

# Tamil Morphological Analyzer Using Support Vector Machines

T. Mokanarangan<sup>(✉)</sup>, T. Pranavan, U. Megala, N. Nilusija, G. Dias, S. Jayasena, and S. Ranathunga

Department of Computer Science Engineering, University of Moratuwa,  
Moratuwa, Sri Lanka

{mokanarangan.11, pranavan.11, megala.11, nilu.11, gihan,  
sanath, surangika}@cse.mrt.ac.lk

**Abstract.** Morphology is the process of analyzing the internal structure of words. Grammatical features and properties are used for this analysis. Like other Dravidian languages, Tamil is a highly agglutinative language with a rich morphology. Most of the current morphological analyzers for Tamil mainly use segmentation to deconstruct the word to generate all possible candidates and then either grammar rules or tagging mismatch is used during post processing to get the best candidate. This paper presents a morphological engine for Tamil that uses grammar rules and an annotated corpus to get all possible candidates. A support vector machines classifier is employed to determine the most probable morphological deconstruction for a given word. Lexical labels, respective frequency scores, average length and suffixes are used as features. The accuracy of our system is 98.73 % and a F-measure of .943, which is more than the same reported by other similar research.

**Keywords:** Tamil · Morphological analyzer · Support vector machine · Natural language processing · Dravidian languages

## 1 Introduction

Morphological analysis is the process of segmentation of words into their component morphemes, and the assignment of grammatical morphemes to grammatical categories and lexical morphemes to lexemes [1]. Tamil language is morphologically rich and agglutinative. Each word is pinned with morphemes and during morphological construction, the original form of the word changes, hence making the morphological deconstruction tough.

Morphological analysis is the basis for many natural language processing tasks such as Named Entity Recognition, Part of Speech Tagging and Machine translation. Morphological analysis can provide a wealth of information. For Tamil in particular, like many other Dravidian languages, a good morphological analyzer can extract many information about a word ranging from verb or noun to tense and gender due to its rich morphology.

Previous attempts on morphological analysis for Tamil have been made using three approaches: rule based, machine learning based, and hybrid approaches that combine

both the rule based and the machine learning approaches. This paper outlines an approach that uses a morphological engine encompassing all grammar rules in Tamil that generates all possible candidates for a word along with the Part of Speech (PoS) tags for each morpheme. These PoS tags, respective frequency scores, average length and suffixes are used as features in a Support Vector Machines (SVM) classifier to select the best candidate out of the candidate list.

Rest of the paper is organized as follows. The next section discusses the previous attempts on building morphological analyzers for Tamil. Third section describes our approach and the fourth section gives the evaluation results. Final section discusses future work and concludes the paper.

## 2 Related Work

First ever Tamil morphological analyzer was built by AU-KBC Research Centre in 2003 [2]. Since then research on Tamil morphological analysis was continued in two directions, using machine learning and using rule based approaches. Selvam and Natarajan [3] carried out research on morphological analysis and PoS tagging for Tamil using a rule based approach via projection and induction techniques. Another morphological analyzer for Tamil was implemented using the sequence labelling based machine learning approach [4]. It was a supervised machine learning approach and a corpus with morphological information was used for training. Another approach used the open source platform apertium [5]. Apertium tool uses the computational algorithm called Finite State Transducers for one-pass analysis and generation, and the database is based on the morphological model called Word and Paradigm. In a very recent research, a rule-based morphological analyzer was presented [6]. Researchers have used a set of rules, a postposition suffix word list and a root word dictionary developed from classical Tamil text. Not considering all the grammar rules coupled with high ambiguity has been the problem for this approach.

Our approach drew inspiration from morphological analyzers designed for two different languages: first from an Arabic morphological analyzer [6]. In this approach, text is broken down into each of the hundreds and thousands of possible lexical labels, which represent their constituent elements including lemma ID and part-of-speech. Features are computed for each lexical token based on their local and document-level context. Based on these features the support vector machines classifier is implemented to do the classification. The second method was from a compound word splitting approach for German [7]. This approach introduced methods to learn splitting rules from monolingual and parallel corpora. These rules were then evaluated against a gold standard [7].

