Malayalam Stemmer

Arun V Krishna[1]
Department of Computer Science
University of Maryland, Baltimore County

Abstract

Morphology is a concept used in Natural Language Processing (NLP) to describe the various forms a word can take. Stemming is the method of removing the affixes from a word and extracting the stem. Malayalam is a highly agglutinative language with hundreds of potential inflections for each word. In Malayalam, an efficient stemmer has yet to be introduced. This paper presents a Malayalam stemmer that conflates words by removing suffixes.

Keywords— Suffix, Strip, Stem, Lemma

1 Introduction

Programming computers to understand natural languages such as English, Hindi, and Malayalam is becoming increasingly important, as Artificial Intelligence focuses on inducing intelligence in computers so that they can act like humans. Subfields such as Natural Language Processing, expert systems, and others have sprung up as a result of artificial intelligence. Natural language interaction between the user and the device may take the form of natural text or natural language expression. As a result, text mining has become a common topic of study in the field of Natural Language Processing. Additionally, text document processing has a number of sub-areas whose accuracy improves only when terms are reduced to their stem form, a process known as stemming.

Stemming can be thought of as a linguistic normalization mechanism. Another technique is lemmatization, which is close to stemming in several ways. Lemmatization produces a

lemma, while stemming produces a stem. The difference is that a meaningful stem is not needed. It will not take into account the POS (Parts Of Speech)tag or the sense of the expression, while the lemma will do so and generate a true and meaningful word.

Information retrieval systems use stemming to boost their efficiency. Documents are categorized as important or irrelevant in information retrieval. The basic assessment metrics are recall and accuracy. Recall returns a percentage value indicating the amount of relevant documents recovered from the total number of relevant documents. The recall is improved by stemming. The terms in a document and the incoming query are usually stemmed in information retrieval systems. A Malayalam stemmer is an important component of Malayalam computing because it paves the way for a variety of Natural Language Processing applications in Malayalam. This demonstrates the applicability of the proposed method. In Malayalam, inflections are generated by adding the root with suffixes. Malayalam has very few prefixes or infixes. The proposed work would define the suffix and, using the suffix stripping process, delete it, returning the root form or stem. The work is done in Python, a high-level programming language that is both simple and efficient.

2 Related Work

Porter Stemmer's Spanish version is used as a preprocessing method for correctly extracting text from each document picture [1]. This aids in the development of the bag-of-words vector. The output of various page classifiers is compared, with stemming being a significant module that affects the system's accuracy. For word segmentation in Mongolian, the process [2] took into account stems and suffixes. A paper written by [3] discusses the use of a predefined pattern to extract stem terms. In any language, the existence of affixes leads to a wide vocabulary. They addressed rule-based, direct, interlingua, transfer, mathematical, example-based, knowledge-based, and hybrid translation methods, all of which rely heavily on stems and affixes. The term must be stemmed for document similarity calculations such as tree based, time series, vector based, and various text clustering methods such as hierarchical, partitioning, mixture, and multilevel clustering [4].

As described in [5] stemming, or the removal of affixes, is used as a preprocessing activity for calculating term frequency and is used in the Map Reduce system for text summarization. The authors of [6] discuss factoring words into morphological components, which necessitates the extraction of the stem.

In Asian languages, such as Arabic [7] stemmers have been produced that eliminate both the prefix and suffix of a given word. Since prefix words are uncommon in Malayalam, we created an algorithm that only considers suffixes. Finite state automata are used in the Indonesian Stemmer [8]. From a list of candidates, the best word candidate is chosen. [9] contains a comprehensive survey on stemming that includes English, Czech, Bulgarian, Turkish, Greek, German, Dutch, Portuguese, and French, among other languages. Some approaches [10] address combining stemming laws and take both suffixes and prefixes into account.

