A Shallow Parser for Malayalam

Thesis submitted in partial fulfillment
of the requirements for the degree of

MPhil
in
Computational Linguistics

by

Devadath V V
201224608
devadathv.v@research.iiit.ac.in

International Institute of Information Technology
Hyderabad - 500 032, INDIA
July 2016

International Institute of Information Technology
Hyderabad, India

CERTIFICATE

It is certified that the work contained in this thesis, titled "**A Shallow Parser for Malayalam**" by **Devadath V V**, has been carried out under my supervision and is not submitted elsewhere for a degree.

Date	Adviser: Prof. Dipti Mishra Sarma

To my parents

Acknowledgments

The first and foremost I would like to thank my advisor Prof. Dipti Misra Sharma who provided me her generous support and inspiration all through this work. It is her guidance and constructive suggestions which have moulded my work in the present shape. I express my gratitude to her.

Next, I would like to thank Sanal Vikram(HCU), for giving more informations about NLP and IIIT-Hyderababd, apart from the support and help for work in Malayalam given during my stay at IIIT-Hyderabad.

I would like to express heart felt gratitude to my first mentors, Sambhav Jain and Himanshu Sharma for teaching me Shell Scripting and Python patiently. Without them I would not have written a single line of code.

I would like to thank Naveen Sankaran for being the big brother who helped me in solving technical problems in no time.

Research is a joint effort. I would like to thank all my friends and lab mates especially Litton J Kurisinkel, Pruthwik Mishra and Vigneshwaran Muralidaran, without their help and support my thesis would not have been completed successfully.

Next I would like to thank Srinivas sir, Ram Babu sir, and Namrata Madam, Laxminarayan Sir for providing the supportive research environment in the lab.

I do not have any words to describe Praveen Krishnan who gave me all the support during my stay at IIIT-H. Thanks for being there as a big brother.

Finally I would like to thank my whole family for the full support given by them for my research.

Abstract

Malayalam is an agglutinative and morphologically rich language as any other Dravidian language. Computational processing of Dravidian languages is not trivial because two or more words can join to form a string of words with a morpho-phonemic change at the point of joining. This process known as "*Sandhi*", in turn complicates the individual word identification. The current work is an attempt to break the barrier of word segmentation and to create a shallow parser for Malayalam, which facilitates non-recursive phrase identification given an input.

In this work, Shallow Parser has mainly 3 modules namely Sandhi Splitter, POS Tagger and Chunker. Since words are the basic components in the sentence, not identifying the individual words in a sentence will affect the output of a shallow parser. Hence in order to tackle the problem of "*Sandhi*", after attempting a few rule-based approaches, we arrived at a hybrid "*Sandhi Splitter*" which gave an overall accuracy of 87% . This system uses Naive Bayes classifier to identify the split point and hand crafted character-level rules to induce morpho-phonemic changes. A CRF based Parts-Of-Speech tagger has been employed for the identification of grammatical category of words. Various experiments with different templates of features showed that Malayalam has more dependence over word-internal features like prefix and suffix information than word-external features like the position of a word in a sentence. The highest overall accuracy we obtained is 91.25%. The final module "chunker" has been employed to find out the non-recursive phrases based on the Parts-Of-Speech tags of the words. This CRF based chunker gave an overall accuracy of 94.33%.

Error propagation is a problem in shallow parsing. Errors created by each module affect the subsequent modules. When each module is put in a pipeline in such a way that the output of the previous module will be the input of the next module, overall accuracy of shallow parser came down to 71% from 92%. This shows the need and importance of a highly accurate sandhi splitter which is the main source of error. It is important to note that the experiments clearly show that morphology is the key factor for processing Malayalam.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

In this digital era, huge amount of textual data is being produced in various Indian languages in the form of online newspapers, social media posts, blogs etc. This has increased the necessity as well as the opportunity to break the language barrier by creating automatic language processing systems like Machine translation and Question Answering systems. This is being done with the help of Machine Learning and NLP algorithms which need huge amount of data. However different languages possess different complexities in terms of processing them computationally.

Words are the main components of a piece of text. All text processing tasks require individual words in the text to be identified. Sanskrit and Dravidian languages are highly agglutinative in nature. In these languages two or more words can join to form a string of words with euphonic change at the point of joining. which in turn complicates the individual word identification.

Malayalam is an agglutinative and morphologically rich Dravidian language. This work is to create a shallow parser for Malayalam, which facilitates non-recursive phrase identification given a raw sentence.

1.2 Syntactic parsing

Syntactic parsing is an important task in Natural Language Processing. It acts as an intermediate stage which gives the structure of sentences that can be used to process semantic, discourse and pragmatic information. Parsing a sentence is a complex task since it needs informations from various levels of analysis like, morphology, POS tags, chunks, semantics etc.. There are two challenges involved in full syntactic parsing.

1. In full parsing, a grammar and search strategy are applied to assign a complete syntactic structure to sentences. The main problem here is to select the most plausible syntactic analysis given the often thousands of possible analyses a typical parser with a grammar may return [22]. This increases the search space and increases the processing complexity.

2. Dependence over various modules causes increase in error propagation which eventually affects the final output.

On the other hand many NLP applications do not need that much information given by a full parser. As a response to these problems in full parsing, Abney introduced the concept of parsing by chunks [1] which gives a shallow syntactic analysis of a sentence that is relatively unambiguous.

1.3 Shallow parsing

Shallow parsing is a task of automatic identification of non-recursive phrases. Shallow parsing is a comparatively less complex process which uses only simpler search space. For recovering the shallow syntactic structure, Abney introduced the concept of parsing by chunks. According to him, a chunk consists of a single content word surrounded by a constellation of function words. To be precise, chunks are non-recursive phrases which contain only the head or content word and its modifiers.

Sentence → He sat on his suitcase.
Chunks → [He]NP [sat]VP [on]PP [his suitcase]NP.

In computational terms, for Abney, chunks are connected sub graphs of a sentences parse tree. They are defined in terms of a content word and their own syntactic structure that can be represented in the form of a tree. However a chunk does not include all the descendants of the root node that may be present in the parse-tree of the complete sentence.

Shallow Parser is not a single module, but a collection of modules which helps in extracting the shallow syntactic information from the input sentence. It has been experimentally proved that shallow parsers are useful in both text and speech processing domains. Verbmobil is a large project on developing speech to speech translation system. Shallow parsers have been used to add the robustness to the system for parsing the ill-formed spontaneous speech [58]. It is proven that the use of Shallow parsers can reduce the search space in full parsing, by identifying the head word and its modifiers [9]. Question answering system on World Wide Web is another place where the shallow parser has been used to process ill-formed data in large quantities [8]. This system uses a tool "SHAPQA"- Shallow parser for Question answering to retrieve the urls with the searched phrase keys.

1.4 Shallow parser for Malayalam

Making a Shallow parser for Malayalam is a challenging task because of its high agglutinative nature. One or more words can join to form a string of words with euphonic change at the point of joining. This process is known as Sandhi. This makes the individual word identification difficult and hence a Part-Of-Speech tagger and also a Chunker. The current work makes an attempt to build a shallow parser

for Malayalam which includes Sandhi Splitter, POS Tagger and a Chunker. Sandhi Splitter identifies and separates the individual words present in a string. building a POS tagger identifies the grammatical category of words and a chunker identifies the non-recursive phrases. Architecture of the Shallow parser pipeline for Malayalam in the current study has been given in **Figure 1.1**.

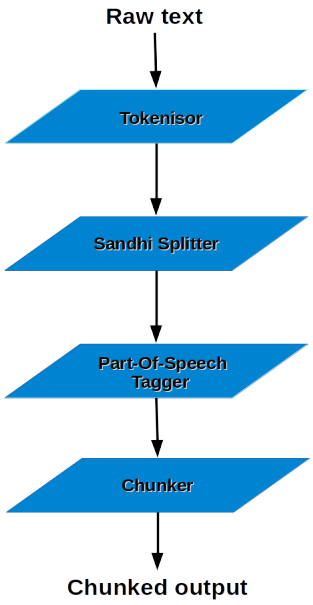


Figure 1.1: Pipeline Architecture

1.5 Summary of contributions

Highly Agglutinative nature of Malayalam makes the task tough to start with POS Tagging or Morph analysis since the words are joined to form a long string. This motivated to build a Sandhi-splitter which eventually helped in developing a Shallow Parser.

- A Hybrid Sandhi Splitter with 87% accuracy has been built using Naive Bayes and character level rules. This system precisely identifies the split points from a given string and applies character level rules to split and retrieve the actual surface form of a word. Statistical part is trained using a self annotated corpus of 2000 words with Sandhi.
- A statistical Part-Of-Speech Tagger has been built using Conditional Random Fields which gives an accuracy of 91%. From the experiments it is understood that the word-internal features along with word-level contexts give better accuracy than using either of them alone.

- A statistical Chunker also has been built using Conditional Random Fields which yields an accuracy of 94%. Experiment shows that the Chunker gives a better accuracy when it uses words and their POS Tags of a window of 2.
- An end to end Shallow parser with CRF has been built which shows the importance and need of a Sandhi Splitter with a high accuracy.
- Created a 70k POS and Chunk annotated corpus which is tokenised and Sandhi split.
- Created Sandhi annotated 2k data which can be used for creating and modifying Sandhi Splitter.

