a step towards this goal, we have identified a set of verbs (Boologarambai 1986) which form complex verbs together with the main verb. We have categorised these according to their structure and function, based on discussions in Boologarambai (1986), and our study, as shown in Table 1.

3.2.1 Verbal paradigm

Tamil verbs can be classified on the basis of criteria that can be either morphological, syntactic or semantic (Paramasivam 2011). Many scholars, including Lisker (1951), Graul (1855), and Arden (1962) have classified verbs based on their morphophonemic changes as part of the conjugations. Graul (1855) has provided an early classification on which other scholars have built their proposals, including Irākavaiyaṅkār (1958) and Sithiraputhiran (2004). Graul's classification was also adapted for the Tamil lexicon project (Rajaram 1986). This classification of Tamil verbal lemmas includes 12 categories or classes and is based on the tense markers

Table 1 List of complex verb forms used in *Thamizhi*Morph

Verb types	Verb roots	Structure of a finite verb
Aspect	இரு (*iru*) 'be', கொண்டிரு (*koṇṭiru*) 'comprehend', விடு (*viṭu*) 'leave'	main verb +VP +ASPECT-VERB +TENSE +PNG
Attitude	போ (*po*) 'go' போடு (*poṭu*)'put', தள்ளு (*taḷḷu*) (push), தீர் (*tiir*) 'solve', தொலை (*tolai*) 'get lost'	main verb +VP +ATTITUDE-VERB +TENSE +PNG
Light Verb	இடு (*iṭu*) 'put', கொடு (*koṭu*) 'give', பார் (*paar*) 'see', வா (*vaa*) 'come'	main verb +INF +NON-ATTITUDE-VERB +TENSE +PNG
Mood	வேண்டு (veeṇṭu) 'want', இரு (iru) 'be', கூடு (*kooṭu*) 'may', முடி (*muṭi*) 'finish'	main verb +INF +MOOD-VERB +TENSE +PNG
Causative	பண்ணு (*paṇṇu*) 'make', செய் (*sei*) 'make', வை (*vai*) 'cause'	main verb +INF +CAUSATIVE-VERB +TENSE +PNG
Passive	படு (*paṭu*) 'suffer', பெறு (*peru*) 'get'	main verb +INF +PASSIVE +TENSE +PNG

on the verbs. We have also adopted this classification and extended it slightly, as shown in Table 2. We have furthermore separately incorporated complex and irregular verbs. As shown in Table 2, when the tense markers conjugate, those are not just added to the root via concatenation. Many changes or new letters may be introduced, and these constructions are handled using alternation rules. For instance, in class 12, when the past tense marker is coined, it is not just added to the root (நட +த் (*naṭa+t*). Instead, the letter ந் (*nt*) is inserted: நட+ந்+த் (*naṭa+n+t*). A similar behaviour has been observed in other classes as well.

In addition to these classes, we have also identified five irregular verbs that are processed separately within the paradigm: காண் (*kaaṇ*) 'see', வா (*vaa*) 'come', சா (*saa*) 'die', தா (*taa*) 'give', வே (*vee*) 'boil'.

3.2.2 Verbal conjugational 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. Rajaram (1986) has identified 21 forms for each verb from a pedagogical perspective. On the other hand, Kumar et al. (2010b) claim that a Tamil verb lemma can take up to 8,000 forms although not all are listed or found in the literature. In our *Thamizhi*Morph we have implemented 260 inflectional forms. 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

Table 2 The Tamil Verbal Paradigm used for *Thamizhi*Morph

No.	Class name	Past tense marker	Present tense marker	Future tense marker
1	செய் (*sei*)	த் (*t*)	கிற், கின்ற் (*kir, kinr*)	வ், உம் (*v, um*)
2	ஆள் (*aaḷ*)	ட் (*ṭ*)	கிற், கின்ற் (*kir, kinr*)	வ், உம் (*v, um*)
3	கொல் (*kol*)	ற் (*ṛ*)	கிற், கின்ற் (*kir, kinr*)	வ், உம் (*v, um*)
4	கடி (*kaṭi*)	த் (*t*)	கிற், கின்ற் (*kir, kinr*)	வ், உம் (*v, um*)
5	அஞ்சு (*angu*)	இன் (*in*)	கிற், கின்ற் (*kir, kinr*)	வ், உம் (*v, um*)
6.1	அடு (aṭu)	ட் (*ṭ*)	கிற், கின்ற் (*kir, kinr*)	வ், உம் (*v, um*)
6.2	நகு (*naku*)	க் (*k*)	கிற், கின்ற் (*kir, kinr*)	வ், உம் (*v, um*)
6.3	உறு (*uru*)	ற் (*ṛ*)	கிற், கின்ற் (*kir, kinr*)	வ், உம் (*v, um*)
7	உண் (*uṇ*)	ட் (*ṭ*)	கிற், கின்ற் (*kir, kinr*)	ப், உம் (*p, um*)
8	தின் (*tin*)	ற் (*ṛ*)	கிற், கின்ற் (*kir,kinr*)	ப், உம் (*p, um*)
9	கொள் (k*oḷ*)	ட் (*ṭ*)	கிற், கின்ற் (*kir,kinr*)	ப், உம் (p,um)
10	நில் (*nil*)	ற் (*ṛ*)	கிற், கின்ற் (*kir,kinr*)	ப், உம் (*p, um*)
11	அபகரி (*abakari*)	த் (*t*)	க்கிற், க்கின்ற் (kkir, kkinr)	ப்ப், உம் (*pp, um*)
12	நட (*naṭa*)	த் (*t*)	க்கிற், க்கின்ற் (kkir, kkinr)	ப்ப், உம் (*pp, um*)

can easily be added to the system without the need for any additional programming using our Meta-Morph rules (Sarveswaran et al. 2019).

3.3 Nominal morphology

Nouns in Tamil are primarily marked for case and number. Number and case suffixes are bound morphemes which are added to stem forms or nouns with an oblique suffix. In addition, the oblique suffix, euphonic and other phonologically motivated material also appears on nouns (Caldwell 1875; Shanmugadas 1982; Lehmann 1993). Oblique nouns are formed by doubling of the last consonant, or by adding oblique suffixes such as அம் (*am*), அத்து (*attu*), and அற்று (*aṭṭu*). Lehmann (1993) shows that there are some free morphemes that also mark case.

Tamil noun stems are singular by default. The suffix கள் (*kal*) expresses plurality on most nouns (Nuhman 1999). The morphology of a Tamil noun is depicted in the formula (8). Euphonic morphs such as அன் (*an*) and இன் (*in*) are purely phonological increments (Lehmann 1993). (9) shows different ways a noun can

be chopped into parts according to the template in (8). A stem can take only one oblique; however it can take multiple instances of euphonic morphs.