## 3 Our Approach

### 3.1 Outline

As show in the Fig. 1 the first step is to get all possible lexical units of a single word and annotate each lexical unit with part of speech tags. In some cases one lexical unit

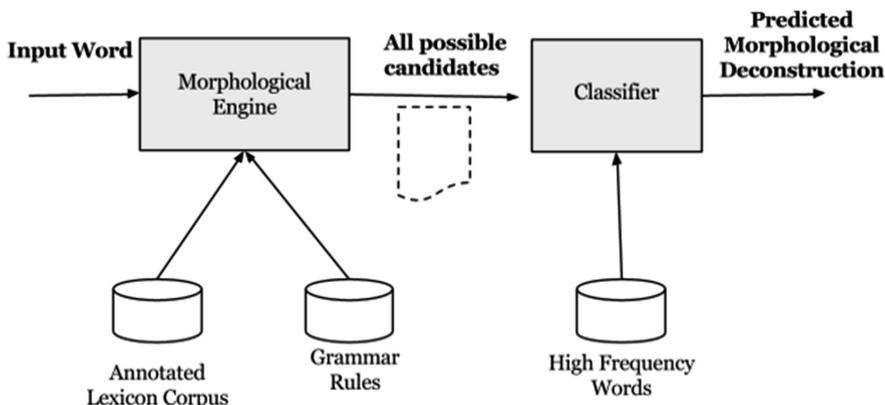


Fig. 1. Outline of the morphological analyzer

can have more than one part of speech tag. For example ஓடு ( $\bar{o}tu$ ) can mean ‘Roof’ and ‘Run’. Morphological rules of Tamil can also affect the spelling of the root.

Consider the following example for the word ஓடினான் (Transliteration -  $\bar{o}tin\bar{a}n$ , Translation – ran) (Table 1).

In the next step, the annotated lexical labels along with other features were fed into the SVM classifier. The SVM then predicts the best candidate for a certain word. The reason to choose SVM over other available options such as multilayer perceptron and boosted is the best trade-off SVM provided between accuracy and training time. This is explained in detail below in Sect. 4.

Table 1. All possible combinations for the word ஓடினான் (transliteration -  $\bar{o}tin\bar{a}n$ , translation – ran)

Morphological Deconstruction	Lexical Labels
ஓடு + இன் + ஆன்	<v><Idainilai><part>
ஓடி + ன் + ஆன்	<n><Idainilai><part>
ஓடு + இன் + ஆ + ன்	<v><Idainilai><part><part>
ஓடி + ன் + ஆ + ன்	<n><Idainilai><part><part>

### 3.2 Data Sources

To generate all possible candidates, annotate with PoS labels for each lexical label, and to get the total frequency of each word, we used two sources: a lexicon corpus along with PoS annotations, and a list of high frequency words along with the frequency score for each word.

### 3.2.1 Annotated Tamil Lexicon Corpus

Annotated Tamil lexicon corpus was obtained from an online Tamil lexicon created by University of Madras [10]. Initially this corpus had 16 different types of lexical labels but eventually we reduced to 5 types: verb, noun, adjective, adverb and other. The purpose of this reduction is to limit the possibilities of combinations of lexical labels and hence reducing the amount of training data. Table 2 illustrates a sample of how words and tags are stored.

**Table 2.** Lexicon words with tag

Word	Tag
அஃகல்	n
அஃகான்	n
அஃகு	v

### 3.2.2 High Frequency Words List

The high frequency word list was built using the usage data obtained by crawling Tamil Wikipedia and other Tamil news websites. Each entry in this list has the word and the word count. Here the word count was used to calculate the frequency score.

## 3.3 Morphological Engine

Morphological Engine is the vital part in the system. Encompassing all the grammar rules regarding morphological construction, this engine generates all possible candidates along with their lexical labels. Some of the rules in morphological engine are shown in Fig. 2.

As illustrated in the diagram. The word கிளியை (Transliteration: Kiliyai, Translation: ‘the parrot’) can be deconstructed under two grammar rules:

*Grammar rule 1:* உயிர் முன் உயிர் புணர்தல் (Transliteration – “Uyir munn uyir punarthal”, Translation: “Vowel on Vowel morphological construction”).

கிளியை - கிளி> + Yakaram> + ஐ

*Grammar rule 2:* இயல்பு புணர்ச்சி (Transliteration – “Iyalpu punarchi”, Translation: “Natural morphological construction”).

கிளியை - கிளி + யை

கிளியை - கிளி + ய் + ஐ

Based on the last letter of the first word and the first letter of the second word the grammar rules define the morphological construction. To ensure that all grammar rules and all types of morphological deconstruction is covered, two Tamil grammar books [9, 10] were followed to obtain 14 rules. Using these rules all the candidate are

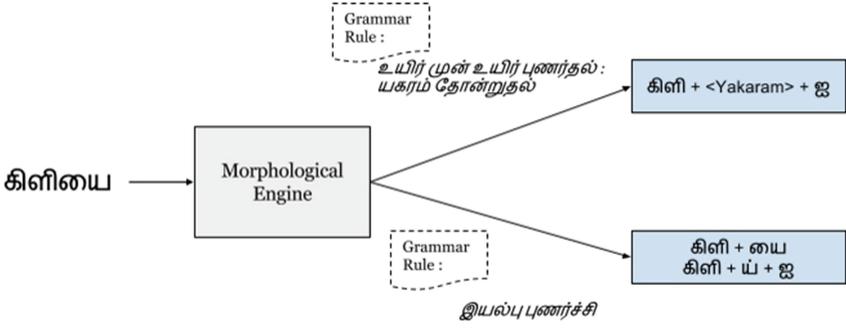


Fig. 2. Grammar rules and morphological deconstruction

generated and then lexical labels. To get all the approaches a finite state machine that uses brute force approach to get all possible combinations was implemented.