1.6 Chapterization

- **Chapter-2 Malayalam** Gives a detailed description about the grammatical structure of the language Malayalam with examples.
- **Chapter-3 Sandhi Splitter** Describes about the theoretical aspects of Sandhi followed by the explanation about the necessity and importance of Sandhi splitter . Then it describes the various methods of Sandhi splitting.
- **Chapter-4 Part-Of-Speech Tagger** Gives an account for the changes in definition of various tags. In continuation, the chapter explains the various experiments conducted with CRF followed by a detailed error analysis.
- **Chapter-5 Chunker** Details about the concept of chunk and chunking along with the previous works in general and particularly in Indian languages. This also gives a detailed account of experiments conducted using CRF along with a detailed error analysis.
- **Chapter-6 Shallow Parser** This chapter discusses about the pipeline of Malayalam Shallow Parser where the output of the previous module will be the input for next module. Experiments and results have been explained for both individual modules as well as pipeline.

Chapter 2

Malayalam

2.1 Malayalam

Malayalam is the official language of Kerala and Lakshadweep islands. It is one of the 22 official languages of India, spoken by around 35 million people[30]. This language belongs to the family of Dravidian languages and is believed to be originated from old Tamil having a strong influence of Sanskrit in its vocabulary at its later time of development[57]. Recently, Malayalam got the status of a classical language[60]. The language has a script of its own and now there are 51 letters including 20 long and short vowels and the consonants.

2.2 Characteristics of the language

Malayalam is an inflectionally rich and agglutinative language like any other Dravidian language such as Tamil, Telugu or Kannada. Even though all Dravidian languages share similar structural properties, there are two major characteristics which make Malayalam unique among Dravidian languages.

- Malayalam does not have predicate agreement between subject/object and verb for gender, number and person.
- Prevalent usage of copulas in utterances.

2.3 Grammatical categories

Traditional as well as modern grammars in Malayalam discuss about the grammatical categories of words. *Kerala Panineeyam* [57] by A R Raja Raja Varma which is the most widely accepted traditional grammar for Malayalam talks about nouns, pronouns, verbs, adjectives, adverbs, indeclinables, and conjunctions. In modern grammar *Malayalam*[32] by Asher and Kumari and *A Grammar for Malayalam*[37] by Ravi Sankar S Nair also describes all the general 8 categories of word, nouns, pronouns,

verbs, adjectives, adverbs, postpositions, conjunctions, and interjections. In this work, grammatical categories are explained based on the modern grammars.

2.3.1 Nouns

Nouns in Malayalam inflect for Gender, Number and case. Gender inflection is only for human beings and for non-human beings a common gender is assigned . “anZ”[1], “i”, “aM” are the masculine, feminine and neuter gender suffix markers respectively[57]. However from Pronouns “anZ”, “alZ”, “wu” are the respective masculine, feminine and neuter gender suffix markers. Animate human nouns take the suffix “maar” when they have gender suffixes and animate non-human nouns take the suffix “kal” [37]. Cases are and case suffixes are given in Table 2.2.

Gender suffixes		Examples	
Gender	Suffix	Example1	Example2
Masculine	anZ	mitukkanZ	kalYlYanZ
Feminine	i	mitukki	kalYlYi
Neuter	aM	veVlYlYaM	kalYlYaM

Table 2.1: Gender Inflection

Case suffixes		Examples			
Case	Suffix	Tree	Elephant	Ram	Kid
Nominatiive	NULL	maraM	Ana	rAmanZ	kutti
Accusative	eV	marawweV	AnayeV	rAmaneV	kuttiyeV
Socialtive	ot	marawwot	Anayot	rAmanot	kuttiyot
Instrumental	AlZ	marawwAlZ	AnayAlZ	rAmanAlZ	kuttiyAlZ
Dative	in/kk	marawwin	Anakk	rAman	kuttikk
Genetive	inrYeV/uteV	marawwinrYeV	AnayuteV	rAmanrYeV	kuttiyuteV
Locative	ilZ	marawwilZ	AnayilZ	rAmanilZ	kuttiyilZ

Table 2.2: Case-Inflections for Noun

2.3.1.1 Derivative Nouns

Derivative Nouns are the Nouns derived from Verbs. These Nouns can inflect for cases and they cannot take any argument as a verb[57]. “iruwwaM”, “kalYikkalZ”, “paTipp”, “ottaM”.

[1]Throughout this work, Malayalam characters are written using WX Transliteration scheme. Please see Appendix for the details.

(1) FAnZ paTipp wutarZnnu.
I study continue.PRST
'I continued the act of studying'

(2) eVnikk natawwaM iRtaM AN.
I.DAT walking like is.COP
'I like the act of walking'

2.3.2 Pronouns

Pronouns in Malayalam are similar to nouns in inflection and they inflect for Gender, Number, Person and Case. 1st Person singular Pronoun is "njAnZ". In nominative case, it uses this form and when it comes to all other cases, oblique forms are used[37]. The first person plural shows a distinction between inclusive and exclusive, where the inclusive being "nammalYZ"(speaker and addressee included) and exclusive being "FaffalYZ"(only speakers included)b[57]. 2nd Person singular pronoun is "nI" and plural form is "niffalYZ". The reflexive form "wAnZ" is also commonly used as second person singular form to address people with equal or lower status. 3rd Person Pronouns are Distal and Proximal pro-

	1st person			2nd person	
Case	Singlular	Plural-Exclusive	Plural-Inclusive	Singular	Plural
Nominative	FAnZ	FaffalYZ	nammalYZ	nI	niffalYZ
Accusative	eVnneV	FaffalYeV	nammalYeV	ninneV	niffalYeV
Sociative	eVnnot	FaffalYot	nammalYot	ninnot	niffalYot
Instrumental	eVnnAl	FaffalYAlZ	nammalYAlZ	ninnAlZ	niffalYAlZ
Dative	eVnikk	FaffalYkk	nammalYkk	ninakk	niffalYkk
Genetive	eVnrYeV	FaffalYuteV	nammalYuteV	ninrYeV	niffalYuteV
Locative	eVnnilZ	FaffalYilZ	nammalYilZ	ninnilZ	niffalYilZ

Table 2.3: Case inflections for P=1st and 2nd person Pronouns

nouns. They inflect for Gender, Number and cases as Nouns do. "*a*" is the distal marker and "*i*" is the proximity marker. From these morphemes, gender and number suffixes can be added to form words like

- "*aw*" = That thing.
- "*iw*" = This thing.
- "*avanZ*" = That male.
- "*ivanZ*" = This male.
- "*avalYZ*" = That female.
- "*ivalYZ*" = This female.

- "*avarYZ*" = That group of people.
- "*ivarYZ*" = This group of people.

"*e*" is the indefinite marker which is being used to create interrogative pronouns in Malayalam. Similar to distal and proximal markers, gender and number suffixes can be added.

- "*ew*" = which thing.
- "*eVvanZ*" = which male.
- "*eVvalYZ*" = which female.
- "*eVvarYZ*" = which group of people.

2.3.3 Postpositions

Postpositions in Malayalam are verbal participles that have lost the link to the verb from which they are derived [37]. Such forms are considered as grammaticalised and they cannot be considered as verbs any more. They serve to extend or modify the meaning of the case suffix or semantically link the noun to verbs.

(3) njANZ vatikoVNt aticcu.
I stick+instr beat.PAST
'I beat with/by stick.'

(4) eVnikk awineV kurYicc arYiyAM.
I.DAT that.ACC about know
'I know about that'

Postposition	Example	Meaning
koVNt	kall koVNt	by stone
Ayi	avan Ayi	for him
ninn	aviteV ninn	From there
oVppaM	enteV oVppaM	with me
pOleV	aw pOleV	like that
veNti	avalyZkk veNti	for her
kuricc	enneV kuricc	about me
prakAram	aw prakAram	according to that

Table 2.4: Postpositions

2.3.4 Verbs

Morphology of verbs in Malayalam is complex because of the rich agglutination. One of the most interesting properties of Malayalam shared by other Dravidian languages and south Asian languages in general is the problem posed by the conventional notion of finiteness[31]. Finiteness is usually related to some of the central properties of a clause such as tense, aspect, agreement and more generally on how a clause is anchored to the utterance context [3]. With the arrival of GB[2], finiteness was analyzed as a structural property of a clause rather than an inflectional property of a verb. Subsequently every clause was subject to a binary classification of whether it is finite or non-finite with various properties on verb such as tense, agreement signalling finiteness[31]. Therefore finiteness within the standard GB and minimalist framework is a morphosyntactic category that relates to

1. Regulation of tense and agreement.
2. The Control of realization of subject argument.
3. Restriction of some domains from application of certain syntactic rules.

This has been discussed in [39]. But these criteria of finiteness fail when we analyse the data in Malayalam. For one, Malayalam does not exhibit agreement as a feature on its finite verbs and interestingly just like every other Dravidian language, contains only one finite verb in a typical complex sentence with every other verbs being non-finite. Secondly, the multiple non-finite verbs show tense markers. Hence considering tense and agreement as criteria for deciding finiteness of a verb is not suitable for Malayalam. Other Dravidian languages show agreement as a feature in finite verbs but Malayalam lacks that as well and hence it is not possible to characterize finiteness in terms of agreement.

