MORPHOLOGICAL ANALYSIS OF MANIPRAVALAM

A PROJECT REPORT

Submitted by

KRISHNARAJ N - 2015115083
MARUTHAMANI M - 2015115087
PRAVIN VIGNESH SK - 2015115100

Submitted to the Faculty of

INFORMATION AND COMMUNICATION ENGINEERING

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

INFORMATION TECHNOLOGY

INFORMATION SCIENCE AND TECHNOLOGY

CEG CAMPUS

ANNA UNIVERSITY

CHENNAI 600 025

APRIL 2019

ANNA UNIVERSITY CHENNAI

CHENNAI - 600 025

BONAFIDE CERTIFICATE

Certified that this project report titled MORPHOLOGICAL ANALYSIS OF MANIPRAVALAM is the bonafide work of KRISHNARAJ N (2015115083), MARUTHAMANI M (2015115087), PRAVINVIGNESH S.K (2015115100) who carried out the project work under my supervision. Certified further that to the best of my knowledge and belief, the work reported herein does not form part of any other thesis or dissertation on the basis of which a degree or an award was conferred on an earlier occasion on this or any other candidate.

PLACE: **Dr.RANJANI PARTHASARATHI**

CHENNAI **PROFESSOR**

DATE **PROJECT GUIDE**

COUNTERSIGNED

Dr. SASWATI MUKHERJEE

HEAD OF THE DEPARTMENT

DEPARTMENT OF INFORMATION SCIENCE AND TECHNOLOGY

COLLEGE OF ENGINEERING, GUINDY

ANNA UNIVERSITY

CHENNAI 600025

ABSTRACT

Many traditional documents are written in Manipravalam language, which is a combination of Tamil and Sanskrit. Understanding such text is difficult. Hence there is a need to translate Manipravalam to Tamil. Tamil and Manipravalam are closely related lanuages with almost similar grammar rules. But Manipravalam has some rules which are not part of Tamil Grammar Rules. Morphological analysis is the first step in Machine translation. Morphological analyser for Tamil language is already available but this cannot be used for the Manipravalam Language. This project modifies an existing morphological analyser for Tamil language to work for Manipravalam language by adding rules and adding new root words to the dictionary. A semi-automatic mechanism to add new root words to the dictionary has been developed. Some of the Manipravalam words are in Sanskrit which cannot be split by the modified analyser. Hence those words are to be classified before giving input to the analyser as non-native words. The modified analyser has been tested with 4000 words and the accuracy is seen to increase from 43% to 76%.

ABSTRACT

ACKNOWLEDGEMENT

First and foremost, we would like to express our deep sense of gratitude to our guide Dr.RANJANI PARTHASARATHI, Professor, Department of Information Science and Technology, Anna University, for her excellent guidance, counsel, continuous support and patience. She has helped us to come up with this topic and guided us in the development of this project. She gave us the moral support and freedom to finish our final year project in a successful manner.

We express our sincere gratitude to Dr.SASWATI MUKHERJEE, Professor and Head, Department of Information Science and Technology, Anna University, for her kind support and for providing necessary facilities to carry out the work.

We are thankful to the final year project lab teachers Dr.E.UMA, Dr.M.VIJAYALAKSHMI, Ms.SINDHU, Department of Information Science and Technology, Anna University, Chennai, for their valuable guidance and technical support.

We also thank the faculty and non-teaching staff members of the Information Science and Technnology, CEG Campus, Anna University, Chennai-25, for their valuable support throughout the course of my research work.

MARUTHAMANI M
KRISHNARAJ N
PRAVIN VIGNESH S K

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

Tamil Morphology is very rich. It is an agglutinative language, like the other Dravidian languages. Tamil words are made up of lexical roots followed by one or more affixes. Many old classic documents were written in Manipravalam language. Manipravalam language is a combination of Tamil and Sanskrit. Although Manipravalam language rules are similar to Tamil language rules, Tamil language processing tools cannot be applied on Manipravalam texts. Since many Sanskrit words will be in Manipravalam documents, there is a need of translation system to facilitate translation of Manipravalam texts to Tamil. Unfortunately, there is no Translation system available currently. The tools for the translation of Manipravalam to Tamil are required to be developed.

Morphological analysis is the first step in such translation. Morphological analyser for Tamil language is already available. Below are the examples for morphological analysis of Tamil and Manipravalam text using existing Tamil analyser.

1. Morphological analysis for Tamil text:

   (a) Veetilirundhu

   Veedu(noun) + il(Locative case) + irundhu(Postposition).

2. Morphological analysis for Manipravalam text:

   (a) Poramaiyaale

   Poramaiyaale - Analyser Not Found

In the first example analyser correctly splits the word and assigns tag. In the second example the analyser cannot split or assign tag for the word because Manipravalam texts has some new grammar rules which are not present in Tamil analyser. It alos lacks of Manipravalam root word dictionary.