(8) Noun = root (+plural) (+oblique) (+euphonic)* (+case)

(9) (a) மரங்களினால் (*marangkalinaal*) 'by trees'

மரம்	கள்	இன்	ஆல்
maram	*kal*	*in*	*aal*
tree	PL	EUPH	INST

(b) மரத்தினுக்கு (*maratinukku*) 'to a tree'

மரம்	அத்து	இன்	கு
maram	*attu*	*in*	*ku*
tree	OBL	EUPH	DAT

(c) ஆவினுக்கு (*aavinukku*) 'to a cow'

ஆ	இன்	உ	கு
aa	*in*	*u*	*ku*
COW	EUPH	EUPH	DAT

Traditional grammarians have identified 8 cases including a vocative (Senavaraiyar 1938; Thesikar 1957). However, modern linguists (Nuhman 1999; Paramasivam 2011; Lehmann 1993) argue that the instrumental case proposed in traditional grammar should be treated as two, namely instrumental and sociative. In our analyser, we adapted this modern classification, and the 9 cases are shown in Table 3. This table also shows some example case suffixes or markers which we have handled in our analyser.[6]

In addition, Tamil also has several free morphemes to mark locative, sociative, ablative and instrumental cases (Lehmann 1993). For instance, சாவியால் (*saaviyaal*) 'with key' can also be written as சாவி மூலம் (*saavi muulam*). However, these free morphemes are not always used to mark case. *Thamizhi*Morph currently does not identify these free morphemes as case markers. They are so far marked with their original lexical category, such as noun, e.g. மூலம் (*muulam*) 'origin', or verbal participial, e.g. கொண்டு (*koṇṭu*) 'take/hold'. However, one can build a tool on top of *Thamizhi*Morph to easily identify the case marking functions of these free morphemes on the basis of the POS tag of the adjacent words.

The plural suffix is a bound morpheme in Tamil that is marked by கள் (kal). However, this marker is also used as an honorary marker in the present Tamil usage, especially with the third person pronoun அவர்-கள் (avar-kal) "he.3SEH" (them).

[6] Thesikar (1957) lists 28 locative markers and 10 vocative markers that are rarely used in the present context, therefore, we have not included them in this version of *Thamizhi*Morph.

Table 3 Cases considered for *Thamizhi*Morph

Case	Morphs/Case marking suffixes	Example
Nominative	–	மரம் (*maram*) 'tree'
Accusative	ஐ (*ai*)	மரத்தை (*marattai*)
Instrumental	ஆல் (*aal*)	மரத்தால் (*marattaal*)
Sociative	ஒடு, உடன் (*oṭu, uṭan*)	மரத்துடன் (*marattuṭan*)
Dative	கு,க்கு,அக்கு,உக்கு (*ku,kku,akku,ukku*)	மரத்துக்கு (*marattukku*)
Ablative	இல்/இன்+இலிருந்து (*il/in+iliruntu*)	மரத்திலிருந்து (*marattiliruntu*)
Genitive	அது, உடைய, இன் (*atu, uṭaiya, in*)	மரத்தின் (*marattin*)
Locative	இல், இடம் (*il, iṭam*)	மரத்தில் (*marattil*)
Vocative	ஆ, ஏ, ஈ (*aa,ee,ii*)	மரமே (*marame*)

The rationality and gender of nouns are important information that will be also shown in the analysis, because this is valuable information for syntactic and semantic processing.

Tamil does not have a definite marker. The definiteness of nouns is expressed with demonstrative markers, or by using the accusative case marker in irrational objects (Lehmann 1993). The object of a sentence is marked by accusative case, however, the accusative case marking is compulsory only for rational objects (Lehmann 1993; Nuhman 1999). Therefore, when an irrational noun has an accusative marker, it is also marked for definiteness.

3.3.1 Nominal paradigm

Rajendran (2009) has proposed a paradigm for noun morphology with 26 classes based on their morphophonological properties. Among these 26 classes, 9 classes are used to capture the morphophonological rules pertaining to pronouns. Pronouns take different forms when inflecting for a case suffix.

In our noun paradigms, we have identified 38 classes for pronouns that include personal, possessive, and interrogative pronouns. We found that although many pronouns are subject to the same morphophonological rules, they produce different analyses or lexical strings. Therefore, these have been sorted into different classes.

The paradigms we used for the nouns other than pronouns is shown in Table 4. We have picked one word to represent each class, and the classes are named on the basis of that representative lexical item. We have chosen to differentiate between classes 6 and 7 even though they have the same last character in the orthography (டு (ṭu)).This is because the nouns in these two classes do differ in their conjugational patterns.

3.3.2 Nominal conjugational forms

We used 36 conjugational forms for Tamil nouns. These cover plural and case conjugations, along with external *Sandhi* markers. Each noun root takes case markers both in its singular form and plural form. Further, nouns in their dative or accusative forms can also take one of four external *Sandhi* markers. Altogether, we have 36

nominal conjugational forms. It is common to suffix postpositions to nouns. We are in the process of also including such constructions as part of *Thamizhi*Morph.

4 Related work

4.1 Approaches for developing morphological analysers

There have been rule-based, machine learning and deep learning approaches proposed and developed for morphological analysers of various languages, including Dravidian ones. However, except for Premjith et al. (2018) for Malayalam, no deep learning-based attempts have been taken in the development of a morphological analyser for any Dravidian languages. This may be due to inadequate data available to train a deep learner, as stated in Bhattacharyya et al. (2019). On the other hand, rule-based approaches yield immediate and high quality analyses and have therefore been widely employed for similar languages.

4.2 Morphological analysers for South Asian languages

A number of studies have been done on FSMs for South Asian languages. One of the earliest was Bögel et al. (2007) for Urdu, which 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. He has identified different classes of nouns, pronouns, verbs, adjectives, numerals, adverbs, conjunctions, postpositions, and particles for which he then implemented a pilot morphological analyser and generator using the two-level morphology approach and XFST tool. Rahman (2016) has developed an analyser and generator for Sindhi as part of his work on a grammar development for Sindhi. He also used XFST, which he then integrated within his grammar.

4.3 Tamil morphological analysers

Antony and Soman (2012) carried out a survey on the state of affairs of computational morphology across Indian languages, and 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 via the Internet. The rest have been carried out since 2010. Among those five efforts, Kumar et al. (2010a, b) and Menaka et al. (2010) are available for download in binary form yet without any data sets.

Menaka et al. (2010) and Kumar et al. (2010a) have used rule-based approaches which only perform morphological generation. On the other hand, Kumar et al. (2010b) used machine learning for the morphological analysis and generation of

Table 4 The Tamil Nominal Paradigm used for *Thamizhi*Morph

No.	Class name	Plural Marker	Sample case markers
1	கடா (*kaṭaa*) or பசு (*pasu*)	க்-கள் (*k-kaḷ*)	வை, வால், வுடன், வுக்கு (*vai, vaal, vuṭn, vukku*)
2	எலி (*eli*) or நெய் (ney)	கள் (*kaḷ*)	யை, யால், யுடன், யுக்கு (*yai, yaal, yuṭn, yukku*)
3	ஈ (*ii*)	க்கள் (*kkaḷ*)	யை, யால், யுடன், யுக்கு (*yai, yaal, yuṭn, yukku*)
4	நாள் (*naal*) or கால் (*kaal*)	கள் (*kaḷ*)	ை, ா, ு, ோ (*i, aa, u, oo*)
5	பலர் (*palar*)	-	ை, ா, ு, ோ (*i, aa, u, oo*)
6	காடு (*kaaṭu*)	கள்(*kaḷ*)	ட்டை, ட்டால், ட்டுக்கு, ட்டோடு (*ṭai, ṭaal, ṭukku, ṭooṭu*)
7	வண்டு (*vandu*)	கள் (*kaḷ*)	ை, ா, ு, ோ (*i, aa, uu, oo*)
8	ஆறு (*aaru*)	கள் (*kaḷ*)	றை, றால், க்கு, றோடு (*r̲ai, r̲aal, kku, r̲ooṭu*)
9	கண் (*kan*)	கள் (*kaḷ*)	ணை, ணால், ணுடன், ணுக்கு (*ṇai, ṇaal, ṇuṭan, ṇukku*)
10	பொன் (*pon*)	கள் (*kaḷ*)	னை, னால், னுடன், னுக்கு (*nai, naal, nuṭan, nukku*)
11	மாணவன் (*maanavan*)	ர்கள் (*rḷ*)	னை, னால், னுடன், னுக்கு (*nai, naal, nuṭan, nukku*)
12	புல் *(pul)*	ற்கள் (*r̲kal*)	லை, லால், லுடன், லுக்கு (*lai, laal, luṭan, lukku*)
13	முள் (*mul*)	ட்கள் (*ṭkaḷ*)	ளை, ளால், ளுடன், ளுக்கு (*ḷai, ḷaal, ḷuṭan, ḷukku*)
14	நாள் (*naal*)	ட்கள் (*ṭkaḷ*)	ை, ா, ு, ோ (*i, aa, uu, oo*)
15	மரம் (*maram*)	ங்கள் (*ṅkaḷ*)	த்தை, த்தால், த்துக்கு, த்தோடு (*ttai, ttaal, ttukku, ttooṭu*)
16	சுவர் (*suvar*)	கள் (*kaḷ*)	ற்றை, ற்றால், ற்றுக்கு, ற்றோடு (*r̲r̲ai, r̲r̲aal, r̲r̲ukku, r̲r̲ooṭu*)