Generally "*unnu*", is the present tense marker, "*i/u*" are the past tense markers and "*uM*" is the future tense marker. "*pOkunnu*(go)" is considered as a finite verb since it can stand alone as a main verb in a sentence. If we analyse the morphology, "*pO*" is the verb root with the meaning "*to go* and "*unnu*" is the present tense marker. But if we look at the adjectival participial form of the verb root "*pO*", that is "*pOkunna*" which means "*that which goes*". This word modifies a noun, has the present tense marker "*unnu*" and cannot stand alone as a main verb. Similarly Past tense of "*pO*" is "*pOyi*", where "*i*" is past tense marker. But adjectival participle form is also available in past tense, "*pOya*". Moreover such forms can have inflections for Aspects also like, "*pOyirunna, pOyikkoVNtirikkunna, pOyirunna*..etc. Hence absence and presence of tense markers cannot be taken as the criteria for finiteness and non-finiteness in Malayalam.

But Traditional Grammars[57] discuss about only two types of verb forms, i.e. Dependent verbs and Independent verbs. Independent verbs have a sense of completeness in the sense that they do not depend on any other verbs in the utterance context. But dependent verbs cannot stand alone and they usually invoke a sense of incompleteness in the utterance context. Again dependent verbs are of two types, verbs which are dependent to nouns and verbs which are dependent to verbs.

[2]Government and Binding Theory https://en.wikipedia.org/wiki/Government_and_binding_theory

In this work, the criteria for finiteness and non-finiteness of a verb is decided based on its ability to stand independently in an utterance or not. Finite verbs can stand alone as completing the utterance and non-finite verbs cannot.

Form	Tense&Aspect	Meaning
pOkunnu	Present.Simple	go
pOyi	Past.Simple	went
pOkuM	Future.Simple	will go
pOyikkoNtirikkunnu	Present.Progressive	going
pOyikkoNtirunnu	Past.Progressive	was going
pOyikkoNtirikkuM	Future.Progressive	will be going
pOyirikkunnu	Prsent.Perfect	has gone
pOyirunnu	Past.Perfect	had gone
pOyirikkuM	Fut.Perfect	would/ should have gone

Table 2.5: Iflections for Tense & Aspect for the verb "to go"

2.3.4.1 Non-Finite Verbs

Non finite verbs are of two types, noun modifiers and verb modifiers. They are the participial form of verbs with all the properties of verbs maintained. In Dravidian languages, a sentence will usually have only single finite verb and rest of the verbs will be non-finite.

2.3.4.1.1 Relative Participles Relative participle is the most productive process to modify nouns in Malayalam[37]. Such forms are created by adding a suffix "*a*" on verbs in present tense and past tense. But unlike adjectives, they retain the properties of a noun modifier as well as a verb.

(5) parYayunna kAryam
say.PRST.adj-prtcpl matter
'the matter which (someone) say(s).'

In the above example, *parYayunna* is in simple present tense. This is created by adding the "*a*" to the simple present tense form of the root "*to say*- "*parYayunnu*. Suffix "*a*" can be added to present continuous or present perfect form of verb as in examples below.

(6) parYaffukoVNtirikkunna kAryam
say.PRST.Cont.adj-prtcpl matter
'the matter which (someone) has/have been saying).'

(7) parYaffirunna kAryam
say.PRST.PRF.adj-prtcpl matter
'the matter which (someone) had said).'

Similarly relative participle form in past tense can also be created by adding suffix "*a*".

(8) parYaffa kAryam
say.PAST.adj-prtcpl matter
'the matter which is said.'

2.3.4.1.2 Adverbial Participles Verb modifiers are dependent either on other non-finite verbs or on the finite verb. Such forms are created by adding suffixes which are in turn creates subordinate clauses and hence make discourse level relations. Various suffixes used are given in the table.

Suffixes "*i & itt* are used to describe the event that has completed or has to be completed before the main event.

(9) fAnZ vIttil poy**i** ooNu kalYYiccu.
I house.LOC go.prtcpl food eat.PAST
'Having gone home I ate food.'

In the above example, the action of going home has happened before the action of eating food. Suffixes "*i & itt* are used interchangeably to describe an event happened before.

"*AnZ*" describes the event that has to happen/happened after the main event.

(10) fAnZ UN kalYYIkk**AnZ** vIttil pokuM.
I food eat.prtcpl house.LOC go.FUT
'I will go home to eat.'

Here suffix "*AnZ*" describes that the event of eating will happen only after the action of going home.

"*uka* can be considered as an infinitive but such forms can come as main verb also. Suffix "*uka*" represents the status. In the example given below, meaning of the word "kalYYIkk**uka**" is "in the state of eating".

(11) fAnZ UN kalYYIkk**uka** AN.
I food eat.prtcpl is.COP
'I am in the state of eating food'

Such forms do not inflect for tense but for aspect. The tense information will be present in the finite verb.

(12) fAnZ pAtikkoVNtirikk**uka** Ayirunnu.
I sing.CONT.prtcpl be.PAST
'I was in the state of continuing to sing'

Simultaneous actions are described by the suffix "*irikke*".

(13) fAnZ pAtikkoVNt**irikkeV** avalYZ vannu.
I sing.CONT.prtcpl she come.PAST
'While i was singing she came.'

In the above example, action of coming happened while the action of singing was happening. conditional events are expressed by "*AlZ*".

(14) nI vann**AlZ** fAnZ varAM.
you come.conditional I come.FUT
'I will come if you come.'

Verbs with "*AnZ & uka*" suffixes do not represent any tense, but can inflect for aspects[37]. Whereas forms with all other suffixes can be inflected for tense and aspect.

Form	Suffix	Meaning
pOyi	i	having gone
pOyitt	itt	after having gone
pOkAnZ/pOkuvAnZ	AnZ	to go
pOka/pOkuka	uka	go(obligation)
pOyirikke	irikke	while gone
pOyAlZ	AlZ	go- conditional
pOkuMpolYZ	pOlZ	when (you) go
pOyappolYZ	pOlZ	when (you) have gone

Table 2.6: Non-finite/ Infinite verb forms

2.3.4.2 Pronominalised Verbs

In Dravidian languages, pronominalised verbs can be created from relative participle form of verbs. These forms can be considered as verbs as well as nouns since these words retain the properties of both nouns and verbs. *pOkunnaw, vannirikkunnavanZ, pAdiyavlYZ.*

- Considering them as nouns, they can inflect for all the cases,
- Considering them as verbs, they can inflect for various tense and aspects. Above all they can take arguments of verbs.

(15) nI **kettaw** sawyaM alla
'you hear.PAST+nml.sfx true no
'What you heard is not true'

(16) nI **kettawilZ** sawyaM illa
'you hear.PAST+nml.sfx+loc true no
'There is no truth is what you heard.'

2.3.5 Adjectives

Adjectives are the modifiers of the nouns. Adjectival suffix in Malayalam and in all other Dravidian languages is "*a*"[57]. Pure adjectives without the properties of verbs in Malayalam are very few like, "nalla(good)", "ceVrYiya(small)", "valiya(big)".. etc.

(17) cerYiy**a** manushyaruM valiy**a** lOkavuM
'little men+and big world+and
'little men and big world.'

2.3.6 Adverbs

Adverbs are the modifiers of verbs. In comparison with adjectives, adverbs have relatively free position in a sentence [37]. Adverbs usually specify time, place, manner etc of an event. Based on that there are different types of adverbs namely, spatial adverbs, temporal adverbs, Manner adverbs etc. With distal and proximal markers "*a & i*", there are separate suffixes for creating spatial, temporal and manner adverbs.

type	suffix	forms	meaning
Spatial	iteV	aviteV/iviteV	there/here
Temporal	polYZ	appolYZ/ippolYZ	that/this time
Temporal	nn	ann/inn	that day/ this day
manner	ffaneV	anffaneV/inffaneV	like that/this

Table 2.7: Various types of adverbs from distal & proximal markers

Other spatial, temporal and manner adverbs are wAlYYeV(under), mIweV(above), innaleV(yesterday), nAlYeV (tomorrow), pawukkeV(slowly), vegaM(fast), peVtteVnn(suddenly), otukkaM(finally), itakku(in between).

2.3.7 Conjunctions

Conjunctions are the connectives of two actions or entities. There are two types of conjunctions, Coordinative and subordinative. Coordinate conjunctions are words or phrases which connect two or more equal parts in a sentence(two NPs, two main clauses etc). Subordinate conjunctions connect two unequal parts in a sentence(a dependent clause with a main clause). In Malayalam, morphological or syntactical evidence for a verb being finite or non-finite in terms of features such as tense, agreement is disputable because of the multiple tense markers in verb forms and lack of agreement. Hence semantic completeness of an event would be the more suitable property for deciding finiteness and infiniteness of a sentence. Many of the common coordination and subordination function are fulfilled by suffixes which attached to Nouns or verbs.

2.3.7.1 Coordinate Conjunctions

Usual coordinate conjunctions like "*and*" and "*or*" are not single words but distributed suffixes that can be attached to nouns or verbs in Malayalam. Suffix "*um*" is for showing and-relation. With verbs, "*um*" attaches with infinitive form of verb.

(18) ayAlY**uM** njAn**uM**
he+and me+and
'He and me'

(19) avanZ elYYuwukay**uM** vAyikkukay**uM** ceywu
he.PRP write.INF+and read.INF+and do.PAST
'He wrote and read.'

Example 16 is the instance for conjunction suffix "*um*" is added to nouns. Where as Example 17 is to show the conjunction of two events where suffix "*um*" is added to the infinitival form of the verbs "*elYYuwuka*" and "*vAyikkukay*". Similarly Suffix "*o*" is for showing or-relation. Example 18 is the instance of adding disjunction suffix "*o*" to nouns for showing disjunction. Example 19 is for showing disjunction of 2 events.