[11] investigates the stemmers produced in Indian languages. A rule-based strategy is used by the Gujarathi stemmer [12] and the Bengali stemmer [13]. Stemming has limited the number of unique terms, and both understemming and overstemming have been considered. They use the minimum stem set model of stemming in [14], which gives an accuracy of 80 to 88 percent. They screened for Hindi, Marathi, Malayalam, and English, among other languages. The system accepts a word list and a suffix list as inputs. There are approaches that use a suffix list and a probabilistic approach[15]. [16] has developed an Urdu stemmer.

Finite State Automata are used to stem in Malayalam stemmer [17]. Singular nouns, plural nouns, and verbs have all been included in the development of FSA. The writers of [18] used a recursive suffix stripping process that had an accuracy of 83.67 percent. A term with three suffixes is processed three times. The device is a submodule of a sentence parser, and its accuracy is determined by the dictionaries used. Rather than root extraction, the morphological analyzer established by [19] extracts morphological features such as noun/verb, number: plural/singular, and tense: past/present/future.

3 Comparison of Stemmers in Indian Languages

Author	Language	Method	Accuracy
Vijay S R	Malayalam	Finite state automata	94.76%
Jikitsha S	Gujarathi	Substitution rules	92.41%
Das S	Bengali	Using hash table containing suffixes	96.27%
Vasudevan N	Hindi	Semi supervised stemming	84%
Ramachandran V.A	Tamil	Iterative suffix stripping	84.79%
Kasthuri M	Tamil	Imperative tense suffixes are stripped	89.42%

4 Morphology in Malayalam

Since the Malayalam language has a lot of morphology, the stemmer we created has more than 150+ suffix replacement laws. The morphology of the Malayalam language is so complicated that a single word may have several meanings. The difficulty of the language makes constructing a stemmer difficult. We may demonstrate this complexity by using a simple word (mavu), which in English means mango tree. The word (mavu) which is a noun can be attached with one preposition as (mavil), (mavulla), (mavinte), (mavo), (mavaya), (mavakkiya), (mavennal), (mavundu), (mavilla), (mavum), (mavanu), (mavinu), (mavalle) , (maville) etc. The same word can come with two suffixes as in the words (mavilninnu), (mavillengil), (mavayathinal), (mavayirikkum), (mavanullathu), (mavupayogichu), (mavileykkanu), (mavundengil), (mavanithu) , (mavilekkayi), (mavillathayi), (mavilottu), (mavukalum) ,(mavinoduchernnu) etc. The word mavu can come with three affixes as the cases (mavilninnukondu), (mavukalumayi), (mavukalumanu) etc and also with four affixes as in the following cases (mavundayirunnappol), (mavilekkayirikkum). Similar inflections happens in the case of verbs also.For example the verb (varuka) can take one affix as in (vannal) , (vannilla), (vannittu) etc . The verb can take two affixes as in the case (vannennal), (vannittundu), (vannittanu), (vannittilla) etc. Some of the morphological forms of the mentioned word are:

Malayalam word	Pronunciation in English	Meaning
മാവിൽ	Maavil	On the mango tree
മാവിൽനിന്ന്	Maavininn	From mango tree
മാവിൽനിന്നു	Maavilninnu	From mango tree
മാവിൽനിന്നാണ്	Maavilninnanu	It is from mango tree
മാവിൽനിന്നുകൊണ്ട്	Maavilninnukondu	Standing on mango tree
മാവുള്ള	Maavulla	Has mango tree
മാവിന്റെ	Maavinte	Mango tree's
മാവിനായി	Maavinayi	For mango tree
മാവോ	Maavo	Is that a mango tree?
മാവായ	Maavaya	Poetic version of saying mango tree
മാവാക്കിയ	Maavakiya	Made from mango tree
മാവുണ്ട്	Maavund	Has mango tree
മാവില്ല	Maavilla	Has no mango tree
മാവില്ലെങ്കിൽ	Maavilengil	If you don't have a mango tree
മാവുണ്ടായിരുന്നു	Maavundayirunnu	Had mango tree
മാവിലെ	Maavile	On the mango tree
മാവിലേക്ക്	Maavilekk	Towards mango tree
മാവുപോലെ	Maavupole	Looks like mango tree
മാവിന്	Maavinu	For mango tree
മാവിലൂടെ	Maaviloode	Between mango trees
മാവാണിത്	Maavanith	This is mango tree
മാവില്ലതായി	Maavillathayi	Dead mango tree
മാവുകളും	Maavukalum	Mango trees (plural)
മാവല്ല	Maavalla	Not mango tree
മാവിലോട്ടു	Maavilottu	Towards mango tree
മാവിലോട്ടല്ല	Maavilottala	Not towards mango tree
മാവിലേക്കായി	Maavilekkayi	Towards mango tree
മാവുമായി	Maavumayi	With mango tree