In this project, we modify an existing morphological analyzer for Tamil text based on rule based approach to support Manipravalam text.

1.1 PROBLEM STATEMENT

Tamil and Manipravalam are closely related languages. Manipravalam grammar rules are similar to Tamil grammar except for some of the rules which are only part of Manipravalam language. Manipravalam has a lot of Sanskrit words which cannot be translated to Tamil. Those words need to be classified as non-native words.

The challenge of this project is that existing morphological analyser cannot be used as it is for Manipravalam texts. The reason is lack of Manipravalam dictionary and missing grammar rules in the analyser. Hence, dictionary creator tool to add Manipravalam root words to the dictionary has to be developed. Also, new rules have to be added to the analyser.

1.2 OBJECTIVES

The objectives of this project are:

- Identify words that can be split and cannot be split by the analyser,
- Add new rules for the suffixes of Manipravalam language,

- Develop a dictionary creator to add missing root words of Manipravalam language.

1.3 ORGANISATION OF THE REPORT

The project report is organised as follows:

Chapter 2 discusses the work of existing systems and various methods required for the proposed system.

Chapter 3 discusses the working of various modules of the proposed system along with the overall system architecture.

Chapter 4 discusses the implementation detail of the proposed system along with the necessary algorithms and illustrates the experimental results of the proposed work.

Chapter 5 concludes the report and proposes possible enhancements that can be done in future.

CHAPTER 2

LITERATURE SURVEY

This chapter discusses the work related to morphological analysis and text classification using SVM.

2.1 EXISTING WORK

2.1.1 Morphological Analysis

Morphology is the identification, analysis, and description of the structure of a given language's morphemes and other linguistic units, such as root words, affixes, parts of speech, intonations and stresses, or implied context. Morphological analyzer is a tool which takes words as input and produces its grammatical structure as output. It tries to identify and segment the words and assigns the grammatical information. Since this is a tedious and challenging job, many researchers have developed rules and rule based syntax for performing morphological analysis. Tamil morphological analyzers were originally built by Aanandhan et al., [1] in 2002 and then by AU-KBC Research Centre in 2003 [2]. Since then research on Tamil morphological analysis has continued in two directions, using machine learning and using rule based approaches. Akilan and Naganathan [3] have developed morphological analyzer for classical Tamil texts using rule based approach. The rules developed by them are dependent on each other and it produces 72 % of accuracy for classical Tamil texts.

Selvam and Natarajan [4] 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 by Kumar et al., [5]. It was a supervised machine learning approach and a corpus with morphological information was used for training.

2.1.2 Text Classification using SVM

Joachims [6] explains the use of Support Vector Machines (SVMs) for learning text classifiers. It analyzes the particular properties of learning with text data and identifies why SVMs are appropriate for this task. This paper explores and identifies the benefits of Support Vector Machines (SVMs) for text categorization. Moreover, in contrast to conventional text classification methods SVMs prove to be very robust, eliminating the need for expensive parameter tuning. This has been proven by considering theoretical and experimental results. The experiment compare the performance of SVMs using polynomial and RBF kernels with four conventional learning methods commonly used for text categorization. The test collection is taken from the Ohsumed corpus compiled by Hersh. From the 50216 documents in 1991 which have abstracts, the first 10000 are used for training and the second 10000 are used for testing. The classification task considered here is to assign the documents to one or multiple categories of the 23 MeSH diseases categories. The results show that SVM works fine and better than the conventional learning methods. Empirical results support the theoretical findings. SVMs achieve substantial improvements over the currently best performing methods and behave robustly over a variety of different learning tasks. Mohamed and Shanmugasundaram [7] explain an approach to cluster words in document containing Tamil words. This paper uses vector space model to cluster the documents. Vector space model is otherwise known as Term-frequency approach. Stop words which are frequent,

meaningless terms are removed from the input text document to decrease the size of the document to be processed. Then the cosine similarity measure is applied to find the similarity between the input text documents. Then clustering is done using K-Medoid algorithm and optimal number of medoids and corresponding clusters are found. Rani and Satvika [8] proposed Text Categorization on Multiple Languages Based On Classification Technique. The objective of the work is the representation and categorization of Indian language text documents using text mining techniques, such as Support Vector Machine, KNN (KNearest Neighbor), and Decision Tree.

CHAPTER 3

SYSTEM DESIGN

This chapter gives the design details of this project.

3.1 OVERALL DESIGN AND DESCRIPTION

Figure 3.1 gives the overall system design of this project. This project implements three modules, namely:

1. Text Classifier,
2. Morphological Analyser, and
3. Dictionary Creator.

Manipravalam document is given as input to the classifier. It separates the words into two classes, native(Tamil and Manipravalam words) and non-native(Sanskrit or Sanskrit sounding words). The Tamil words are given as input to the morphological analyser. The analyser splits those words which has root words found in the dictionary. For the missing root words, dictionary creator is used to add those to the dictionary.