Tamil. They 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 corpus. However, neither data sets, sources nor any detailed documentation are available, except for a sample corpus with 270,000 tokens. The extendability of this work to aid grammar development is also questionable, and would need to be researched. An email exchange with the authors has established that they do not work in 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%. Only the research publication could be retrieved, and there are no associated data sets or rules available in the paper, or online. Lushanthan et al. (2014) have proposed a morphological analyser and generator for Tamil, which has been implemented using XFST. The authors have used transliteration to handle the Tamil script, given that the current version of XFST has rendering issues, although it supports Unicode internally. The authors have considered 2,000 noun and 96 verb stems as part of their analysis and

generation. They have tested the 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 in India developed a morphological analyser in 2001 called *Atcharam* that has recently been added to the GitHub repository.[7] It was developed for TAB (TAmil Bilingual) encoded text as a stand-alone application using Java. There is, however, no detailed technical documentation or rule set, although some data in the form of a list of words are available in the repository. These 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[8] has 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.[9] This is also developed as a stand-alone application using Java covering the analysis of verbs and nouns. However, no 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 (cettān) '(he) died' do not seem to be handled, as no analysis is generated. 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 (*carvēśvara n*), a proper noun, showed that it is made up of the root of சர்வே (*carvē*), the future tense marker வ் (*v*), and the past tense marker ன் (*n*). That is, it not only mistakes a proper noun for a verb, but it also provides a completely wrong analysis with two contradictory tense markers. Additionally, 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, we did however manage to run this tool, yet since there is no proper documentation, it would be difficult for anyone to extend this stand-alone Java tool to make it usable for a particular need.

There is a Tamil Shallow Parser,[10] published in 2009, which also provides morphological information in addition to POS and Chunking. However, there are no papers which cover the development of the Tamil shallow parser. The authors of this tool have developed similar tools for Hindi, Telugu and Bengali, as documented in Avinesh and Karthik (2007). The authors report results for POS and shallow parsing, but not for the morphological analysis. It is not clear what approach has been used, particularly for the morphological analysis as this is not discussed or evaluated. The analysis of this tool comes in a custom version of Shakti Standard Format (SSF) (Bharati et al. 2007), which is not detailed in any of the literature found online. The original tool which was published in 2009 is also available for download. The data and rules in this tool are encrypted. However, we were not successful in executing it.

[7] https://github.com/tacola-auceg/morpha_ta.
[8] https://github.com/Pranavan135/Tamil_Morphological_Analyzer.
[9] https://github.com/tacola-aucse/Morphological-Analyzer-For-Tamil.
[10] http://ltrc.iiit.ac.in/analyzer/tamil.

The first author of this tool has now re-implemented it using Python.[11] However, the new version no longer includes morphological parsing. We therefore used the online demo to do testing, and the results are reported under the evaluation section.

There has also been an attempt to develop a Morphological analyser for Tamil using a support vector machine (Mokanarangan et al. 2016). Although the writers have reported an accuracy of 98.73%, their system is not available, and it is not clear what data sets or analyses have been used for training and testing, as there is no data available in the paper, or online.

5 Design choices

Our research on existing Tamil morphological analysers has shown that existing analysers either could not be not found, are incomplete, or not maintained. All of these tools are dependent on versions of various programming languages, and the logic seems directly coded in that particular programming language. This makes these tools difficult to maintain, test or extend. Also, some of these applications process text in ASCII (i.e., transliterated) format, and do not support Unicode encoding. Since Unicode has now become the de-facto method of encoding text, all applications are expected to have Unicode support. Further, we are in the process of developing a ParGram style computational grammar for Tamil, which requires a morphological analyser with a good, precision coverage, implemented with the use of a finite-state approach that interfaces with the grammar.

Therefore, we decided to develop a Finite-State transducer based morphological analyser. We wanted to make sure that our tool is technology, or programming language neutral so that it can be accessed via any programming language without wrapping it with an API (Applications Programming Interfaces). We also wanted it to be light-weight, so that it could be run on any commodity hardware, and be open source so that anyone can take it and extend it as needed or desired.

5.1 Technology stack

The application of Deep Learning in almost every NLP task has become common in the computational world. However, it has not yet become the state of the art for morphological analysis. It is essentially the lack of sufficient quality data that is the bottleneck for the application of deep learning approaches (Marcus 2018) and most Indic languages, including Tamil, do not have sufficient annotated data needed for supervised machine learning.

On the other hand, Finite-State Transducers (FST) have shown proven success in the past for morphologically rich South Asian languages as discussed in Sect. 4. Moreover, the development of a computational grammar using Lexical Functional Grammar and XLE for the ParGram project (Butt and King 2002) is also in our

[11] https://github.com/avineshpvs/indic_tagger.

project pipeline, which requires a Finite-State morphological analyser. Therefore, it was decided to use a FST. In addition to morphological analysis, FST can also be used for morphological generation; this is an added advantage.

There are several currently available tools with which to implement an FST-based morphological analyser. These include XFST, OpenFST, HFST, and Foma. Among these, XFST has been the standard tool for developing morphological analysers. This is because an XFST-based morphological analyser can easily be integrated into a computational grammar built using XLE. However, XFST has limited support for Unicode characters, especially for complex ones like Tamil. Additionally, it is a closed source, and proprietary tool. Foma, on the other hand, has support for Unicode, and is an open-source software that can be easily extended to web applications. For these reasons it was decided to use Foma to implement our morphological analyser.

5.2 Scope of annotations

We decided to capture and encode all available information that words express via their form. Apart from POS and morphological information, we have therefore also represented morphophonological information.