The word given in the diagram is relatively easy to deconstruct, now consider a complicated word: ஓடிச்சென்றான் (Transliteration – Oodichchendraan, Meaning – ‘He ran’).

All candidates possible for this is:

- ஓடி+<EkaramVallinam>+சென்று+ஆன்,
- ஓடு+<Idainilai>+ச்+சென்றான்
- ஓடி+<EkaramVallinam>+சென்றான்
- ஓடு+<Idainilai>+ச்+செல்+<Idainilai>+ஆன்
- ஓடு+இ+<MellinamVallinamUyirMun>+சென்றான்
- ஓடு+இ+<MellinamVallinamUyirMun>+சென்று+ஆன்

### 3.4 Classifier

#### 3.4.1 Features Set

##### Frequency Based Scores

This frequency based approach was proposed by Koehn and Knight [8] to split compound words in German. The more frequent a word occurring in a training corpus, the bigger the statistical basis to estimate translation probabilities, and the more likely the correct translation probability distribution is learned. This insight leads to define a splitting metric based on word frequency [8].

Given the count of words in the corpus, the split S with the highest geometric mean of word frequencies of its parts  $p_i$  ( $n$  being the number of parts) is selected. Here  $count(p_i)$  is frequency count the word  $p_i$  obtained from the high frequency words list.

$$argmax_s \left( \prod_{p_i \in S} count(p_i) \right)^{\frac{1}{n}}$$

Consider the following example: ஓடி+<EkaramVallinam>+சென்று+ஆன்

- Frequency scores =  $(\text{count}(\text{ஒடி}) + \text{count}(\text{சென்று}) + \text{count}(\text{ஆன்}))^{\frac{1}{3}}$   
 $= (550 + 2524 + 120)^{\frac{1}{3}}$   
 $= 14.33$

### Lexical Labels

Other important feature set is the lexical labels generated by the morphological engine. The morphology engine has been developed for the particular case of Tamil and the particular set of lexical labels. This tagging order gives more priority to more commonly occurring patterns and indirectly covers more subtle grammar patterns in Tamil.

### Suffix (விசுதி)

Tamil is a morphologically rich language with many morphemes pinned to each word. But in many cases, certain morphemes do not appear as suffixes for certain type of words. For example ‘ஐ’ suffix is not present in a verb. In retrospect, the model was fed with the final suffix of a word as a feature to the system to differentiate verbs, adjective, adverbs and noun stem based words.

### Average Length

This is a new feature that has not been tried in any previous approaches. When the model was tried on with only the above mentioned features, it was found that for some compound noun cases, the morphological deconstruction was going a step further.

For example:-

மாநகரசபை -மா+நகரம்+சபை (Expected)  
 - மா+நக+ர+ச+பை (Output without average length feature)

Therefore, to eliminate this issue, we introduced a threshold feature called average length. It is obtained by calculating the average length of the lexical parts in the candidate. This feature was found out based on the factor analysis carried out on the training data.

## 3.5 Training Data

Over 70,000 words were manually labelled and used as training data. Correct morphological disambiguation candidate was labelled as ‘Yes’ while mismatches were labelled as ‘No’.

### 3.6 Prediction

Using the training data, a probabilistic model was built using the SVM classifier. The candidates are then classified using the classifier and the one with highest probability of classified as ‘Yes’ is selected. This probabilistic model not only provides us with the best candidate but also if there is ambiguity the top candidates are displayed. This feature can come in handy while implementing a Part of Speech tagger for Tamil.

## 4 Evaluation Results

Upon generating all the candidates, the next step is to feed the data into the classifier to select the best suitable candidate. We selected SVM because of the best trade-off between accuracy and time taken to build the model. Table 3 illustrates the comparison of accuracies between various classifiers.

Table 4 illustrates the difference in accuracy by using average length and not using average length as features.

Table 5 illustrates the difference in accuracy by using average length and not using frequency scores as features.