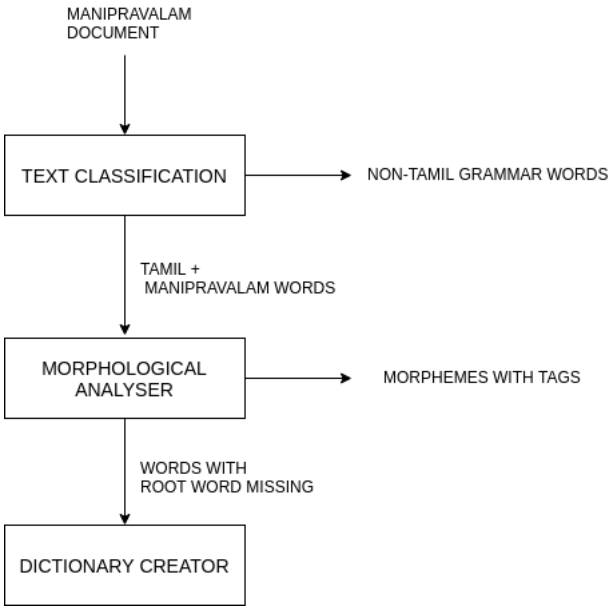

Figure 3.1: Overall System Design

3.2 TEXT CLASSIFIER

Text classification is an automated process of classification of text into predefined categories. This can be done with the help of Natural Language Processing and different Classification Algorithms like Naive Bayes, SVM, and Neural Networks. In this project, text classifier consists of two phases namely training and testing. The training dataset used for classification has two classes labelled as 0 and 1. Label 1 indicates words which can be split and processed by analyser which includes Tamil words and Manipravalam words and label 0 indicates words which cannot be processed by morphological analyser. The words in the training dataset are vectorized using character frequency - inverse term frequency (CF-ITF) vectorization. CF-ITF is used to convert a collection of raw terms to a matrix of CF-ITF features. CF-ITF features are numerical values. The vectorized values and labels are trained using SVM linear kernel to devlop a SVM predictive model. In testing phase, Manipravalam text document is taken as input and preprocessed to remove spaces and commas. The preprocessed data (manipravalam words) are fed into the SVM predictive model to classify