Internal *Sandhi* refers to a phonological process triggered across two morphs within a (prosodic) word, external *Sandhi* is triggered when such a phonological process happens at the boundary of two words. External *Sandhi* can occur when the second word begins with one of the following consonants: க் (k), ச்(c), த் (t), ப் (p). The *Sandhi* is triggered when further licensing conditions are met — a detailed description of these lies outside the scope of this paper. While internal *Sandhi* is purely morphophonological in nature, external *Sandhi* is also subject to syntactic or semantic constraints. Since Tamil orthography closely reflects the phonology of the language, a *Sandhi*'s effects on the orthography must necessarily be dealt within the development of a resource such as a computational grammar and morphological analyser. For instance, in the clause காளையைப் பிடித்தான் (*kalaiyaip pidittan*) 'he caught the bull', the first word *kalaiyaip* can be analysed as in (10), where ப் (*p*) is a *Sandhi* (SANDHI-P). Here the SANDHI-P comes in via assimilation after the accusative because the following word begins with a பி (pi). Similarly, we have also handled க் (SANDHI-K), ச் (SANDHI-C), and த் SANDHI-T) in our analyses.

(10) காளையைப் (*kalaiyaip*)

காளை	ஐ	ப்
kalai	yai	p
bull	-ACC	-SANDHI-P

Tamil also has two euphonic increments (Lehmann 1993): அன் (*an*) and இன் (*in*), which can be added onto nouns and verbs. These increments are also captured by our analysis, even though these are purely phonologically motivated.

Apart from the morphological features which are carried by morphs in a word, we have also included the lexically specified features shown in Table 5 in our analysis. Since it is not always possible to extract these features orthographically, we

thought that such information should be included additionally in the analysis, as these are useful for developing applications or resources such as POS taggers and computational grammars.

5.3 Morpheme labels

We use our own labelling scheme in our morphological analyser though there is an effort to harmonise morpheme labels across languages and tools, especially with a cross-lingual morphological transfer in mind, for example, the UniMorph project (Kirov et al. 2016). However, UniMorph does not capture all our required concepts, such as rationality of nouns, and the strong/weak nature of verbs, which are necessary for the morphological processing of Tamil. In addition, since we have developed this analyser having grammar engineering in mind, it is always good to mark the morpheme information of a single morph together. For instance, person, number, gender and rationality of a given noun can be marked by a single morph in Tamil. It is thus easier to mark it as a single morpheme in order to reduce the complexities in modelling. In contrast, the UniMorph project proposes separate labelling for all of this morphological information. Taking into account all of this, it was decided to design a transparent scheme for our morpheme labels, which can then be mapped to other annotation schemes as required.

Analyses of each word are given the following form in our system:

$$root| + morpheme1 = morph| + morpheme2 = morph|...$$

Apart from the morpheme information, the morph which corresponds to the morpheme is also recorded in the analysis for future use. Additionally, each morpheme is separated using a morpheme boundary '|', similar to what is used by Beesley and Karttunen (2003) to mark term boundaries (they use 'TB'). We made our choice in part because "|" is the symbol used in Universal Dependencies (UD) to separate features there.[12]

6 *Thamizhi*Morph

Based on the design choices outlined in the previous section, we have developed a morphological analyser and generator for Tamil using a Finite-State approach with the aid of Foma. This section outlines the architecture of our tool, including the preprocessing steps, data gathering approaches, compilation of rules, and the development of the tool.

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

Table 5 Lexically specified features

Morpheme label	Meaning
Fin	Finite
Nonfin	Non-finite
Weak	Weak verb: Paramasivam (2011) discusses how the weak/strong nature of a verb can be used to determine transitivity, volitivity, affectedness and ergativity in most cases, even if he does not agree with this argument. However, for our purposes, we have also marked this information as it may be useful morphosyntactic information for our grammar engineering.
Strong	Strong verb
Verb	Verb
Noun	Noun
Rat	Rational
Irra	Irrational
Sim	Simple verb: it has a single verb root
Complex	Complex verb: it has more than one verbal roots, like passives

6.1 Pre-processing

Due to the nature of the Tamil Unicode encoding and input methods, a character can be represented by multiple code sequences. For instance, the letter ஹ்ம can be represented by either: ஹ + ◌்ம or ஹ + ◌் + ◌ம. We have, therefore, developed a script to convert all input to Unicode normalised form before being fed to the system for analysis or generation. That ensures that each letter and word is represented in a unique manner.

6.2 Compilation of Lexicons

Lexicons for Tamil verbs, nouns, and other particles have been compiled from various sources as outlined below via books, a dictionary, and corpora. The words were then classified on the basis of the paradigms outlined in Sects. 3.2 and 3.3. Adjectives, adverbs, and other particles such as conjunctions have been compiled as separate lists.

6.2.1 A lexicon of verbs

A lexicon of 3300 lemmata of Tamil verbs have been compiled from the following two verified sources:

1. Ramakrishnan (2014) has identified 369 of the most frequently used verbs in Modern Tamil. This analysis is based on a corpus of 7 million tokens compiled from the web and has taken into account expert advice on linguistic matters.

This list has been included in the contemporary Tamil dictionary Cre-A (Ramakrishnan 2014).

2. Irākavaiyaṅkār (1958) surveyed the Tamil classic literature up until 1958, where he identified 3124 lemmas, and categorised these into 12 classes as per the classification proposed by Graul (1855) and Sithiraputhiran (2004). However, some of these forms are not used in the contemporary language. Nevertheless, since the analysis of these verbs is necessary in order to process historical Tamil texts, the entire list has been used for the development of our FSM.

In addition, a lexicon of complex verbs has been manually constructed by joining the infinitival form or verbal participial form of verbal roots, together with secondary verbs (Boologarambai 1986) as identified in Sect. 3.2 and in Table 1. For instance, all the modal complex verb constructions are done by joining the infinitival form of the verb together with a modal auxiliary verb, as in (11) (a). Similarly, aspectual markers are always joined with a verbal participial form of the root verb, as in (11) (b).

(11)

(a).

செய்ய	+ வேண்டும்	= செய்யவேண்டும்
(*seiy-a*)	+ (*vendum*)	= (*seiyavendum*)
do-INF MARKER	+ want.FUT	= do.INF.want.FUT
'want to do'		

(b).

செய்து	+ முடித்தான்	= செய்துமுடித்தான்
(*seiy-tu*)	+ (*muditaan*)	= (*seiytumuditaan*)
do-VPART MARKER	+ finish.PAST.3SM	= do.VPART.finish.PAST.3SM
'(he) did finish'		

6.2.2 A lexicon of nouns

A lexicon of 80,000 Tamil nouns has been collected from online databases, glossaries, and corpora. An initial level of cleaning was additionally conducted in order to ensure that these are proper words, and that they adhere to Tamil orthography and morphology. To ensure this, we developed a Python script to filter out words that begin and end with letters that are unacceptable according to the Tamil grammar (Thesikar 1957), and its current usage. According to traditional Tamil grammar, a word cannot start with the letters ர (*r*) and ற (ṟ), yet these are being used widely in the present context. We adapted a corpus-based approach to identify the current usages of this type, and in doing so, we amended our morphological analyser.

6.3 Meta-Morph rules

We integrate the novel concept of Meta-Morph Rules that we developed and presented in Sarveswaran et al. (2019). Meta-Morph Rules are lexical rules in the form of metadata that is fed into the development of our Foma morphological analyser. As discussed above as part of the review of Tamil morphological analysers, most of the previous efforts at encoding the morphotactics of Tamil have been deeply coupled with a particular programming logic. Other efforts have relied on heavy manual effort.

The definition and use of Meta-Morph rules help us to focus upon the analysis of the language without the distraction of being bound by a particular programming logic. It additionally allows for the automation of the generation of lexical entries, which, when done manually, is not only a tedious and time-consuming task, but also prone to error. This is particularly true for a language like Tamil where each verb may display several hundred inflections. Therefore, even if a paradigm approach is used, it is challenging to write rules, maintain them and perform regression testing without the aid of a meta-grammar.