(20) avan**O** njAn**O**
he.or me.or
'He or me'

(21) njAnZ varukay**o** kANukay**o** illa
I.PRP come.INF+or see.INF+or no
'I do not come or see.'

Other main coordinate conjunctions are *pakSheV, kAraNaM, pinneed* etc. Coordinate conjunctions connects semantically independent events.

(22) njAnZ vannu pakSheV avanZ vannilla
I.PRP came.PAST but he come.PAST.neg
'I came but he did not come.'

In the above example,

2.3.7.2 Subordinate Conjunctions

Number of subordinate conjunctions are more compared to coordinate conjunctions. Common subordinate conjunctions are *eVnkilZ, eVnnAluM, eVnn, eVnkiluM* etc. According to traditional grammar [57], "*eVnZ*" is a verb root which does not inflect for every tense, aspect and modality. This form refers to whatever came before that. But over the time, adverbial participle forms of this verb roots like *eVnkilZ, eVnnAluM, eVnn, eVnkiluM* became frozen and now being used as connectives. Since "*eVnZ*" refers to previous event, they are treated as subordinate conjunctions in this work.

(23) avanZ varuM **eVnkilZ** njAnZ varAM
he.PRP come.FUT if I come.FUT
'If he will come, then I will come.'

In the above given example both the two verbs "varuM" and "varAm" are finite, but the conjunct "eVnkilZ" makes the first verb "varuM" semantically dependent to the second verb "varAM". Hence "eVnkilZ" is a subordinate conjunction.

(24) avanZ varuM **eVnn** avanZ parYaffu.
he.PRP come.FUT if he say.PAST
'He told that he will come'

In the above example, "**eVnn**" behaves similar to "that" in English. Given below is an example of "concession". When conditional form **eVnkilZ** is added with "*um*", the word **eVnkiluM** gives the sense of concession.

(25) avanZ varilla **eVnkiluM** njAnZ avaneV vilikkuM
he.PRP come.NEG even though I he.ACC call.FUT
'Even though he will not come, I will call him.'

2.3.8 Interjections

Interjections are the words or phrases that is being used to express particular emotions or sentiments of the speaker. Such words are used very less in text when compared to speech. Few common interjections in Malayalam are *ayyO, O, daivmeV*.

(26) ayyO aw pOyi
alas that loss.PAST
'alas lost it.'

Chapter 3

Sandhi Splitter

3.1 Introduction

Sandhi splitting is the primary task for computational processing of text in all Dravidian languages. In these languages, words can join together with morpho-phonemic changes at the point of joining. This phenomenon is known as Sandhi. Sandhi splitter splits the string of conjoined words into individual words.

3.2 Sandhi

The word *Sandhi* has its origin in Sanskrit with the meaning "*union*". Sanskrit grammarian, Panini in his treatise on Sanskrit grammar called *Ashtadhyayi*, used the term *Samhita* for explaining the process *Sandhi*. *Samhita*[1] is the close proximity of two letters either within a word or between two words which results into the natural phonetic combination of these letters[27]. The process of *Sandhi* or "*union*" in languges, makes the energy consumption minimal for pronunciation. Gerard Huet[21] explains *Sandhi* as a phenomenon which occurs *when a word w_1 is followed by a word w_2, some terminal segment of w_1 merges with some initial segment of w_2 to be replaced by a smoothed phonetic interpolation, corresponding to minimizing the energy necessary to reconfigurate the vocal organs at the juncture between the words*. The presence of *Sandhi* is abundant in Sanskrit and all Dravidian languages. Sandhis are of two types, Internal and External. Internal Sandhi is within a word where it exists between root(s) and suffixes.

$$\mathbf{waw}(that) + \mathbf{smin}(LOC.suf) \rightarrow \mathbf{wasmin}(in\ that)$$

External sandhi is between two words.

$$\mathbf{wasya}(his) + \mathbf{eva}(only) \rightarrow \mathbf{wasyEva}(his\ only)$$

This property of languages ie joining different types of words, which in turn becomes a hurdle for computational processing. Due to Sanskrit's highly synthetic behaviour, even a sentence with 50-60

[1] parassannikarRaH saMhiwA - Ashtadhyayi:1.4.108

words can be found in old Sanskrit manuscripts . Compared to Sanskrit, *Sandhi* in Dravidian languages is relatively simple on two aspects.

- Sandhis are generally unambiguous as opposed to Sanskrit.
- Number of words inside a Sandhied string tend to be very less.

3.3 Sandhi in Malayalam

Even Though *Sandhi* in contemporary literature is less in comparison with old Malayalam literatures, the presence of *Sandhi* between words is still prevalent in both Spoken and written Malayalam.

avanAraaN

Above given is a sentence in Malayalam which means *"Who is he ?"*. It is composed of 3 independent words, namely "avan (*h*e)", "aar(*w*ho)" and "AN(*i*s)".

anganeyANennANaddehaM parYanjaw.

This is also a valid sentence in Malayalam, which can be roughly translated into English as *"He told that it is like that"*. Here the first string contains 4 individual words in it, namely "affaneV(*l*ike that)", "AN(*i*s)", "eVnn(*t*hat)", "AN(*i*s)", "AxxehaM(*h*e)" and the second string "parYaffaw" which means *"that which said"*.

However, in general a sandhied string can be split into valid words by splitting only at unique split-points in any context. for a word is very less in Malayalam.

3.3.1 Internal Sandhi

Internal Sandhi exists between a root or a stem with a suffix or a morpheme. In the example given below,

para + unnu = para**yu**nnu

Here **para** is a verb root with the meaning "to say" and **unnu** is an inflectional suffix for marking present tense. They join together to form para**yu**nnu, meaning "say"(PRES).

Jaya + ude = jaya**yu**de

Here **Jaya** is a proper noun, and **ude** is genetive marker. They join together to form jaya**yu**de, meaning "Jaya's".

3.3.2 External sandhi

External sandhi is between words. Two or more words join to form a single string of conjoined words.

ceyyu**M** + enkil = ceyyu**me**nkil

ceyyu**M**) is a finite verb with the meaning "will do" and **enkil** is a connective with meaning "if". They join together to form a single string ceyyu**me**nkil.

For computational purposes, *External Sandhi* is the matter of concern because, individual words present in the conjoined string cannot be identified.

3.4 Sandhi Splitting

Sandhi splitting is the process of splitting a string of conjoined words into a sequence of individual words, where each word in the sequence has the capacity to stand alone as a single word. To be precise, *Sandhi splitting* facilitates the task of individual word identification within such a string of conjoined words.

ceyyu**me**nkil = ceyyu**M** + enkil

3.4.1 Importance of Sandhi Splitting

Words are the important components of a meaningful text which explains why word identification is crucial for the computational processing of the text. All the text processing tasks require individual words in the text to be identified.

In Sanskrit and Dravidian languages, the identification of words becomes complex when words are joined to form a single string with euphonic changes at the point of joining. Moreover, the *Sandhi* can happen between any linguistic classes like, a noun and a verb, or a verb and a connective etc. This leads to misidentification of classes of words by POS tagger which eventually affects parsing. *Sandhi* acts as a bottle-neck for all term distribution based approaches for any NLP and IR task. Thus, for computational processing of these languages, *Sandhi* has to be split for identifying the individual words present in the conjoined string.

3.4.2 Sandhi Splitting vs Word Segmentation

Sandhi splitting is a different type of word segmentation problem. Languages like Chinese [5], Vietnamese [17] Japanese, do not mark the word boundaries explicitly. Highly agglutinative language like Turkish[40] also needs word segmentation for text processing. In these languages, words are just concatenated without any kind of morpho-phonemic change at the point of joining, whereas morpho-phonemic changes occur in Sanskrit and Dravidian languages at the point of joining [33].

3.5 Related works

Mittal [33] adopted a method for Sandhi splitting in Sanskrit using the concept of *optimality theory*[42], in such a way that it generates all possible splits and validates each split using a morph analyser. In another work, statistical methods like Gibbs Sampling and Dirichlet Process are adopted for Sanskrit Sandhi splitting [38]. [28] used CRF for making fully statistical Sandhi Splitters for Agglutinative languages.

To the best of our knowledge, only two related works are reported for the task of Sandhi splitting in Malayalam. Rule based compound word splitter for Malayalam [36], goes in the direction of identifying morphemes using rules and trie data structure. They adopted a general approach to split both external and internal sandhi which includes compound words. But splitting a compound word which is conceptually united will lead to the loss of linguistic meaning. Another work [14], which is a hybrid approach for sandhi splitting in Malayalam, employs a TnT tagger to tag whether the input string to be split or not and splits according to a predefined set of rules. But this particular work does not report any empirical results. In our approach, we precisely identify the split point in the string using statistical methods and then apply predefined set of character level rules to split the string.

3.6 Various Approaches

Broadly,there are 2 types of approaches that have been tried, one being Rule-based, and the other being a Hybrid method where the split points are identified statistically and the split is effected by character level rules to induce the morpho-phonemic changes.

3.6.1 Rule based Methods

Rule based methods are the ones in which hand-crafted rules are employed to accomplish a task. For the task of Sandhi Splitting, two rule based methods have been attempted, where one uses a look-up dictionary and the other uses a suffix list.

3.6.1.1 Look up Dictionary Based

In Look up Dictionary Based approach, a string is being split into its possible linear combinations of characters and then hand-crafted character level rules are applied to those two words to revert the morpho-phonemic changes that occurred post-Sandhi. Then those words are checked in the look-up dictionary for lexical validity. If those words are present in the dictionary, then it is considered a valid split.