5 Methodology

The proposed method has a list of root words, has the rules that causes inflections and then uses a suffix stripping algorithm to get the stem. A stem does not have to be the word's morphological root. When all of a word's morphological variations conflate to a single stem after stemming, the stemmer's accuracy is high. Overstemming and understemming are the two most common stemmer problems. Overstemming occurs

when words other than morphological variants conflate. When morphological variants do not conflate, understemming occurs.

5.1 Inflections

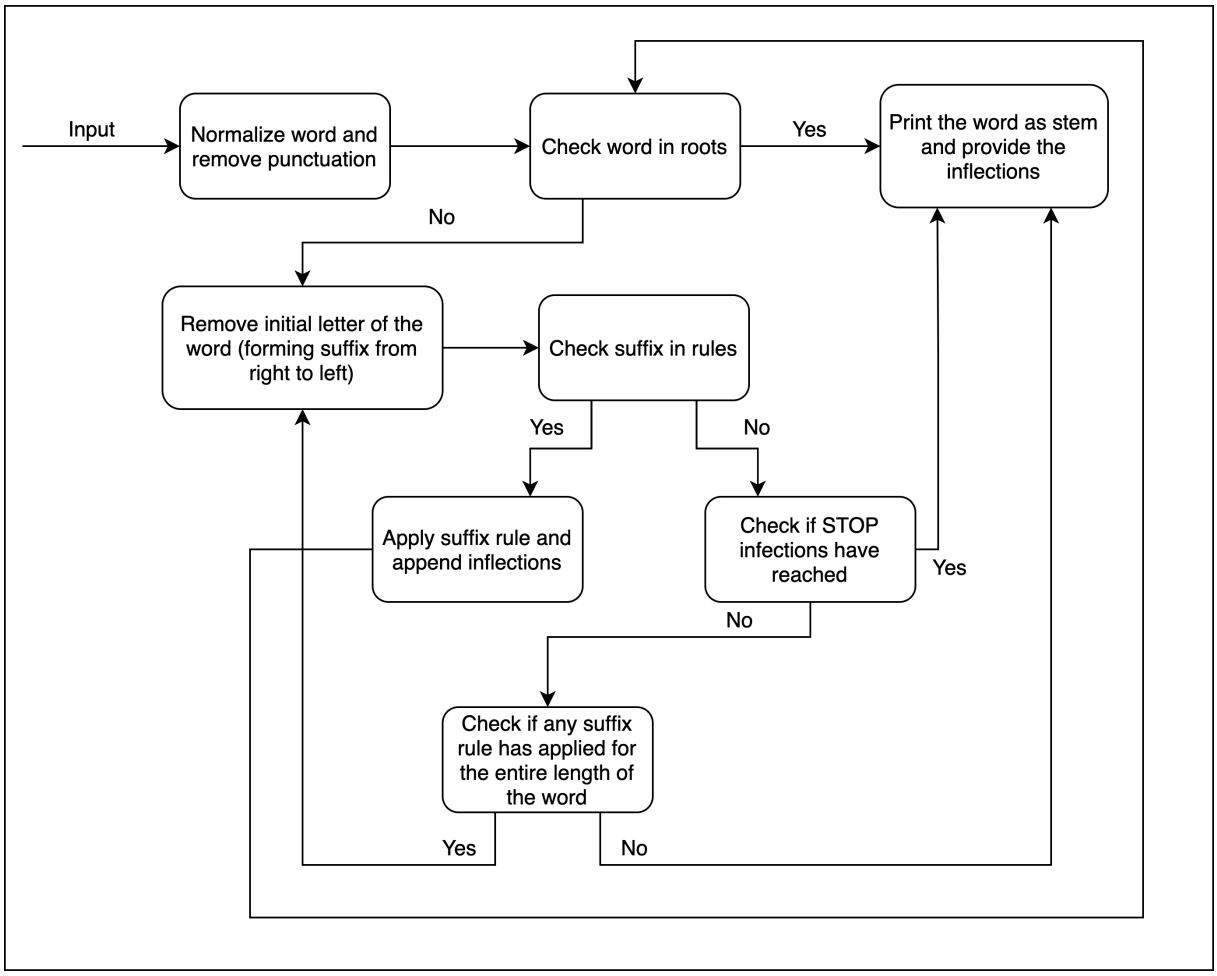
Case markers accompany noun groups in Malayalam. In Malayalam, there are nine case markers. Nominative, accusative, dative, genitive, sociative, locative, instrumental, ablative, and vocative are the different types of nouns. They're often inflected depending on who's speaking, how many people are speaking, and whether they're male or female. Verbs are grammatical units that have the ability to change tense (past, current, and future). Grammatical rules like prathigrahika, somyogika, udheshika, prayochika, sombandhika, aadharika, sambhodana, verbs, numerals, and some miscellaneous special rules are considered for stemming.