We initially developed our MAG by entering all of the necessary lexical strings manually, which is a tedious task that took time and energy. However, this manual process helped us to understand the overall generalised morphological structure of Tamil. 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.

The idea was to find a way of stating the morphotactics needed to analyse and generate Tamil words in a manner that would be transparent, programming language independent and easy to maintain. As shown in Snippet-1, we hit upon a format that contains the following information: (1) The word classes to which the information applies (Line 1); (2) the inherent lexical specifications for that word, for instance, the classes here can be finite, simple, and indicative (Line 2); (3) the order of the morphemes (Line 3); (4) particular patterns, for example, as in Line 4 where it is stated that verbs (of the classes defined in Line 1) which contain euphonic markers (the material used to fulfill phonological phrasing requirements) are constructed only with past tense verbs, and only with a specific png marker (e.g., `pngeuph` in Line 4).

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
```

We found the writing of rules in the Meta-Morph format illustrated by Snippet-1 to be quick, easy and transparent. In terms of translating these descriptive statements

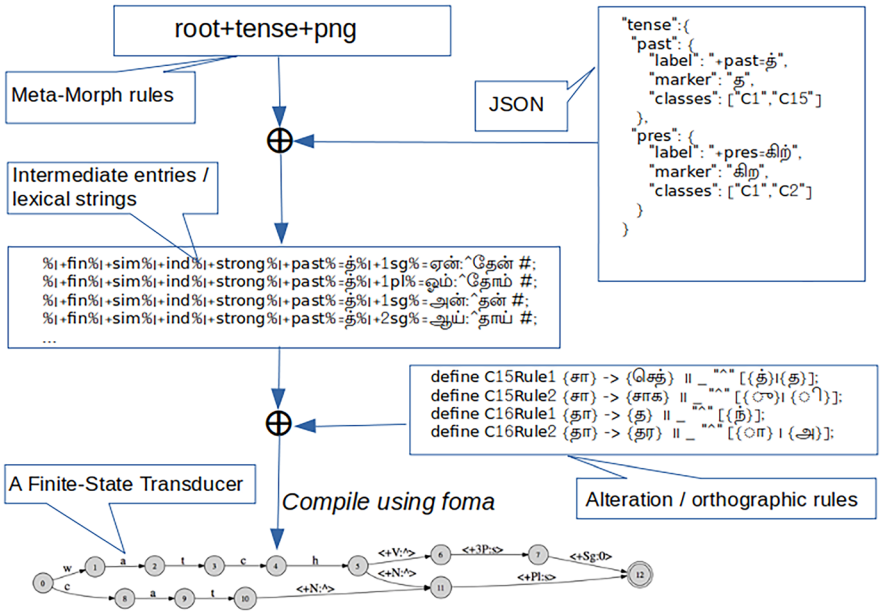

Fig. 1 The process outline: shows how actual FST is build from Meta-Morph Rules and other components

into an implementation, we found that defining feature-value pairs using JSON[13] to be the most efficient way forward. Adding in the extra step of formuiating Meta-Morph Rules coupled with the JSON knowledge base helped us to significantly accelerate the process of developing our FSM for Tamil. Adding a lexical string or new conjugation form now becomes very straightforward: All that is required is to 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. A complete set of Meta-Morph Rules for finite and indicative verbs are shown in Appendix A: Snippet-4. An overview of all the system components and processes is shown in Fig. 1.

The JSON files contain detailed morphophonological and orthographic information about the values for the labels in the Meta-Morph Rules. As shown in Snippet-2, data are stored in the JSON files as key-value pairs which are also human-readable. In addition to labels, values corresponding to each morph are also stored in the JSON files, as shown in Snippet-2. For instance, tense is defined as consisting of the values past, future (fut) and present (pres) and these values can themselves be further specified, as demonstrated for past tense, where two different possibilities are provided. This information becomes part of the lexical analysis. This data structure provides desirable flexibility for defining different tense markers and labels for different classes, and these data can be referred to at different levels when writing the

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

Meta-Morph Rules. For instance, as shown in Snippet-1, both the `tense` feature or the `past` feature can be referred to independently from one another. Furthermore, in case there was a mistake in the labelling or in the specification of the value of a marker, corrections can now easily be done directly in the descriptive but hierarchical JSON text file without needing to engage with the details of the FSM programming logic.

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"] }
}
}
```

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.

Once the Meta-Morph Rules are finalised, they can be parsed to produce actual lexical strings that are then fed to Foma to compile an FST. A parser has been developed using Python to parse these Meta-Morph Rules to generate lexical rules for Foma. A sample of a compiled Meta-Morph Rule is shown in Snippet-3. As mentioned previously, we use the pipe "|" symbol to mark morpheme boundaries. The % in Snippet-3 is used to allow us to escape special characters in the lexical string.

Snippet 3 Snippet of intermediate entries or lexical/analysis string

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

Apart from the generation of these intermediate entries, orthographical rules have been written for each class in the paradigm, as necessary, based on the description in Sect. 6.4. 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.

6.4 Orthographical rules

Writing orthographic rules is a complex task for a language like Tamil. In most cases, the affixation of suffixes is not just a mere addition to a lemma. Rather, several orthographical changes take place during the affixation process in Tamil, due to grammatical and phonological reasons. These complicate the process of writing orthographic rules. The following are common orthographical changes observed when suffixation occurs. These have been programmed to be handled by our analyser and generator:

- Morpheme/s can just be suffixed to the lemma. However, when there is a consonant followed by a vowel these two together become composite. For instance, செய் (*cey*) 'do' + த் (*t*) 'past tense marker' + ஆன் (*aan*) 'third person, masculine, singular and rational' = செய்தான் (*ceytaan*) '(he) did', where த் (*t*) + ஆ (*aa*) together form the composite character தா (*taa*).
- A new letter is introduced in addition to the morpheme. For example நட (*naṭa*) 'walk' + த் - (*t*) 'past tense marker' + ஆன் (*aan*) 'third person, masculine, singular and rational' = நடந்தான் (*naṭantāṉ*) 'walked (he)', where a letter ந் (*n*)is introduced. Some researchers consider this as a *Sandhi* letter, but some modern linguists consider ந்த் (*nt*) as the past tense marker (Nuhman 1999).
- Two letters together can form a new letter during suffixation. For example கொள் (*koḷ*) 'take' + ட் (*ṭ*) 'past tense marker'+ ஆன் (*aan*) 'third person, masculine, singular and rational' = கொண்டான் (*koṇṭaan)* 'he took', where ள் (*ḷ*) becomes ண் (*ṇ*).
- A new morpheme, called a euphonic morpheme, can be introduced in addition to the required morphemes. For instance in the word செய்தனம் (*ceythanam*) '(we) did', செய் (*cey*) 'do' + த் (t) 'past tense marker' + (அன் *an* (euphonic marker)) + அம் (*am*) (first person, neuter, plural and rational).
- Irregular verbs in Tamil may undergo a complete change in their surface form when they are conjugated. தா (*taa*) 'give' takes different forms such as தா / தரு / த (*taa/taru/ta*). For instance, in past tense forms, the lemma

becomes த (*ta*), and in present and future tense forms it becomes தரு (*taru*). The imperative form is தா (*taa*). The variations in forms yield: தந்தான் (*tantaan*) 'gave(he)', தருவான் (*taruvaan*) '(he) will give', and தா (*taa*) 'give'.
– A consonantal glide may be introduced when there are two consecutive vowels. Tamil has two such consonantal glides: ய் (*y*) and வ் (*v*).