3.6.1.2 Root and Suffix Based

The idea behind this method is that the basic structure for any word in any language will be $Root + Suffix$. The presence of a sequence of root-suffix pattern is a clear indication of a valid set of splits which is exploited in this method. Here, a given string is traversed backwards from the last character and checked against with the suffix list. On encountering a valid suffix, the backward traversal continues until another valid suffix is encountered. If the other suffix is found, then the split is done at the boundary of that suffix.

3.6.1.3 Problems with Rule based method

Eventhough rule based methods provide good insights about the problem, the solutions provided by them are not efficient. In the first method 3.6.1.1, it is a nearly impossible task to create the list of words which encompasses all the possible inflections and derivations of every possible word in the language. Moreover, it will be computationally a complex task to maintain a dictionary. The problem with second method3.6.1.2 is that the listed suffixes could be a valid non splittable part of a string. For such cases one will have to write many rules and that will be endless and inefficient. All rule based methods will have to maintain a list of rules or a list of suffixes/characters.

3.6.2 Hybrid Method

In a hybrid approach,split points are identified statistically and subsequently the string is split into words by applying a set of pre-defined character level sandhi rules **Table 1**. The scope of these rules are largely unambiguous because the split point is already determined statistically unlike a purely rule-based approach.

3.6.2.1 Sandhi Rules

Sandhi rules for external sandhi are identified out of a corpus of 400 sentences from Malayalam literature, which includes text from old literature(texts before 1990) as well as modern literature. In comparison with text from old literatures, modern literature have very less use of sandhi. By analysing

the text, we were able to identify 5 largely unambiguous character level rules which can be used to split the word, once the split point is statistically identified. Sandhi rules identified are listed in the **Table 1**.

R	Rule	Example
1	$(CSC)V_s =$ $(CSC)S + V$	vAkkilla = vAkk+ illa word(Noun)+no(verb)
2	$(ya/va)V_s = V$	pediyAN = pedi + AN fear(Noun) + is(verb)
3	$maV_s = M + V$	paNamilla = paNaM + illa money(Noun) + no(verb)
4	$(ra/la/lYYa/na/Na)V_s = rZ/lZ/lYYZ/nZ/NZ + V$	mukalYYilAN= mukalYilZ + AN above(Loc)+is(verb)
5	$CV_s =$ $(CS) + V$	ANeVnn = AN + eVnn is(verb)+that(quotative)
6	just split	avanZvannu = avanZ+vannu He(Noun)+Came(verb)

C=Consonant, V=Vowel, V_s=Vowel symbol, S=Schwa

Table 3.1: Sandhi rules

Rules in **Table 1** are given in a particular order corresponding to their inherent priority which avoids any clash in the order of application of rules. Every character level Sandhi rule is based on a consonant and a vowel. *Rule* **5** is the most general rule for Sandhi that we could identify in Malayalam which states that a group of a consonant and a vowel can be split into a consonant and a vowel. *Rule* **1** is a special case of *Rule* **5**, which is framed in order to treat the special case of consonant gemination with a schwa in between. This rule is being introduced , because the consonants specified in rules **2, 3** *and* **4** can appear as the last consonant in the case of geminated consonants. This will lead to an inappropriate split of the string by any of the rules **2, 3** *or* **4**. So every word, with an identified split point should be primarily checked against rule 1 to avoid wrong split by other rules in the case of a sandhi exhibiting consonant gemination. *Rule* **2** is to handle the process of phonemic insertion that happens post-Sandhi. In Malayalam, these extra characters ya/va can be inserted between words after sandhi in certain contexts. *Rule* **3** and *Rule* **4** are to handle the case of phonemic variation caused due to the process of Sandhi. *Rule* **3** enforces that, the letter 'ma' with a vowel symbol becomes 'M' and a vowel, while *Rule* **4** enforces that, the characters, 'ra/la/lYYa/na/Na' with a vowel symbol becomes *chillu*[2], 'rZ/lZ/lYYZ/nZ/NZ' and a vowel.

[2]A *chillu* is a pure consonant which can stand alone independent of vowels

3.6.2.2 Split point identification

The hybrid approach utilises the phonological changes [25] due to the presence of sandhi. The remaining part of the paper explains this approach in detail, with theoretical formulation, Experimental Results and Error analysis.

When the words are conjoined, they undergo phonological changes at the point of joining. These phonological changes can be evidential in identifying the split point in the given string of conjoined words. For example, words Wx and yZ are conjoined to form a new string $Wx^{'}y^{'}Z$. As a part of this process, substring x in the original string Wx has undergone a phonological change to become $x^{'}$ and the substring y in the original string yZ has undergone a phonological change to become $y^{'}$. We try to identify the split point between $x^{'}$ and $y^{'}$ in $Wx^{'}y^{'}Z$ using $x^{'}$ and $y^{'}$ as the evidence. Going forward, we use S_p to denote *S*plit Point and N_{sp} to denote Non Split Point. $P(S_p|x^{'}, y^{'})$ will give the probability of a character point between $x^{'}$ and $y^{'}$ within a conjoined string to be a split point. A character point will be classified as split point if,

$$P(S_p|x^{'}, y^{'}) > P(N_{sp}|x^{'}, y^{'}) \tag{3.1}$$

As per our observation, the phonological changes of x to $x^{'}$ and of y to $y^{'}$ are independent of each other. So,

$$P(S_p|x^{'}, y^{'}) = P(S_p|x^{'}) * P(S_p|y^{'}) \tag{3.2}$$

To produce the values of $x^{'}$ and $y^{'}$ for each character point, we take k character points backwards and k character points forward from that particular point. For example, the probability of the marked point $PointX$ in the string given in the **Figure 1** below to be a split point is given by **equation (3)**

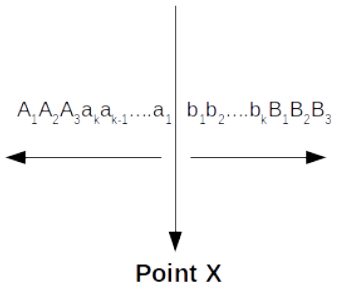


Figure 3.1: Split point

$$P(S_p|a_1a_2...a_k) * P(S_p|b_1b_2...b_k) \tag{3.3}$$

Where $a_1a_2...a_k$ is $x^{'}$ and $b_1b_2...b_k$ is $y^{'}$. The value of k is experimentally optimized.

The split points of different agglutinated strings in training data are annotated in the following format.

$$WordN = i_1, i_2, i_3, ...i_z$$

This indicates that the string $WordN$ needs to be splitted at "z" character indexes $i_1, i_2, i_3, ...i_z$ within the string.

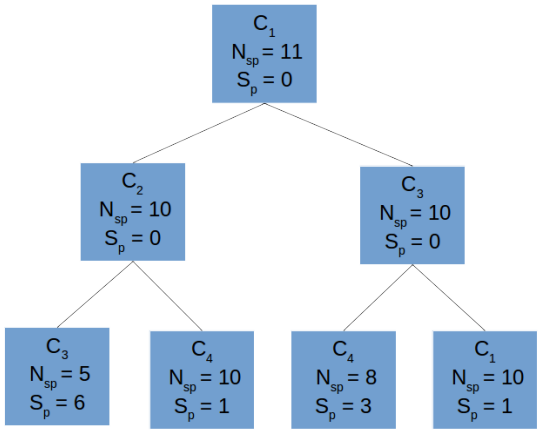


Figure 3.2: Word trie

We employ two Orthographic Tries [59] to statistically capture the phonological differences in $x^{'}$ and $y^{'}$ for split points and non-split points. A mould of orthographic tries used is given in **Figure 2**. The trie in **Figure 2** is trained with character sequences $c_1c_2c_3, c_1c_2c_4, c_1c_3c_4, c_1c_3c_5$. In the first trie, the path from root to k nodes represents the string $a_1a_2...a_k$. Each node $i(1 \leq i \leq k)$ in the path stores the number of occurrences of split points and non-split points in the entire training data which are preceded by $a_ia_{i-1}...a_1$. In the second trie, a path from root to k nodes represent the string $b_1b_2...b_k$. Each node $i(1 \leq i \leq k)$ in the path stores the number of occurrences of split-points and non-split points which are succeeded by $b_1b_2...b_i$. The frequencies stored in these trie nodes are used to calculate $P(C|a_1a_2...a_k)$ and $P(C|b_1b_2...b_k)$ where C is S_p or N_{sp}.

As split points are rare within a string $P(C|a_1a_2...a_k)$ and $P(C|b_1b_2...b_k)$ needs to be smoothed. For this purpose, we use the information stored along the depth of the tries for the strings $a_1a_2...a_k$ and $b_1b_2...b_k$ as follows

$$P(C|a_1a_2...a_k) = \sum_{i=initial\ skip}^{k} P(C|a_ia_{i+1}...a_k) \tag{3.4}$$

$$P(C|b_1b_2...b_k) = \sum_{i=initial\ skip}^{k} P(C|b_ib_{i+1}...b_k) \tag{3.5}$$

Here the initial-skip decides optimum 'smoothing range' within the string.The value of initial-skip decides the threshold phonological similarity that needs to be considered while smoothing. The experimental optimisation of initial-skip is done. The identified split points are splitted using the applicable sandhi rules.

3.6.2.3 Data and Results

3.6.2.3.1 Data Set We created a dataset which contained 2000 Sandhi bound words for training. Each of the split point within the word are annotated with the within-word index of the corresponding character point. The test data contains 1000 random words, out of which 260 are words with sandhi.