5.2 Suffix stripping algorithm

To improve the task's efficiency, inflections must be converted to a stem in information retrieval tasks. This is the job of a stemmer, and it can be accomplished with the help of a suffix stripping algorithm, which removes suffixes from inflections and conflates them into a single stem.

The algorithm presented in the paper takes word or sentences as input, fixes utf-8 encoding, removes punctuation and tokenizes the input. Each work in the list of tokens is looped and is checked if the word is present in bag of words. If the word is present in the bag of roots print the word itself as stem with empty inflections. If no root is found, the initial letter from the word will be removed forming a suffix and the suffix will be checked in for the grammatical suffix rules. If suffix matches, the rule will be applied and corresponding inflections will be appended. Again the word with the applied rule will be checked in the bag of roots and the loop continues until all the inflections are found and the stem is generated. If no rules are matched, the word will be checked in for some STOP rules. If any of the STOP rules get matched, then the word and inflections will be printed. There is also an mechanism which will check if any rules has been assigned to the entire length of the word. This is avoid the infinite looping situation. Words like

name of a person may not be in the bag of roots and there may be no rules to apply. So to consider such cases this check is introduced. The algorithm used is shown below:




```
def stemmer(text):
    words = text.split()
    roots = load_root()
    rules = load_rules()
    result_dict = dict()
    for word in words:
        inflections = []
        word_provided = deepcopy(word)
```

```
        word = normalize(word.translate(str.maketrans('', '',
                string.punctuation)))
        found = True
        while found:
            index = 1
            is_changed = False
            if word in roots:
                break
            while index < len(word):
                suffix = word[index:]
                if suffix in rules:
                    word = word[:index] + rules[suffix][0]
                    inflections.append(rules[suffix][1])
                    found = True
                    is_changed = True
                    break
                index += 1
                if inflections and "STOP" in inflections[-1]:
                    inflections.pop()
                    found = False
                    break
            if not is_changed:
                break
        result_dict[word_provided] = {"stem": word,
                        "inflections": inflections}
    return result_dict
```

6 Acknowledgement

I sincerely express my gratitude to University of Maryland, Baltimore County for providing me with the opportunity to work on the topic "Malayalam Stemmer". I also thank Dr. Claudia Pearce for her expert supervision, constructive criticism, and helping me complete the paper work as a part of coursework.

References

[1] Marçal Rusiñol , Volkmar Frinken , Dimosthenis Karatzas , Andrew D. Bagdanov, Josep Lladós, "Multimodal page classification in administrative document image streams", International Journal of Document Analysis and Recognition, IJDAR 17: 2014.pp. 331–341

[2] Hongxi Wei · Guanglai Gao, "A keyword retrieval system for historical Mongolian document images", International Journal of Document Analysis and Recognition , IJDAR, 2014.Vol 17:pp.33–45.