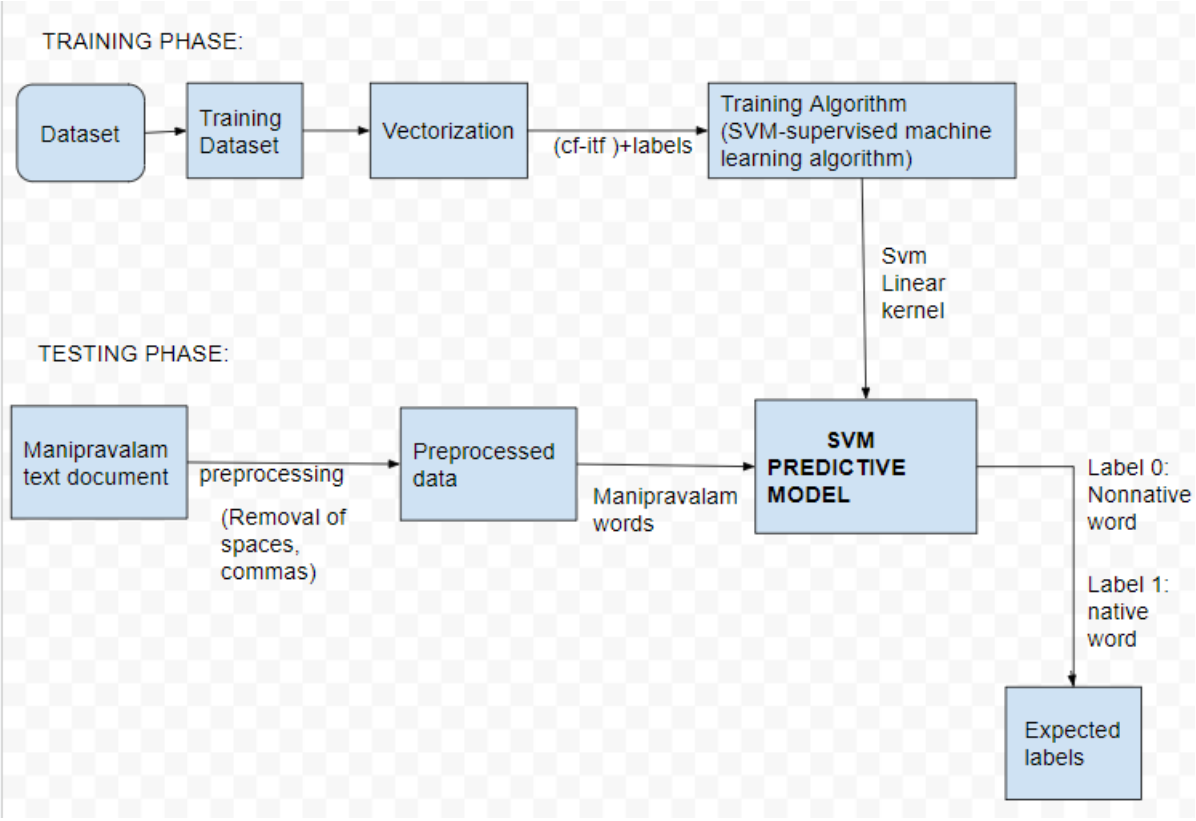

Figure 3.2: Classifier

the words which are native and non-native. Figure 3.2 depicts the flow of the classifier.

3.3 MORPHOLOGICAL ANALYSIS

Morphological analysis of a word is the process of segmenting the word into component morphemes and assigning the correct morpho syntactic information. For a given word, a morphological analyzer will return its root word and the word class.

3.3.1 Existing Morphological Analyzer for Tamil

Tamil is a morphologically rich language in which most of the morphemes coordinate with the root words in the form of suffixes.

PostPosition	அற்க், படும், ஆல்தான், வைத்து, உடைய, கொண்டு, எதிரே, பதிலாக, படி, பற்றிய, மூலமாக ,அன்படி,இன்படி,ஆததால், அதால், மாறு
Clitics	என்னும், ஆகிலும், ஆயினும், ஆவது, அம்மா, அப்பா, அய்யா
Suffix	அலாம், கலாம், இடம், உக்கு, உக்க், இற்கு, இற்காக, அற்காக, அற்கு, அதற்கு, அக்கு , க்கு இலிருந்து,இடமிருந்து,இலிருந்த்,இடமிருந்த், உடைய, அற்று,
Tense	கிற், கிறு, க்கிறு, கின்ற், கின்று, க்கின்று, க்கிற், க்கின்ற்
Sandhi	ஆண், தின், உண், பூண், த்து, ப், வ், ந், இன், அன், ற்று,ஓடு, ழ், ஓடு, ஆள், ஆன், ஆர், அடா, அடி, ஏ, ஆம்,கள், தான்,

Figure 3.3: Partial List of Suffixes

Morphological analyzer takes a derived word as input and separates its root word and associated morphemes. The rule based approach is used and the function of each suffix is indicated.

Example for Morphological analysis:

1. Veetilirundhu

   Veedu(noun) + il(Locative case) + irundhu(Postposition).

This is the text in first paragraph. This is the text in first paragraph. This is the text in first paragraph. Tamil nouns can take case suffixes after the plural marker. They can also have post positions after that. Tamil words consist of a lexical root to which one or more affixes are attached. Most Tamil affixes are suffixes. Tamil suffixes can be derivational suffixes, which either changes the part of speech of the word or its meaning, or inflectional suffixes, which mark

categories such as person, number, mood, tense, etc. The words can be analyzed like the one above by identifying the constituent morphemes and their features can be identified. Figure 3.3 gives a partial list of suffixes and their functions.

The modules involved in morphological analysis are:

- Tamil unicode converter
- Number check
- Dictionary
- Check Prefix
- Rules
- TagSet

Tamil unicode converter will help to convert Tamil ASCII text to convert Tamil UNICODE text. Number check module checks whether the entered text is number or not. Dictionary module initialises dictionary for verb, Noun, Adjective, Adverb, Particle, Negative finite Verb, conjuntion, Interjection, Interrogative, Adjective, Finite Verb, Postposition, Intensifier and also load dictionary in memory. checkIsRoot module checks whether the entered string is a Root word. checkPrefix module is implemented only when the checkIsRoot has failed. It checks for prefix in the entered string. Rules is the module where the rules for the Language Grammar are written. checkPrefix module has sub-modules like Noun, Verb, Adjective, Adverb, Paritcle, Sandhi, Tense. These sub-modules are used to identify the correct morpho syntatic information.

3.3.1.1 Unicode Generator

For a given word, it converts the Tamil word to its unicode value. For example, Thiruvaimoziyil - Thiruvaimozi + ei + il.

The values are 2980- thi, 2992- ru, 2997- vaa, 2991- ei, 2990- mo, 2996- zi.

3.3.1.2 Check Numbers

For a given input string it should check whether it is a number or date or time. The Date Time tag includes the days, months in Tamil language and also months, days in English language.