3.6.2.3.2 Results In our experiments, we tried evaluating the accuracy of split point identification, split accuracy of different sandhi rules and overall accuracy of the system. By overall accuracy, we mean percentage of the words with sandhi in which the split is exactly as expected. We have conducted experiments on split point identification with different values of k and initial Skip.

k-S	P	R	F	Accuracy
3-1	85.37	75	79.85	89.6
3-2	84.73	73.26	78.58	88.7
4-1	86.56	76.04	80.96	90.1
4-2	85.48	73.61	79.10	89
5-1	88.37	79.16	83.51	90.9
5-2	85.94	74.30	79.30	88.7
6-1	88.50	80.20	84.15	91.1
6-2	85.59	76.38	80.88	89.2

Table 3.2: Results

Here *P* implies precision, *R* implies Recall, *F* implies F-measure and *k-S* implies k and Initial skip respectively. k=6 and initial Skip=1 have shown the better result. As per our observation, phonological changes as a part of sandhi would not happen beyond a range of six characters in each of the participating words. So the upper bound for k value is taken as 6.

3.6.2.4 Error Analysis

In Split point identification, most of the incorrectly identified split points are character points between a word and inflectional suffix attached to it. As the system evolve, this error can be rectified by the use of a post-processor which maintains the finite list of inflectional suffixes in the language. Wrong splits in the middle of an actual word are very few in number and will reduce as the size of the training data increases.

Most of the unidentified split points are due to the presence of certain rare patterns. These errors will be reduced with the incorporation of words from diverse texts in the training data.

When it comes to rules, we have used only character level rules for splitting the identified split points. Rule 1,4 and 5 go ambiguous at certain rare contexts. To resolve this, certain word level information like POS tags are required. But for an accurate POS tagging, particularly for this disambiguation purpose, a sandhi splitter with a good level of accuracy is inevitable. Our Sandhi splitter can contribute for a better POS tagger. Vice versa, a POS tagger can complement the Sandhi splitter.

Rule	Accuracy
1	92.10
2	96.29
3	100
4	80.64
5	87.71

Table 3.3: Rule accuracy when k=6 and S= 1

3.6.3 Sandhi Splitter using CRF

In 2015 [28], the same idea of finding split points by mining character level patterns has been implemented along with the automatic word segmentation and correction using Conditional Random Fields. They used "Wx" transliteration scheme instead of utf-8 characters for their experiments. There are 2 stages, one for split point identification and other for segmentation with correct morpho-phonemic changes. In the first stage, each character is classified to split points and non split points, with morpho-phonemic change at the split point as features. In the second stage, in order to predict the correct morpho-phonemic change, they have extracted certain number of classes from the training data where

all the morpho-phonemic changes can be classified. Then they retrained the training data with this labels and finally outputs the change as labels. The experiment results have been presented for both Telugu and Malayalam. The reported scores are presented in the table given below.

Language	Training data	Test data	Precision	Recall	F-Measure	Accuracy
Malayalam	1926	1000	96.94	96.09	96.51	90.50

Table 3.4: Data and Results of CRF Sandhi splitter

Fully statistical sandhi splitter using CRF gives almost similar performance on the same data.

3.6.4 Various methods of evaluation

There are 3 different ways to evaluate the overall accuracy of a word segmentation system. The data can be represented in the algebraic form, where the conjoined string comes on the Left Hand Side and the individual words present in that conjoined string come on the Right Hand Side.

3.6.4.1 Method 1

In the first method, only those lines are considered as true where the LHS = RHS. Then the overall accuracy of a sandhi splitter is given as

$$\frac{Number\ of\ lines\ where\ LHS\ =\ RHS}{Total\ number\ of\ instances} \tag{3.6}$$

Disadvantage of this method is that, it misses out a a lot of correct words identified.

3.6.4.2 Method 2

In this method, all the correct tokens in RHS will be given *1/n*$^{\text{th}}$ score where "*n*" is the actual number of tokens should be present in RHS.

$$\frac{Sum\ of\ partial\ scores}{Total\ number\ of\ instances} \tag{3.7}$$

This method, does not provide any valid information about the performance.

3.6.4.3 Method 3

In this method, all the correct tokens on the RHS will be given 1. According to this, the overall accuracy of a sandhi splitter can be identified by

$$\frac{Number\ of\ correct\ words\ in\ RHS}{Total\ number\ of\ actual\ words} \tag{3.8}$$

Since the task of sandhi splitting is a word identification problem, the correct method of evaluation would be **Method 3** which takes the number of correct words identified into account. Hence in this work, **Method 3** will be followed.

Method	Accuracy
Method 1	91.1
Method 2	92.83
Method 3	87.5

Table 3.5: Results of various sandhi splitter evaluation methods

Due to the unavailability of the data used by [28], results of various evaluation methods explained above on that data could not be presented.

3.6.5 Conclusion

Various rule based, statistical and hybrid methods for sandhi splitting have been explained. Rule based methods are effort-intensive since management of dictionaries or rules are involved. But statistical and hybrid methods are comparatively less effort-intensive and give good performance. The task of statistical split point identification has confirmed that there exists a character level pattern in the text where the Sandhi can be split. Various evaluation methods for automatic sandhi splitters are also explained. It is found that the one which takes number of correct words into account would be the apt one for evaluating the overall accuracy of an automatic sandhi splitter since sandhi splitter facilitates the task of individual word identification.

Chapter 4

Part Of Speech Tagger

Words can be classified into different classes based on their grammatical properties known as Part of Speech which generally says the status of a word in a sentence. Part of Speech Tagger is tool which automatically assigns a Part of Speech label to every word in the input sentence.

4.1 Part of Speech

Classification of words based on their grammatical properties can be traced back to ”Niruakta”(BC 5th/6th), a work on Etymology by Sanskrit grammarian Yaaska. In this work, he talks only about four classes Noun, Verb, Prefixes and Particles. When it comes to Western counterparts, Plato and Aristotle also talked about similar classification. By the end of 2nd century BC, Dionysius Thrax distinguished between 8 word classes namely: Noun, Verb, Pronoun, Participle, Article, Preposition, Adverb, and Conjunction, which is still being followed in traditional as well as modern grammars. Traditionally the definition of Part Of Speech has been based on morphological and syntactic structure [61]. Schachter [50] states that POS seem to occur in every natural language.

4.2 Part of Speech Tagging

Parts-of-Speech Tagging is a process of assigning tags to the words in a text as corresponding to a particular part of speech, based on its definition and context. This is the first step towards understanding the structure of any language. It finds its major applications like Speech Recognition, Speech Synthesis, Information retrieval etc. A lot of work has been done on POS tagging. Input to a POS tagger will be a string of words and output will be the POS tag for each word in the input. POS tags give significant

amount of information about the word and its neighbors.

Various methods have been attempted for different languages and broadly those methods can be classified in to two types namely Rule based and Statistical.

4.2.1 Rule based

Rule based methods are where the hand-written rules are used to disambiguate and decide the Part-Of-Speech Tag of a word. The earliest algorithms for automatic POS tagger involved two stage architecture, where the first stage contained the use of dictionary to assign a list of possible POS-tags to the word and in the next stage, hand crafted rules were employed to decide the most suitable tag of the word from the list of possible tags in the given context[24]. *ENGTWOL* is a similar rule based POS tagger with two stage architecture, where the first stage contains all the entries in the lexicon annotated with their morph information and second stage contains morph level and syntactic constraints. In this method also, the first stage gives a list of all the possible POS-tags for a words and second stage disambiguate and decides the most suitable POS-tag.

4.2.2 Statistical

Statistical taggers generally resolve tagging ambiguities by using a training corpus to compute the probability of a given word with respect to its context[24]. Mainly there are 3 approaches, Supervised, Semi-Supervised and Unsupervised.

There are 2 types of methods in supervised learning. One is being Generative and other is being Discriminative. Generative methods are where the joint distribution of individual labels are calculated. Whereas Discriminative models calculates the conditional probability with respect to the features.

4.2.3 POS Taggers for Indian Languages

Various rule based, statistical and Neural Network based methods have been attempted for Part-Of-Speech tagging in various Indian languages.

One of the major attempt for creating POS Tagger for morphologically rich Indian languages was using lexicon and decision tree algorithm [56]. Aniket [12] used MEMM, which showed a considerable improvement in accuracy. Another work in Bengali language showed that Performance issues created by stochastic systems can be improved by leveraging morphological information [13]. Manish [52]

used HMM and Naive stemming POS tagging which showed that morphological information can be harnessed without the morphological analyzer or lexicons. Asif [19] has experimentally showed that SVM works better than CRF,HMM and MEMM for Bangla. But on the other hand, Agarwal[2] showed that CRF consistently outperformed both MEMM and HMM for POS tagging in Hindi and Telugu. Coming to the language Kokborok, SVM outperformed CRF in POS tagging[18]. A CRF based POS-Tagger for Gujarathi [41] also has been reported. Meanwhile [20] attempted to create a rule based POS tagger for Hindi, using Regular Expressions, Rules and Lexicons which performed par with statistical models.

Similarly, various works have been reported for Telugu, Kannada, and Tamil using statistical methods like HMM, CRF, SVM, MEMM [43, 44, 16]. It is found that SVM and CRF outperformed both HMM and MEMM [26].

Various works[46, 30, 4, 23] have been published for POS tagging in Malayalam. Since these POS taggers failed to provide proper empirical results, models, data or code, we propose our own POS tagger.