6.5 Morphological guesser

We have also implemented a guesser to analyse nouns, verbs, adjectives, and adverbs that are not part of our lexicons. We identified common suffixes along with corresponding analyse to develop this guesser. If the guesser does not match any suffixes which we have listed, then it will recognise the word as a noun with a nominative case. This is a useful component to tackle out-of-vocabulary problems.

7 Evaluation

There are currently no benchmark data sets available for Tamil to evaluate language processing applications, including morphological analysers. Researchers tend to use their own data sets to evaluate and report results. In Sarveswaran et al. (2019) we reported an evaluation that we did using a POS tagged corpus found online. In that evaluation, we checked whether a given word in the corpus could be analysed using our analyser. This evaluation was useful for us to check the overall coverage. In this paper, we report on two experiments. In the first instance, we have taken text from an elementary Tamil text book that is part of the Sri Lankan school curriculum. This contains 612 unique words, and comprises words with a good sample coverage of different types of POS, compound words, and foreign words. We have conducted a comparative evaluation of our *Thamizhi*Morph and the IIIT's Tamil shallow parser with respect to this data set. Secondly, we have used a Tamil text from UD v2.5 to do a detailed error analysis.

7.1 Comparing *Thamizhi*Morph and IIIT Parser

We were not successful in running IIIT's shallow parser locally, and the version which has been re-implemented using Python does not have a morphological analysis part. Therefore, we relied on the available web interface.[14] Since it is a shallow parser, input has to at least be a phrase. Further, we also found that the analyser does not provide a morphological analysis based on the context of a given word. It just generates all analyses in Shakti Standard Format (SSF) annotation (Bharati et al. 2007). For instance, Fig. 2 shows the analysis for தம்பி வந்தான் - *(thambi vanthan)*

[14] http://ltrc.iiit.ac.in/analyzer/tamil.

- brother.NOM come.PAST.3RSM - 'younger brother came'. As shown in Fig. 2, the first word is always missing in the analysis produced by the IIIT parser for some reason. Figure 3 shows the analysis for the same clause from *Thamizhi*Morph. As shown in Fig. 3, in addition to the regular analysis of 'brother', the morphological analyser also provides a guessed version via the guesser integrated for nouns.

We passed the 612 words taken from the elementary school Tamil text to IIIT shallow parser, using our script, and to *Thamizhi*Morph. The results are shown in Table 6. As shown, out of 612 words, IIIT parser analysed 585 words and *Thamizhi*Morph analysed 571 words. Table 6 also shows how many of those analyses are correct analyses, and what percentage of lemmas are correctly predicted. While *Thamizhi*Morph gives always correct analysis, it failed to guess lemma associated with 12 words. This is due to the complex nature of the Tamil writing system. However, this can be corrected by adding more alternation rules.

An error analysis showed that the IIIT parser produces incorrect morphs for some verbs. For instance, in the verb தெரியும் (*teriyum*) 'will know', the future tense is marked as ப்ப் (*pp*), though it is null in this case. Another source of errors is the dropping of some characters or the introduction of new characters. Examples include the roots கொடுப்பீர்கள் (*koṭuppeerkal*) '(you) will give' and ஆச்சரியப்பட்டது (*aaccariyappaṭṭatu*) '(it) was surprised' become *கொடுபீர்(*koṭupeer*) and *ஆச்சரியம்பட்டா (*aaccariyampaṭṭaa*) instead of கொடு (*koḍu*) 'give' and ஆச்சரியப்படு (*aaccariyappaḍu*) 'be surprised' respectively. The analyses that are produced are meaningless. In addition, the analyser always provides one analysis for a given word, irrespective of the context. That is, it always resolves potential ambiguity in only one way. We conclude that even though this tool is available and accessible through the internet, it does not show sufficiently good performance.

The errors found with *Thamizhi*Morph are instances either of out-of-vocabulary items or of derivational formations that have not as yet been included in the FSM, yielding items that are unknown to the analyser. The out-of-vocabulary errors are easily remedied, as we can simply add the missing vocabulary to our lexicon table. The derivational formations are more complex and we have begun adding derivations on a case by case basis. We do note that the initial focus of our work is the development of a morphological analyser for Tamil inflectional, not derivational morphology. As such it is not surprising that *Thamizhi*Morph cannot as yet analyse instances of derivational morphology. However, we do have the flexibility to now proceed with necessary corrections and extensions.

7.2 Evaluating *Thamizhi*Morph using UD Tamil Treebank

In the second experiment, we used the text in the available Tamil Universal Dependency Treebank v2.5 to evaluate *Thamizhi*Morph. The treebank, in total, consists of 8,635 tokens from 600 sentences. There are 3,567 unique words prior to tokenisation, and this is increased to 4,055 tokens after multi-word tokenisation. Because there are inaccuracies in the multi-word annotations, and the UD annotations, we decided to work with the list before tokenisation, i.e. with the 3,567 unique words.

Home தம்பி வந்தான் Instructions

```
<Sentence id="1">
1     ((        VGNF   <fs af='வா,v,m,sg,3,,ந்த்,nw_AnY' head="வந்தான்"
                       tense="PAST" paradigm="v25" finite="Y">
1.1   வந்தான்  VM     <fs af='வா,v,m,sg,3,,ந்த்,nw_AnY' name="வந்தான்"
                       tense="PAST" paradigm="v25" finite="Y">
      ))
2     ((        NP
2.1   உந்க்    NN
      ))
</Sentence>
```

Fig. 2 A sample output from the IIIT shallow parser

தம்பி வந்தான்	
தம்பி	தம்பி தம்பி|+noun|+nom தம்பி தம்பி+guess|+noun|+sg|+nom
வந்தான்	வந்தான் வா|+fin|+sim|+ind|+strong|+past=ந்த்|+3sgm=ஆன்

Fig. 3 A sample output from *Thamizhi*Morph

Of these, *Thamizhi*Morph successfully analysed 3,023 words but failed to analyse 544 words, that is 84.7% of the words have been successfully analysed. The following are the reasons for the errors:

- 240 words were not present in our lexicon. Most of these are due to the differences in Indian Tamil and Sri Lankan Tamil. For instance, டாஸ்மாக் (*ṭaasmaak*), டிஜிபி (*DGP*) *etc*. Some of these are acronyms. Further, our lexicons did not include some proper nouns.
- there are several spelling mistakes in the treebank. In some cases the same word has been written in multiple wrong forms. For instance, ஆமதாபாத் (*aamataabaat*) 'Ahmedabad' is wrong. *கம்யூனிஸட் / கம்யூனிஸ்ட் *(camyuunisaṭ/ camyuunisṭ*) 'communist' are two forms, where one is correct and other one is wrong.
- Noun+Verb compound constructions were not analysed correctly in our tool. An example is: கவலைப்படாதீர்கள் (*kavalappaṭaateerkal*) '(you) do not worry'. We have now added this to our lexicon in terms of a complex root, adding the root N+V as a lexical item.
- analyser and guesser did not parse some Noun+Noun compound constructions correctly. An example is இரண்டரை (*iraṇṭarai*) 'two and a half'. These have now been added as items to our lexicon so that these constructions can be analysed.