[3] Arwa Alqudsi, Nazlia Omar, Khalid Shaker,"Arabic Machine Translation:A survey" , Artificial Intelligence Review, 2014. Vol. 42, pp. 549-572.

[4] Elaheh Asghari , MohammadReza KeyvanPour, "XML document clustering: techniques and challenges", Artificial Intelligence Review , 2015. Vol. 43, pp.417-436.

[5] Nagwani N K, "Summarizing large text collection using topic modeling and clustering based on MapReduce framework", Nagwani Journal of Big Data 2:6, 2015.

[6] Gheith A, Abandah, Alex Graves , Balkees Al-Shagoor, "Automatic diacritization of Arabic text using recurrent neural networks", International Journal on Document Analysis and Recognition, 2015, Vol 18, pp. 183-197.

[7] Osama Mohamed Elrajubi,"An Improved Arabic Light Stemmer" , Third International Conference on Research and Innovation in Information Systems, 2013, pp. 33-38.

[8] Ayu Purwarianti, “A Non Deterministic Indonesian Stemmer”, International Conference on Electrical Engineering and Informatics,2011.

[9] Cristian Moral, Angelica de Antonio, Ricardo Imbert, “A survey of stemming algorithms in Information Retrieval”, Information Research, 2014, Vol.19, No.1.

[10] Walid Cherif , Abdellah Madani, Mohamed Kissi, “Integrated effective rules to Improve Arabic Text Stemming”, IEEE , 2014. pp. 1077 – 1081.

[11] Vishal Gupta,Gurpreet Singh Lehal , “A survey of Common Stemming Techniques and Existing Stemmers for Indian Languages”, Journal of Emerging Technologies in WebIntelligence ,2013, Vol 5, No. 2 , 157-161.

[12] Jikitsha Sheth, Bankim Patel, Dhiya : “A Stemmer for morphological level analysis of Gujarati language” , International Conference on Issues and Challenges in Intelligent Computing Techniques,2014,pp. 151-154.

[13] Redowan Mahmud Md, Mahbuba Afrin, Md. Abdur Razzaque Ellis Miller, Joel Iwashige, “A rule based Bengali stemmer”, ICACCI,2014, pp.2750 – 2756 .

[14] Vasudevan N, Pushpak Bhattacharya, Little by little Semi Supervised Stemming through Stem Set Minimization, ijcnlp , 2013.

[15] Vasudevan N.,Pushpak Bhattacharyya, Optimal Stem Identification in Presence of Suffix List, Computational linguistic and Intelligent Text Processing , 13th International Conference CICLing, ed . Alexander Gelbukh, New Delhi, Springer Berlin Heidelbeg, ,2012, pp.92- 103 .

[16] Rohit Kansal, Vishal Goyal and Lehal G.S, Rule Based Urdu Stemmer, Proceedings of COLING,2012, pp. 267 - 276.

[17] Vijay Sundar Ram and Sobha Lalitha Devi ,Malayalam Stemmer, Morphological Analysers and Generators, ed. Mona Parakh, LDC-IL, Mysore, 2010,pp. 105 –113.

[18] Vinod P.M, jayan V, Bhadran V.K, Implementation of Malayalam Morphological Analyzer Based on Hybrid Approach, Proceedings of the Twenty-Fourth Conference on Computational Linguistics and Speech Processing, ROCLING, 2012, pp. 307 – 317.

[19] Jisha P. jayan , Rajeev R.R, Dr S. Rajendran, Morphological Analyser and Morphological Generator for Malayalam - Tamil Machine Translation, International Journal of Computer Applications,2011, Vol.13-No.8, pp 15 - 18.

[20] Prejitha U, Sreejith C, Reghu Raj P.C,Lalitha : A light weight Malayalam Stemmer using Suffix Stripping Method, International Conference on Control Communication and Computing (ICCC),2013,pp. 244 – 248 .