If input string is netru then the Morphological Analyser should return netru. The dictionary contains the Tamil number words. They are nooru, naarpathu, naangu, moonru, irupathu, irandu, onru, pathu, irunooru, arupathu, etu, eezu, aayiram, ainthu, aimbathu, kodi, thonooru, laksham, ezunooru.

If the input string is from this set of words. It starightaway assigns the tag as 'numbers'.

3.3.1.3 Check Prefix

This module identifies the root word for the given string by utilising the Tag sets and dictionary. CheckPrefix will further call entity, pronoun, clitic, Negative finite verb, pronoun oblique, auxillary verb, particle, and intensifier, adverb.

3.3.1.4 Tag Set

The tag set consists of the following categories:

Noun, verb, adjective, adverb, conjunction, negative finite verb, pronoun, postposition, locative case , accusative case, interjection, intensifier, interrogative adjective, interrogative noun, non-tamil, finite verb, and particle.

3.3.2 Working of Manipravalam Morphological Analyzer

Algorithm 1 explains the flow of analyzer and the Figure 3.4 depicts the working of morphological analyser.

Algorithm 1 Working of analyzer

Input Manipravalam Text
Output Morphemes with Tags

1: It first converts Manipravalam text into UNICODE.
2: After UNICODE conversion, it checks for numbers in the text. If it is number then there is no need to analyse.
3: Then it checks for root word in the dictionary, if it is a root word then return.
4: If it is not a root word, then it pre-processes the text with some rule based conversion and again checks for root word.
5: If root word found, it assigns the tag and returns.

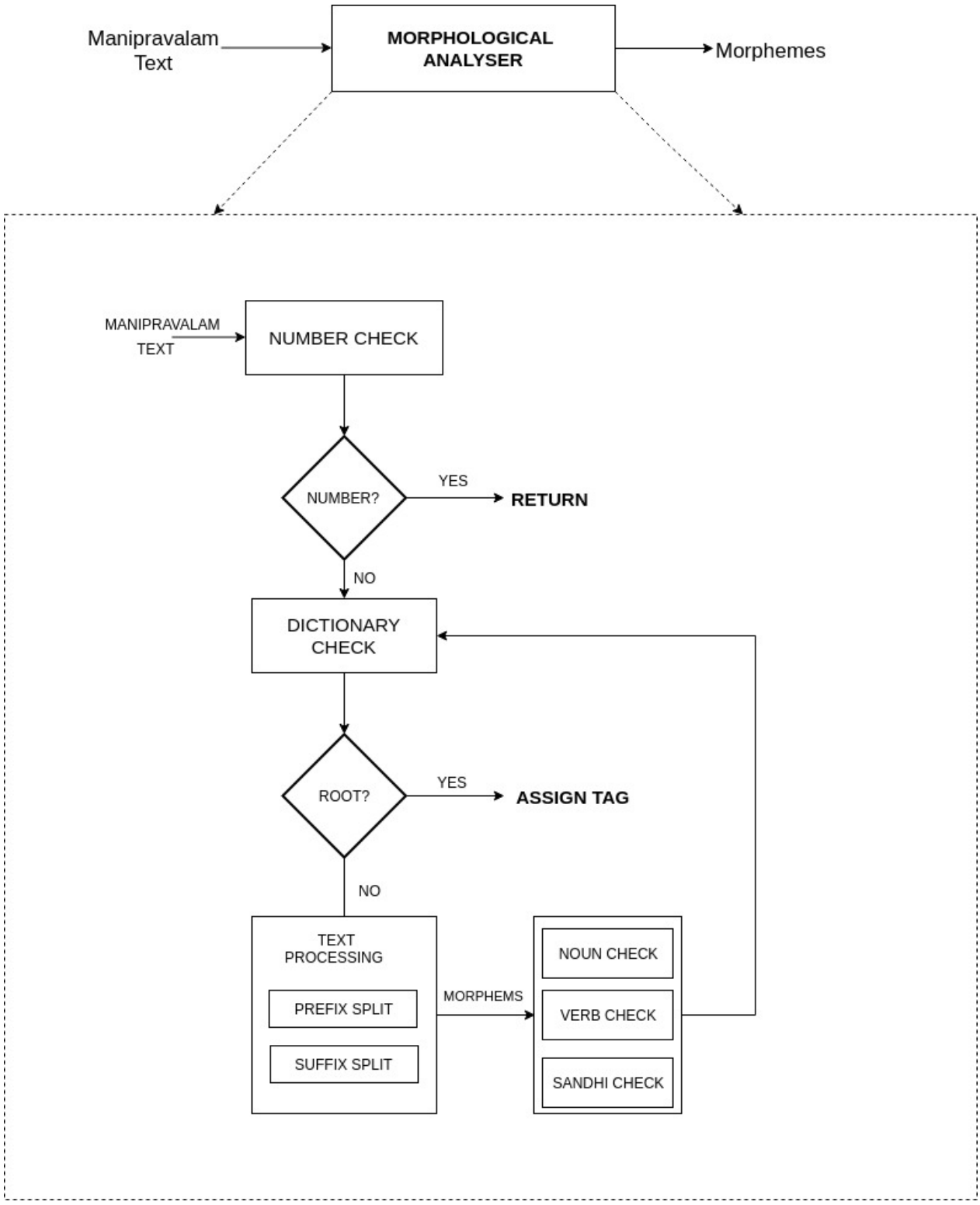

Figure 3.4: Working of Morphological Analyser

Example of morphological analysis for Manipravalam text:

1. Poraamaiyaale

   Poraamai (Verb) + e (Verbal participle suffix) + aale(Auxillary verb)

2. Sonnadhayirukkai

   Sol (Verb) + nn (Past tense maker) + a (Relative participle suffix) + th (Past tense maker) + aai (Second person singular suffix) + irukkai (Auxiliary verb)

3.3.3 Rules

Rules in the morphological analyser denotes the possible suffixes of a language. Existing morphological analyser for Tamil has all the suffix rules related to Tamil language which makes the analyser to split those suffix from the root words. Manipravalam language has some rules which are not part of Tamil. Those rules are found by testing the analyser with inputs of Manipravalam text. Rules are added to the analyser into the possible suffix category based on the output of testing.

Figure 3.5 has the list of newly added rules and their categories with examples.

3.4 DICTIONARY CREATOR

Dictionary creator module get the words from the output of analyser which are unanalysed as a input. It has an interface to edit the words before it is put into dictionary and buttons based on the dictionary files like verb, noun, adverb,and adjective. Using these buttons and the interface the unanalysed

RULE SUFFIX	CATEGORY	MANIPRAVALAM WORD
irukkai	locative case	sonnadhaayirukkai
pOlE	postposition	ennumappole
AlE	adjective	kidakkaiyale
vENdu	Verbal participle suffix	aatravendittru
enkiRa	adverb	vaasagamenkira
vENum	verb	pirakkavenumo
enRum	Pronominal third person femine singular	barathathvamendrum
AyiRRu	Finite verb suffix	amruthamaayitru
ittu	Auxiliary verb	ivattraiyittu
irukkum	Finite verb suffix	thoorasthanaayirukkum
thoRRu	Past tense marker	aagaranthottra

Figure 3.5: Rules and Their Categories

words are put in the appropriate dictionary. Figure 3.6 is the user interface of dictionary creator.

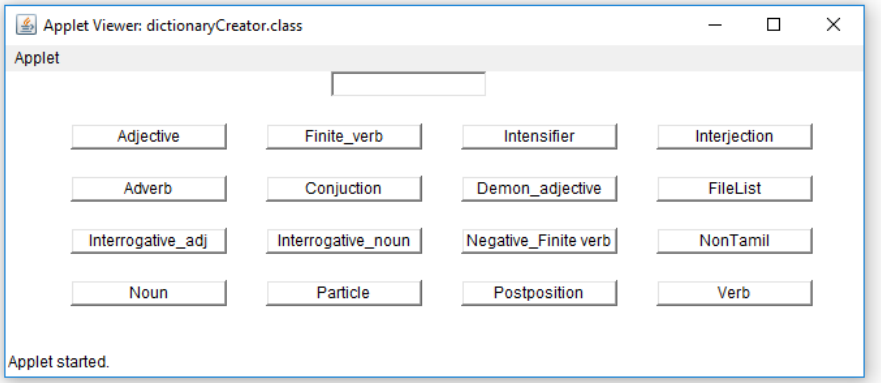

Figure 3.6: Dictionary Creator

CHAPTER 4

IMPLEMENTATION

This chapter deals with the modules and sub-modules involved in implementing the Morphological analysis on Manipravalam Text.

4.1 TOOLS AND LIBRARIES USED

This section has the list of tools and libraries used throughout this project.

4.1.1 Classification

- Tool : jupyter notebook
- Language : Python 3.0
- Libraries : numpy, pandas, sklearn

4.1.2 Analyser and Dictionary Creator

- Tool : Netbeans
- Language : Java
- Packages : Applet

4.2 CLASSIFICATION

Support vector machine classifier algorithm is used for text classification. It is a supervised learning approach. SVC linear kernel module is used to develop the classifier. It takes corpus of 3000 words. The main purpose of the classifier is to classify the words which can be split and cannot be split by the analyser. Pronunciations are used to classify the words and this is used as a feature in the model. Those words which can be split by the analyser are classified separately and given as the input to the analyser. The classifier cannot handle strings. So the strings should be vectorized. The vectorizer used is cf-itf(character frequency-inverse term frequency) Vectorizer. The Corpus will be split into two data sets, Training and Test. The training data set will be used to fit the model and the predictions will be performed on the test data set. This can be done through the test_train_split from the sklearn library.