In working with the UD Tamil treebank, we identified numerous annotation problems with the treebank. One necessary adjustment is the need of extending the

Table 6 *Thamizhi*Morph vs IIIT Tamil Shallow parser using a corpus of 612 words

	IIIT parser (%)	*Thamizhi*-Morph (%)
Analysis found	95.6	93.3
Right analysis found (out of successful analyses)	96.2	100
Right Lemma prediction (out of successful analyses)	94.4	97.9

current UD morphological labels to reflect all of the actual morphological information we have in Tamil. This can be done via the language-specific features proposed in the UD guidelines themselves.[15] For example, we found that the current version of the UD morphological feature inventory does not have labels to mark rationality, euphonic markers, and *Sandhi* effects.

We have also developed a tool to populate the *Thamizhi*Morph morphological annotations to the CoNLL-U format[16] which is used in UD treebanks annotation as well. We believe that our extension could be a useful resource for the creators of Tamil UD treebank.

8 Conclusion

In this paper, we described the design and performance of a Tamil Morphological Analyser cum Generator, *Thamizhi*Morph. Tamil continues to be a low-resource language in terms of the processing tools/applications available for others to use and extend. We have contributed to ameliorating this situation by developing a set of resources, including lexicons of verbs and nouns, Meta-Morph rules, and a list of 1M words which are generated from *Thamizhi*Morph that are all available for others to use and extend.[17] ,[18] The FST models published are programming language independent resources that can further be used for language processing applications. We are currently mainly using them in the context of Tamil grammar development, but we are seeking to also integrate them into the development of machine translation applications (Ranathunga et al. 2018) and spell checkers (Uthayamoorthy et al. 2019). Since we have also made our rules and lexicons openly available, our work can be easily extended to other similar languages. The Meta-Morph Rules which we have published are simple to modify and to extend and can be used as a basis for the development of a morphological analyser for other Dravidian languages.

[15] https://universaldependencies.org/docs/ext-feat-index.html.
[16] https://universaldependencies.org/docs/format.html.
[17] http://nlp-tools.uom.lk/thamizhi-morph/.
[18] https://github.com/sarves/thamizhi-morph.

Although no benchmark resources exist for an evaluation of NLP applications developed for Tamil, we designed two evaluation experiments to test the coverage and accuracy of *Thamizhi*Morph. The results are very good in that identified errors are either due to out-of-vocabulary items or derivational formations that have not as yet been implemented.

In future work, since we can generate a large amount of morphologically parsed data using *Thamizhi*Morph, we can perform several experiments. In particular we will experiment how morphological embedding will perform compared to Byte-Pair Encoding (BPE) in the context of Neural Machine Translation (NMT) specially in the context of Tamil translation. We will also explore the possibilities of using the current rule-based inflectional morphological analyser to develop a deep learning based analyser for both inflectional and derivational morphology. Further, the current analyser gives us all possible analyses for a given surface form. As a next step, we will also develop a contextual morphological analyser by merging *Thamizhi*Morph and a Part of Speech tagger. In addition, we intend to explore whether the Meta-Morph Rule interface can be further generalised and used for other South Asian Languages. We also create a benchmark data set with quality data as part of our future work. We have furthermore identified the need for more in-depth linguistic studies of verbal constructions, especially complex verbal predication so as to identify and implement the right approach in *Thamizhi*Morph.

Appendix A: A sample meta-morph rules for finite verbs

Snippet 4 Snippet of Meta-Morph rules

```
classes=C1,C2,C3,C4,C5,C61,C62,C63,C7,C8,C9,
C10,C11,C12,C13,C14,C15,C16,C17,C18
commonLabels=+verb+fin+sim
v-ind=root+tense+png
#Euphonic forms
v-euph=root+past+euph+pngeuph
v-euph=root+pres+euph+pngeuph
#imperatives
v-imp=root+imp+pngimp
v-impneg=root+imp+neg+pngimpneg
#causatives
v-caus=root+caus+tense+png
#optative
v-opt=root+opt
##### non-finite ############
commonLabels=+verb+nonfin+sim
#particles
v-illai=root+inf+illai
v-aam=root+inf+aam
v-ttum=root+inf+ttum
v-aakaatu=root+inf+aakaatu
#infinitives
v-inf=root+inf
v-infSandhi=root+inf+sandhi
##Verbal participle positive
v-vpartpos=root+vpart
v-vpartposSandhi=root+vpart+sandhi
##Verbal participle negative
v-vpartneg=root+neg+vpart
#Conditional positive forms
v-conpos=root+con
#Conditional negative forms
v-conneg=root+neg+con
#Adjectivial forms
v-adjpart=root+past+adjpart
v-adjpart=root+pres+adjpart
v-adjpart=root+futANDadjpart
v-adjpart=root+neg+adjpart
```

Appendix B: Screen capture of an analysis from *Thamizhi*Morph

*Thamizhi*Morph will provide all possible analyses for a given word. For instance, ஒசெய்யும் (*ceyyum*) can be analysed as do.ADJPART or do.FUT.3SGN or do.FUT.3PLN. As shown in Fig. 4, *Thamizhi*Morph will show all these possible analyses.

Fig. 4 *Thamizhi*Morph: Screen capture of an analysis

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=∅

Acknowledgements We would like to thank Lauri Karttunen from Stanford University, Rajendran Sankaravelayuthan from Amrita University, and Mans Hulden from the University of Colorado Boulder for their thoughts and technical support in making this work possible. We would also like express our appreciation to Maris Camilleri from the University of Essex for her support in language editing, and two anonymous reviewers for their valuable comments and inputs to improve this menu script. 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 also supported by the DAAD (German Academic Exchange Office).

Funding Open Access funding enabled and organized by Projekt DEAL.

References

Agesthialingom S (1971) A note on Tamil verbs. Anthropol Linguist 13:21–125

Allauzen C, Riley M, Schalkwyk J, Skut W, Mohri M (2007) OpenFst: a general and efficient weighted finite-state transducer library. International conference on implementation and application of automata. Springer, Berlin, pp 11–23

Annamalai E, Dhamotharan A, Ramakrishnan A (2014) Akarātiyiṉ putiya patippil taṟkālat tamiḻ ilakkaṇa viḷakkam (in Tamil), pages xxxi-xlvii. Crea-A Publishers, India

Antony PJ, Soman KP (2012) Computational morphology and natural language parsing for Indian languages: a literature survey. Int J Sci Eng Res 3:1

Arden AH(1962) A progressive grammar of common Tamil. Christian Literature Society, Chennai

Avinesh PVS, Karthik G (2007) Part-of-speech tagging and chunking using conditional random fields and transformation based learning. Shallow Parsing South Asian Lang 21:21–24

Balaram P (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

Baskaran S, Bali K, Choudhury M, Bhattacharya T, Bhattacharyya P, Jha GN, Rajendran S, Saravanan K, Sobha L, Subbarao KV (2008) A common parts-of-speech tagset framework for Indian languages. In: Proceedings of the sixth international language resources and evaluation. LREC, Morocco

Beesley KR, Karttunen L (2003) Finite-state morphology: xerox tools and techniques. CSLI, Stanford

Bharati A, Sangal R, Sharma DM (2007) SSF: Shakti standard format guide. Language Technologies Research Centre, International Institute of Information Technology, Hyderabad, India, pp 1–25. http://sampark.org.in/sampark/web/ssf-guide-4oct07.pdf