4.3 Corpus creation

Malayalam has an annotated corpus which is created as the part of ILMT project. But this corpus is not Sandhi split, and contains a substantial amount of of noise which is mostly unrecoverable. Hence we are creating a separate corpus which has subsequently Sandhi split, POS Tagged and Chunked. For the task of POS Tagging, a general news corpus of 70k words have been self annotated manually with the help of bootstrapping. Annotation is done using an Annotation tool Sanchay[54].

4.3.1 Tagging Scheme

Tag-set used for POS annotation was BIS tag-set. BIS tag-set is a hierarchical tag set designed for various Indian languages including Malayalam. For Malayalam, it contains 45 tags including the root tag in the hierarchy. Annotation of Parts-Of-Speech has been done to facilitate the dependency parsing using Computational Paninian Grammar(CPG)[7]. Hence there are various differences in Parts-Of-Speech tag given to a word in comparison to the usual tags.

4.3.1.1 Differences in tagging

There are differences in terms of definitions given for certain category or type of words in comparison with usual linguistic definitions. Those differences are mentioned below.

- NST is a special tag added for Indian languages. They are nouns denoting Time and Space like "purYaww(outside)", "mump(before)". In these languages, such words can function as nouns and in certain contexts as post-positions also. Hence such words are given a separate tag NST [6].
- Pronominalised verbs are the verbs with the properties of verbs as well as nouns. Such words are tagged as Non-Finite Verbs(V_VM_VNF) since they can take arguments. The noun-inflections present on them will be captured at the level of morph analysis.
- Only manner adverbs are tagged as "Adverbs"(RB), because spacial and temporal words have "karaka" relations with the verb. Hence they will be considered as Nouns(NN).

4.4 Parts-OF-Speech Tagger using Naive Bayes

Malayalam is a morphologically rich language. From a detailed linguistic analysis, it is understood that most of the POS tags can be identified from the prefix and suffix information present in the word and the importance of relative position of the word in identifying the POS tags is relatively very less. Hence to verify this, similar method used for the task of Sandhi Splitting, ie Character level Trie Based Naive Bayes, has been implemented. This method captures the prefix and suffix information present in the word. A single fold experiment has been conducted and the results are presented below.

Training data	Test data	Precision	Recall	F-Measure	Accuracy
56k	14k	90.48	90.52	90.50	90.48

Table 4.1: Results of POS Tagger using Naive Bayes

This experiment validates the hypothesis that most of the POS tags can be identified from the prefix and suffix information present in the word. But the disadvantage of this method is that, it cannot capture position information. Hence in order to capture both positional and morphological information we decided to use Conditional Random Fields Classifier. Moreover it is experimentally proved that CRF outperforms generative approaches like HMM and MEMM in POS Tagging in Telugu and Hindi [2].

4.5 Conditional Random Fields

Conditional random fields (CRF) are a probabilistic framework for labeling and segmenting structured data introduced by Lafferty in 2001 [29]. CRF is a discriminative model where conditional probability distribution over label sequences given a particular observation sequence is calculated rather than a joint distribution over both label and observation sequences. The primary advantage of CRF over Hidden Markov Models[45] is their conditional nature, resulting in the relaxation of the independence assumptions required by HMM in order to ensure tractable inference. Additionally, CRF avoids the label bias problem, a weakness exhibited by Maximum Entropy Markov Models (MEMMs)[48] and other conditional Markov models based on directed graphical models. CRFs outperform both MEMMs and HMMs on a number of real-world tasks in many fields, including in Bioinformatics, Computational Linguistics, and Speech Recognition.

4.6 Experiments

Various experiments with CRF++[1] have been conducted with different feature templates. 5 fold validation has been done for every experiment. Total size of the annotated corpus is 70k and each fold contains 14k words. In every fold, training data is 56k and test data is 14k.

4.6.1 Experiment Type - 1

The general criteria for identifying the POS tag of a word also includes its context, which means POS tag of the current word can also be influenced by the POS tag of the previous and/or subsequent word(s). Hence the first chosen feature for the task of POS tagging was the current word and its two previous and subsequent words along with the combination of current and previous/subsequent words.

4.6.1.1 Result

Given below are the evaluation scores for Experiment Type - 2. This will be considered as the baseline result.

[1] https://taku910.github.io/crfpp/

Exp.No	Feature	Precision	Recall	F-Measure	Accuracy
1	2w+1 w combined	79.15	79.18	79.17	79.14

Table 4.2: Results of Experiment Type -1

4.6.1.2 Error Analysis

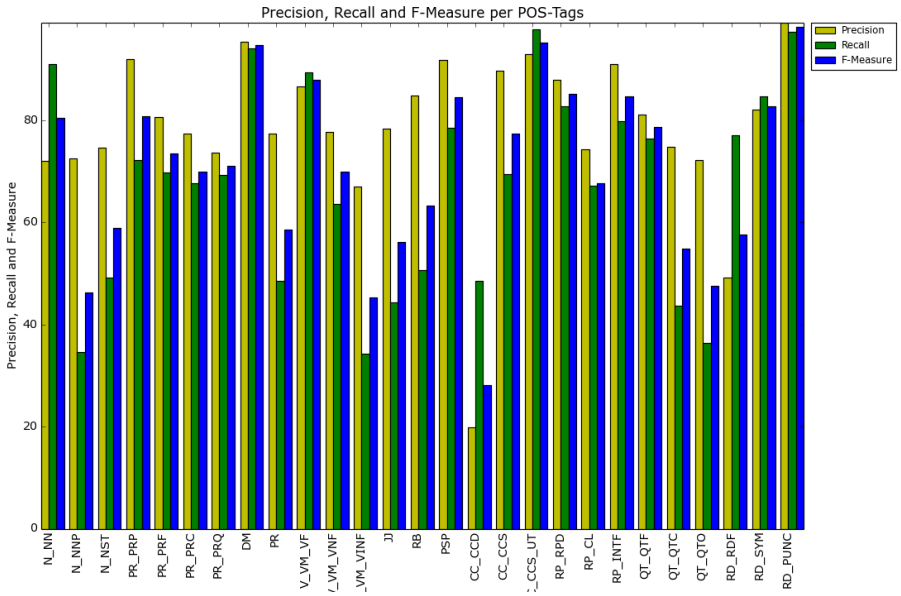


Figure 4.1: Results - Exp.No 1

This first experiment is to study the performance when only word level contexts are used. This experiment gave an overall accuracy of 79%. For most of the tags, recall is less than 60% which indicates that using only word level contexts is not sufficient for deciding the tags.

The major confusion in tags is in identifying nouns(N_NN) and proper nouns(N_NNP). Proper Nouns are nouns which represent the name of a Person/Place/Organisation/Products etc. Since their syntactical behavior is similar to those of nouns, most of the NNPs are getting identified as NNs.

Another source of error is with infinitive form of verb(V_VM_VINF). Most of the times, VINF got confused with non-finite verb (VNF), because VINF and VNF share similar syntax. VINF got confused also with NN. Ideally verb forms with the suffix, "*uka & AnZ* are tagged as VINF. Forms with '*uka* " are ambiguous in that they can function as infinitives as well as nominalized verbs. Hence we can find

nouns and VINFs in the same context.

This experiment shows that identification of tags like ”CC_CCD”(Coordinate conjunctions), ”RD_RDF”(Foreign words), PR(Pronouns), ”JJ”(Adjectives), ”RB”(Adverbs) etc are comparatively less identifiable by using only the contextual features. Number of instances with such tags are very less in the corpus. Pronouns have syntactical as well as morphological behavior similar to that of nouns. Most of the times, these tags are wrongly identified as Nouns, since the number of nouns are relatively very high. N_NSTs are nouns representing spatial and temporal things like below, above, etc. N_NSTs are getting confused with nouns since N_NST can be a noun according to the context. The other confusing tag for “JJ” is ”QT_QTF”(quantifier). Since the position QTF is before noun, some times “JJ”s are getting tagged as QTF. Because of the scrambled position, “RB”s are confused with the ”NN”. In few cases, “RB”s can come in the beginning of sentence whereas the verb which it qualifies may be in the middle or at the end of the sentence.

To summarize, number of nouns in Malayalam is very high when compared to any other class. But in order to capture the context of a particular category, it requires a lot of instances in the training data since Malayalam is a relatively free word order language. Similarly the finer tags like VINF,VNF, VF for verbs, QTF, QTO, QTC for quantifiers etc share similar contexts. Most of the words from categories other than nouns and verbs have very high level of morphological similarity with nouns and verbs. Hence the tag is not precisely decidable for the system with only word level context.

4.6.2 Experiment Type - 2

From Experiment Type - 1 it is visible that only using word level context as feature is not powerful enough to identify the finer distinctions. Since Malayalam is a morphologically very rich language, the next experiment is conducted with morphological features. Since morph analyser for Malayalam is not available, first and last five characters of each word has been taken as extra features along with the features of previous experiment. The intuition behind taking first and last five characters of each word is that in almost all these words, these characters can capture the prefix and suffix information.

4.6.2.1 Results

Given below are the evaluation scores for Experiment Type - 2.