**Table 3.** Comparisons of accuracy between various classifiers

	Multilayer perceptron	Boosted decision tree	Support vector machine
Correctly classified instances	91.236 %	85 %	98.73 %

**Table 4.** Accuracy difference between with and without using average length

Accuracy without using average length	92.83 %
Accuracy using average length	98.73 %

**Table 5.** Accuracy difference between with and without using frequency scores

Accuracy without using frequency scores	47.36 %
Accuracy using frequency scores	98.73 %

Tables 6 and 7 show the results obtained from 10-Fold cross validation test for over 30,000 words. Table 6 illustrates the overall accuracy of the system while Table 7 illustrates the detailed accuracy by class.

**Table 6.** Stratified cross validation

Correctly classified instances	98.7376 %
Kappa statistic	0.8869
Mean absolute error	0.0265
Root mean squared error	0.1033
Relative absolute error	23.9067 %
Root relative squared error	43.8834 %

**Table 7.** Detailed accuracy by class

TP rate	FP rate	Precision	Recall	F-measure	Range of coverage area	Class
0.89	0.01	0.888	0.899	0.89	0.99	Yes
0.99	0.10	0.994	0.993	0.99	0.99	No

## 5 Conclusion and Future Work

We presented a morphological engine for Tamil that uses grammar rules and an annotated corpus to get all possible candidates. A support vector machines classifier was employed to determine the most probable morphological deconstruction for a given word. Lexical labels, respective frequency scores, average length and suffixes are used as features. The accuracy of our system is 98.73 %, which is more than the same reported by other similar research.

Tamil is a morphologically rich language. Computationally, each root word of can take a few thousand inflected word-forms, out of which only a few hundred will exist in a typical corpus. This morphological analyzer which uses a different approach from previous approaches have proved to be effective.

Though the main intention of this approach is to tackle the ambiguity sometimes this approach fails when encountering name entities. It tends to break into meaningless morphological disambiguation. This is a pitfall that should be taken care of in the further researches.

Since most Dravidian language share the same characteristics, hoping that this approach can be used in other languages to get a highly accurate morphological analyzer. The analyzer not only outputs the construct the deconstructed morphology but also the lexical labels.

As future work we intend to build on this approach and along with it build a PoS tagger and Name Entity recognizer that uses the features extracted from morphological analyzer. Once these goals have been achieved we eventually hope to build a successful Tamil machine translator and eventually preserve an ancient endangered language.

## References

1. Jayan, J.P., Rajeev, R., Rajendran, S.: Morphological analyzer and morphological generator for Malayalam - Tamil machine translation. *Int. J. Comput. Appl. (0975 – 8887)* **13**(8), 15–18 (2011)
2. [Au-kbc.org](http://Au-kbc.org). Tamil Morphological Analyzer (2015)
3. Selvam, M., Natarajan, A.M.: Improvement of rule based morphological analysis and POS tagging in Tamil language via projection and induction techniques. *Int. J. Comput.* **3**(4), 357–367 (2009)
4. Anand Kumar, M., Dhanalakshmi, V., Soman, K.P., Rajendran, S.: A sequence labeling approach to morphological analyzer for Tamil language. *Int. J. Comput. Sci. Eng.* **2**(6), 1944–1951 (2010)
5. Parameshwari, K.: An implementation of APERTIUM morphological analyzer and generator for Tamil. *Probl. Parsing Indian Lang.* **11**, 41–44 (2011)
6. Akilan, R., Naganathan, E.R.: Morphological analyzer for classical Tamil texts: a rule-based approach. *Int. J. Innov. Sci. Eng. Technol.* **1**(5), 563–568 (2014)
7. Shah, R., Dhillon, P.S., Liberman, M., Foster, D., Maamouri, M., Ungar, L.: A new approach to lexical disambiguation of Arabic text. In: *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, Cambridge, Massachusetts, pp. 725–735, 09–11 October 2010
8. Koehn, P., Knight, K.: Empirical methods for compound splitting. In: *Proceedings of the Tenth Conference on European Chapter of the Association for Computational Linguistics*, Budapest, Hungary, 12–17 April 2003
9. Nuhman, M.A.: *அடிப்படைத்தமிழ்இலக்கணம்*, Revised edn, pp. 93–260. Poobalasingam Publications, Sri Lanka (2010)
10. Naavalur, A.: *தமிழ்இலக்கணம்*, 10th edn, pp. 88–180. Poobalasingam Publications, Sri Lanka (2008)



<http://www.springer.com/978-3-319-41753-0>

Natural Language Processing and Information Systems  
21st International Conference on Applications of  
Natural Language to Information Systems, NLDB 2016,  
Salford, UK, June 22-24, 2016, Proceedings  
Métais, E.; Meziane, F.; Saraee, M.; Sugumaran, V.;  
Vadera, S. (Eds.)  
2016, XV, 488 p. 100 illus., Softcover  
ISBN: 978-3-319-41753-0