Bhattacharyya Pushpak, Murthy Hema, Ranathunga Surangika, Munasinghe Ranjiva (2019) Indic language computing. Commun ACM 62(11):70–75

Boologarambai A (1986) A study of auxiliaries in the old and the middle Tamil. Ph.D. thesis, Centre of Advanced study in linguistics, Annamalai University, India

Butt M, King TH, Nino M-E, Segond F (1999) A grammar writer's cookbook. CSLI, Stanford

Bögel T, Butt M, Hautli A, Sulger S (2007) Developing a finite-state morphological analyzer for Urdu and Hindi. In: Hanneforth T, Würzner K-M (eds) Finite-state methods and natural language processing. Potsdam University Press, Potsdam, pp 86–96 (Revised Papers of the Sixth International Workshop on Finite-State Methods and Natural Language Processing)

Caldwell R (1875) A comparative grammar of the Dravidian or South-Indian family of languages. Trübner, Stuttgart

Christo K, John S-G, Roger Q, David Y (2016) Very-large Scale Parsing and Normalization of Wiktionary Morphological Paradigms. In: Chair) NCC, Choukri K, Declerck T, Goggi S, Grobelnik M, Maegaard B, Mariani J, Mazo H, Moreno A, Odijk J, Piperidis S (eds) Proceedings of the tenth international conference on language resources and evaluation (LREC 2016). European Language Resources Association (ELRA), Paris, France. ISBN 978-2-9517408-9-1

Crouch R, Dalrymple M, Kaplan RM, King TH, Maxwell III JT, Newman P (2017) XLE documentation. Palo Alto Research Center, Palo Alto

Gary M (2018) Deep learning: a critical appraisal. arXiv preprint arXiv:1801.00631

George LH (2000) Statement on the status of Tamil as a classical language

Graul K (1855) Outline of Tamil grammar. Leipzig University, Leipzig

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

Kimmo K (1983) Two-level morphology. Ph.D. thesis, University of Helsinki

Krister L, Miikka S, Tommi P (2009) HFST tools for morphology-an efficient open-source package for construction of morphological analyzers. International workshop on systems and frameworks for computational morphology. Springer, Berlin, pp 28–47

Kumar M, Anand V. Dhanalakshmi, Rajendran S (2010a) A novel data driven algorithm for Tamil morphological generator. Int J Comput Appl 6:52–56

Kumar M, Anand V. Dhanalakshmi, Soman KP, Rajendran S (2010b) A sequence labeling approach to morphological analyzer for Tamil language. Int J Comput Sci Eng 2(06):1944–1995

Lauri K, Kenneth RB (2001) A short history of two-level morphology. ESSLLI-2001 Special Event titled—Twenty Years of Finite-State Morphology. http://www.helsinki.fi/esslli/

Lehmann T (1993) A grammar of modern Tamil. Pondicherry Institute of Linguistics and Culture, Pondicherry

Lehmann T (1998) Old Tamil. In: Stanford BS (ed) The Dravidian languages. Routledge, London, pp 75–99

Lisker L (1951) Tamil verb classification. J Am Orient Soc 71(2):111–114

Mans H (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. Association for Computational Linguistics, pp 29–32

Melanie S (2012) A rule-based morphological analyzer for Murrinh-Patha. In: Proceedings of the 8th international conference on language resources and evaluation (LREC 2012). European Language Resources Association (ELRA), Istanbul, Turkey, pp 751–758

Menaka S, Sundar RV, Lalitha DS (2010) Morphological generator for Tamil. In: Proceedings of the knowledge sharing event on morphological analysers and generators (March 22–23, 2010) p 82–96

Miriam B, Holloway KT (2002) Urdu and the Parallel Grammar project. In: Proceedings of the 3rd workshop on Asian language resources and international standardization, COLING. Association for Computational Linguistics, pp 39–45

Mokanarangan T, Pranavan T, Megala U, Nilusija N, Gihan D, Sanath J, Surangika R (2016) Tamil morphological analyzer using support vector machines. International conference on applications of natural language to information systems. Springer, Berlin, pp 15–23

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

Paramasivam K (2011) Contemporary Tamil Grammar. Adaiyaalam, Trichy

Parameshwari K (2011) An implementation of APERTIUM morphological analyzer and generator for Tamil. Parsing Indian Lang 41

Peyman P, Qun L, Andy W (2018) Improving character-based decoding using target-side morphological information for neural machine translation. arXiv preprint arXiv:1804.06506

Philipp K, Hieu H, Alexandra B, Chris C-B, Marcello F, Nicola VB, Brooke C, Wade S, Christine M, Richard Z et al (2007) Moses: Open source toolkit for statistical machine translation. In: Proceedings of the 45th annual meeting of the association for computational linguistics companion volume proceedings of the demo and poster sessions. pp 177–180

Pope GU (1979) A handbook of the Tamil language. Asian Educational Services, New Delhi

Premjith B, Soman KP, Anand Kumar M (2018) A deep learning approach for Malayalam morphological analysis at character level. Proc Comput Sci 132:47–54

Rahman Mutee U (2016) Developing a Sindhi computational resource grammar in lexical functional grammar framework. Ph.D. thesis, Faculty of Engineering Science and Technology, Isra University, Hyderabad

Rajaram S (1986) English–Tamil pedagogical dictionary. Tamil University, Thanjavur

Rajendran S (2009) Preliminaries to the preparation of a spell and grammar checker for Tamil. https://www.academia.edu/12504639. Accessed on 3 Feb 2018

Ramakrishnan S (ed) (2014) Cre-A: dictionary of contemporary Tamil. Cre-A, Chennai

Ranathunga S, Farhath F, Thayasivam U, Jayasena S, Dias G (2018) Si-Ta: machine tanslation of Sinhala and Tamil official documents. In: 2018 National Information Technology Conference (NITC), pp 1–6

Sarveswaran K, Butt M (2019) Computational challenges with Tamil complex predicates. In: Butt M, King TH, Toivonen I (eds) Proceedings of the LFG19 conference, Australian National University. CSLI, Stanford, pp 272–292

Sarveswaran K, Mahesan S (2014) Hierarchical tag-set for rule-based processing of Tamil language. Int J Multidiscip Stud 1(2):67–74

Sarveswaran K, Gihan D, Miriam B (2019) Using meta-morph rules to develop morphological analysers: a case study concerning Tamil. In: Proceedings of the 14th international conference on finite-state methods and natural language processing. Association for Computational Linguistics, Dresden, Germany, pp 76–86

Schiffman HF (2008) The Ausbau issue in the Dravidian languages: the case of Tamil and the problem of purism. Int J Soc Lang 2008(191):45–63

Senavaraiyar (1938) Tholkappiyam–Eluththathikaram. Thirumahal, Chunnakam

Shanmugadas A (1982) Aspects of Tamil Language and Grammar (in Tamil). Muttamil̲ veḷiyīṭṭuk kalakam, Sri Lanka

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

Sivaneasharajah L, Weerasinghe A R, Herath DL (2014) Morphological analyzer and generator for Tamil language. In: International conference on advances in ICT for emerging regions (ICTer), 2014. IEEE, pp 190–196

Thesikar SN (1957) Nannool Viruthiyurai. Vithiyanubalana Press, Chennai

Uthayamoorthy K, Kanthasamy K, Senthaalan T, Sarveswaran K, Dias G (2019) DDSpell: a data driven spell checker and suggestion generator for the Tamil language. In: 2019 19th international conference on advances in ICT for emerging regions (ICTer), pp 1–6