4.3 MODIFICATIONS TO ANALYSER

4.3.1 Pre-processing

In pre-processing stage unwanted symbols and spaces were removed by the removeSymbols() in analyser.java file. Without pre-processing the analyser were not provide any results for many of the words. After pre-processing the analyser's result were improved.

4.3.2 Adding Root Words

When an Manipravalam text is given as an input to the existing analyzer it only separates the suffixes of the word as root words are not found in the dictionary. Some root words are added to the dictionary by analyzing the output to make it work for Manipravalam text. Morphological Analyser is applied on Manipravalam text and the Figure 4.1 gives the result before and after adding root words to the dictionary.

As many of the Manipravalam root words are missing in the existing Tamil dictionary, a semi-automatic system is developed to add those missing root words to the appropriate dictionary file in a semi-automatic way.

Manipravalam words	Root words	Before adding the root words	After adding the root words
1.உபயவிபூதியும்	உபயவிபூதி	த் + உ + இ + ய் + உம	உபயவிபூதி + ய் + உம்
2.கிஞ்சித்கரிக்க	கிஞ்சித்கரு	Analyser Not Found	கிஞ்சித்கரு + இ + க்க் + அ
3.ஸம்ச்லேஷத்தை	ஸ்அம்ச்லேஷம்	அத்து + ஐ	ஸ்அம்ச்லேஷம் + அத்து + ஐ
4.அனுஸந்தித்து	அனுஸ்அந்து	Analyser Not Found	அனுஸ்அந்து + இ + த்த் + உ
6.நிர்வஹிக்கும்படி	நிர்வஹ்	Analyser Not Found	நிர்வஹ் + இ + க்க் + உம் + படி
7.நிர்வ்ருதியென்று	நிர்வ்ருது	Analyser Not Found	நிர்வ்ருது + ய் + என்று
8.ஸுகமாய்	ஸ்உகம்	ய் + என்று	ஸ்உகம் + ஆய்
9.ஸுகிக்கிறார்	ஸ்உகி	ஆய்	ஸ்உகி + க்கிறு + ஆர்
10.அவிச்சேதத்தைப்	அவிச்சேதம்	Analyser Not Found	அவிச்சேதம் + அத்து + ஐ + ப்

Figure 4.1: Result Before and After Adding Root Words

4.3.3 Adding Grammar Rules

The existing Tamil morphological analyser will not separate the Manipravalam words. In this modified Morphological analyser new grammatical rules are added to support those Manipravalam words.To make the existing morphological analyser to work for Manipravalam language new rules to be added. These rules

were added in different files. First the Unicode value of the new rule to be added in the constant.java file. By adding this the analyser can get the Unicode value of the newly added rule. Then the code for splitting was added in the appropriate file for each rule based on the tag based on the tagset. The algorithm 2 explains the steps to add rules.

Algorithm 2 Steps to Add New Rules

Result: New Rules

1: Find the correct suffix to be added as a rule.
2: Add that suffix with the unicode value in the constant.java .
3: Find which tag is possible for that suffix.
4: After finding tag file add the suffix with to the result stack.
5: After adding to the result stack remove the suffix from the word.

The rules added in the files are listed below.

1.vENum

Initially the unicode value for vEnum is defined in Constant.java file . For example, the unicode value for vEnum is 27,8,19,5,23. The rule code snippet for vENum is written in PNG.java file. Likewise the same procedure is followed for adding the rules. Figure 4.2 gives the code snippet for vENum. Figure 4.3 gives the suffix and the files where it was added. After adding the rules, the manipravalam words which could not be handled by the existing morphological analyser is now handled. Figure 4.4 gives the result before and after adding the rules.

```
if (ByteMeth.endsWith(topElmt, Constant.venum))
{
        //clia.unl.unicode.utils.Utils.printOut(Analyser.print, x + "venum");
        s.pop();
        s.push(new Entry(Constant.venum, Tag.AuxiliaryVerb));
        topElmt = ByteMeth.subArray(topElmt, 0,
          topElmt.length - Constant.venum.length);
        s.push(new Entry(topElmt, -1, oldTopElmt));
        Sandhi.k(s);
        return true;
}
```

Figure 4.2: Code Sample for Rule (VENUM)

	RULES	RESPECTIVE FILES
2	irukkai	VerbalCase.java
3	pOlE	Postposition.java
4	AlE	VerbalCase.java
5	vENdu	VerbalParticiple.java
6	enkiRa	Tense.java
7	vENum	PNG.java
8	enRum	VerbalMisc.java
9	AyiRRu	Png.java
10	ittu	VerbalParticiple.java
11	irukkum	Png.java
12	thORRu	VerbalParticiple.java

Figure 4.3: Rules and Its Respective Files

Suffix	Word	Before adding the rule	After adding the rule
இருக்கை	சமைந்திருக்கை	Analyser Not Found	சமை+ந்த்+உ+இருக்கை
போலே	என்னுமாபோலே	Analyser Not Found	என்னும்+ஆ+போலே
ஆலே	வருகையாலே	Analyser Not Found	வருகை+ய்+ஆலே
வேண்டு	ஆற்றவேண்டிற்று	ட் + உ + இற்று	ஆற்ற+வேண்டு+ உ +இற்று
என்கிற	வாசகமென்கிற	ன் + கிறு + அ	வாசகம்+என்கிற
வேணும்	பிறக்கவேணுமோ	உம் + ஓ	பிறக்க+வேணும் +ஓ
என்றும்	பரதத்வமென்றும்	என்று + உம்	பரதத்வம் +என்றும்
ஆயிற்று	அம்ருதமாயிற்று	Analyser Not Found	அம்ருதம்+ஆயிற்று
இட்டு	இவற்றையிட்டு	Analyser Not Found	இவற்றை+ய்+இட்டு
இருக்கும்	விஷயமாயிருக்கும்	Analyser Not Found	விஷயம்+ஆய்+இருக்கும்
தோற்று	ஆகாரந்தோற்ற	ற் + உ + அ	ஆகார் +தோற்று +அ

Figure 4.4: Result Before and After Adding Rules

4.4 TESTING AND RESULTS

4.4.1 Classification

The Training Data will have 80% of the corpus and Test data will have the remaining 20%. Figure 4.5 lists the sample test data and their respective class lables.

- 10 cross fold validation accuracy: 0.88

s.no	words	label
1	இப்படி	1
2	பரஸ்வரூபம்	0
3	நிர்ணீதமாயிற்று	1
4	பரஸ்வரூப	0
5	ப்ரதிஸம்பந்தியான	0
6	ஸ்வஸ்வரூபமிருக்கும்படி	0
7	என்னென்னில்	1
8	உடன்மிசை	1
9	உரிரெனக்	1
10	கரந்தெங்கும்	1
11	பரந்து	1
12	என்று	1
13	சரீராத்ம	1
14	பாவத்தைத்	1

Figure 4.5: Sample Test Results

4.4.2 Analyser

After adding these rules the accuracy of the morphological analyser has improved. Around 4000 were tested. Figure 4.6 shows the accuracy of analyser before and after modifications.

STATE	ACCURACY
EXISTING MORPHOLOGICAL ANALYSER	43%
AFTER CLASSIFIER	51%
AFTER ADDING ROOT WORDS	64%
AFTER ADDING RULES	76%

Figure 4.6: Analyser Accuracy

CHAPTER 5

CONCLUSION AND FUTURE WORK

5.1 CONCLUSION

This documentation explains the working of the existing morphological analyser and the modifications done on that to support Manipravalam texts. The accuracy of the modified morphological analyser has improved and now it can handle 76% of the Manipravalam words with the help of newly added rules and root words.

5.2 FUTURE WORK

In future with the help of this morphological analyser's output and processes like Named entity recognition, POS tagging, Chunking, and a parallel tagged corpus can be created. Using this corpus, a machine learning model can be created to translate Manipravalam language texts into Tamil.

REFERENCES

1. P. Aanandhan, Ranjani Parthasarathi, T.V. Geetha, and K. Saravanan. Morphological analyzer for tamil. *International Conference on Natural language Processing*, 2:257–267, 2002.

2. Tamil morphological analyser. *au-kbc.org*, 2003.

3. R. Akilan and Naganathan. Morphological analyzer for classical tamil texts. *International Conference on Natural language Processing*, 3:437–447, 2014.

4. Selvam and Natarajan. Improvement of rule based morphological analysis and pos tagging in tamil language via projection and induction techniques. *International journal of computers*, 3:357–367, 2009.

5. M. Kumar, V. Dhanalakshmi, K.P. Soman, and Rajendran Sankaravelayuthan. A sequence labeling approach to morphological analyzer for tamil language. *International Journal on Computer Science and Engineering*, 2:345–349, 2010.

6. Thorsten Joachims. Text categorization with support vector machines: Learning with many relevant features. *International Journal on Computer Science and Engineering*, 4:137–142, 2009.

7. Syed Sabir Mohamed and Hariharan Shanmugasundaram. Experiments on document clustering in tamil language. *ARPN Journal of Engineering and Applied Sciences*, 13:125–134, 2018.

8. Kapila Rani and Satvika. Text categorization on multiple languages based on classification technique. *International Journal of Computer Science and Information Technologies*, 7:167–177, 2016.