Exp.No	Feature	Precision	Recall	F-Measure	Accuracy
2	2w+5p+5s+1 w combined	91.078	91.11	91.09	91.07
3	2w+5p+5s	91.20	91.24	91.22	91.20
4	1w+5p+5s	91.24	91.28	91.26	91.24
5	1w+4p+6s	91.25	91.29	91.27	91.25
6	1w+3p+7s	91.06	91.10	91.08	91.05

Table 4.3: Results of Experiment type 2

4.6.2.2 Error Analysis

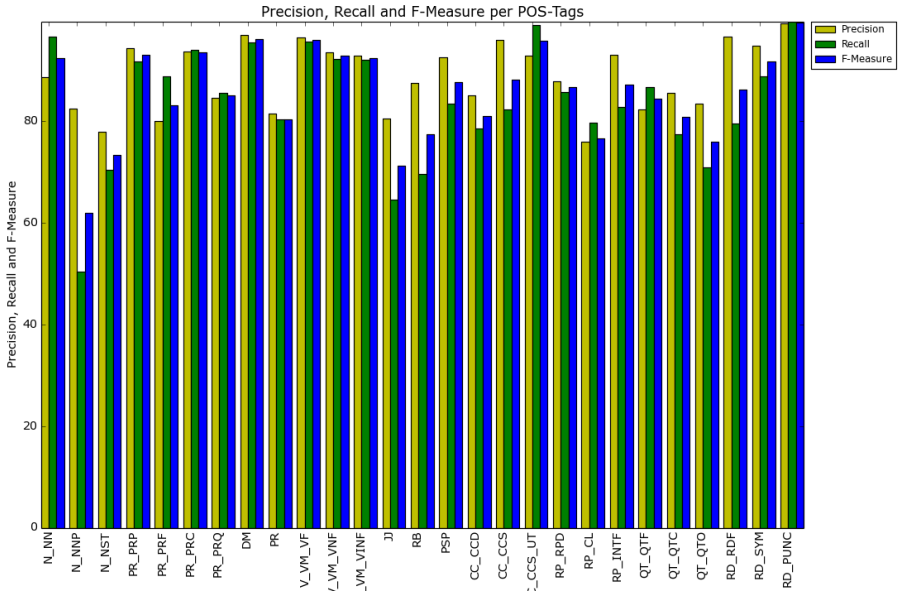


Figure 4.2: Results - Exp.No 2

In Exp.No2, morph level features have been included. This experiment showed a 12% increase in overall performance of the tagger compared to the baseline. In comparison to Exp.No.1, both Precision as well as Recall of every tag has been considerably increased, especially those tags whose figures were very less. However tags like ”NNP”, ”N_NST” , “JJ”, “RB”, “QT_QTO”(ordinals) have less Recall when compared to other tags.

Major source of the errors is with the tag NNP. Due to the morphological and syntactical similarity of NNPs with NNs, many instances of NNPs got tagged as NN. Recognition of such Named Entities

itself is one of the toughest problems in NLP.

Second major source of errors is with JJ(adjectives). ”JJ”s are mostly confused with the tag ”NN”. Adjectives in Malayalam can be made out of nouns by adding the adjectival suffix “*a*” as **svAwanwrya** in “**svAwanwrya** *samaraM*(freedom struggle)”. **svAwanwryaM** is the actual word and when it comes to a compound that becomes **svAwanwrya** and such forms are tagged as “JJ”. Since the morphological similarity is high, JJs are getting tagged as NNs. similarly JJs are confused with QT_QTFs because both JJs and QTFs modify noun and their position is mostly before noun.

Third source of error with RB. Because of the morphological similarity and scrambled position, “RB”s are confused with the tag ”NN”. In few cases, “RB”s can come in the beginning of the sentence where the word it qualifies may be in the middle or end of the sentence.

The fourth source of error is NST. As mentioned earlier, NSTs are spatial and temporal nouns. Due to the morphological similarity and position of them in sentences as of nouns, they are getting tagged as NNs.

Fifth source of errors are cardinals(QT_QTC) and ordinals(QT_QTO). In many instances, due to their contextual and morphological similarities, QT_QTC is getting tagged as QT_QTO and vice verse. similarly, both the tags are confused with quantifiers(QT_QTF).

To summarize, most of the tags are confused with nouns since number of nouns in training data is very high in comparison with their respective frequency. Hence, the morphological similarity with confused tags become very high. Moreover finer distinction of finite, non-finite, and infinite verbs are confusing for the tagger because, each of the category has only very subtle morphological variation with the other.

4.6.3 Experiment Type - 3

This particular experiment is to verify the power of morphological features in deciding the POS tag of a word. In manual tagging, most of the times, the tags are decided by the morphological features present in the word.

4.6.3.1 Results

Given below are the evaluation scores for Experiment Type - 3.

Exp.No	Feature	Precision	Recall	F-Measure	Accuracy
1	4p+6s	89.91	89.96	89.93	89.91

Table 4.4: Results of Experiment Type - 3

4.6.3.2 Error Analysis

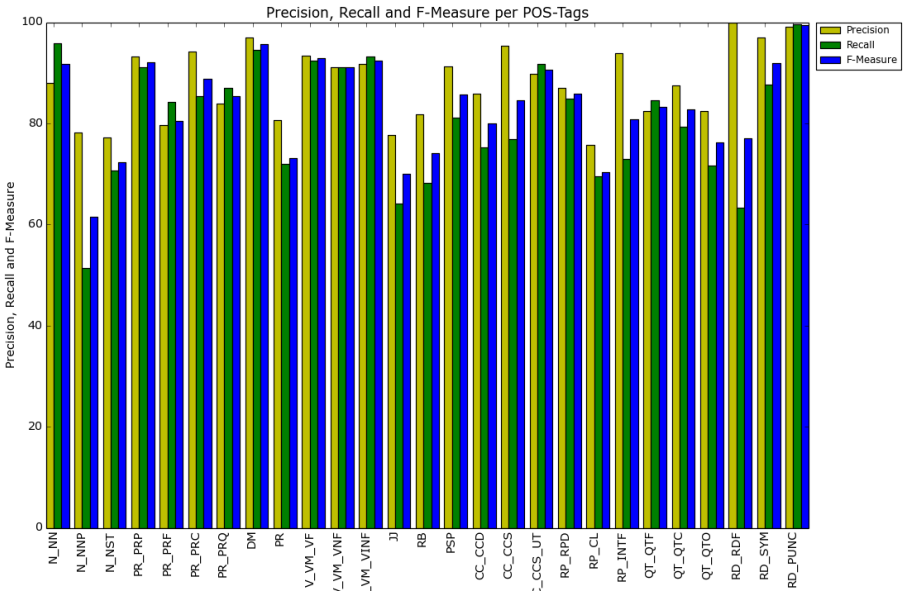


Figure 4.3: Results - Exp.No 7

In comparison to Experiment 4.1, all the tags except “JJ” and “RB” maintained a good level of Precision and Recall. Confusion between “JJ” and “NN”, as well as “RB” and “NN” also has been increased. This says that both RB and JJ need context level information as well for deciding the tag. Precision and Recall of tags like ”CC_CCD”, ”RD_RDF”, “RP_NEG” which were very less in Experiment 4.1 have been increased considerably, because morphological information is the main deciding factor.

4.7 Overall Results

Given below are the evaluation scores for various experiments conducted.

Exp.No	Feature	Precision	Recall	F-Measure	Accuracy
1	2w+1 w combined(**Base Line**)	79.15	79.18	79.17	79.14
2	2w+5p+5s+1 w combined	91.078	91.11	91.09	91.07
3	2w+5p+5s	91.20	91.24	91.22	91.20
4	1w+5p+5s	91.24	91.28	91.26	91.24
5	1w+4p+6s	91.25	91.29	91.27	91.25
6	1w+3p+7s	91.06	91.10	91.08	91.05
7	4p+6s	89.91	89.96	89.93	89.91
8	1rf+4p+6s	90.10	90.14	90.12	90.10
9	1lf+4p+6s	90.56	90.60	90.58	90.56

Table 4.5: Results

4.8 Conclusion

This chapter explained various experiments conducted for creating Parts-Of-Speech Tagger for Malayalam. Parts-Of-Speech Tagger has been created using Conditional Random Fields and experiments have been conducted with various features. Experiment with the feature 4 prefix and 6 suffix characters and single word context gave the highest accuracy 91.25%. From these experiments, it is visible that, Malayalam has more dependency over word internal features like suffix and prefix information than word external information like word contexts in deciding the grammatical category of a word.

Chapter 5

Chunker

Structure of a sentence is decided by the phrases/constituents present in in that sentence. Phrases are a group of words or possibly a single word that functions as a meaningful unit of a sentence which is determined by the grammatical category of words. Chunks are non-recursive phrases. Chunking is the identification of chunks. [1]

5.1 Chunking

Chunking is a task of identification of non-recursive phrases from an input sentence. It has been introduced by Abney[35] as a response to reduce the computational effort at the level of full parsing by assigning partial structure to a sentence. One of the fundamental principles of chunking is that the structure of chunks is more dependent on syntactic restrictions which can be represented by templates [35].

5.2 Chunks

Concept of chunks in the field of Natural Language Processing has been introduced by Steven Abney in 1992 [1]. According to him, a chunk consists of a single content word surrounded by a constellation of function words. To be precise, chunks are non-recursive cores of phrases which contains only the head or content word and its modifiers.

Phrases $\rightarrow$ (S (NP The bald man) (VP was (VP sitting (PP on (NP his suitcase)))))

Chunks $\rightarrow$ [The bald man]NP [was sitting]VP [on]PP [his suitcase]NP

In computational terms, for Abney[1], chunks are connected sub-graphs of a sentences parse tree. They

are defined in terms of a content word and their own syntactic structure that can be represented in the form of a tree. However a chunk does not include all the descendants of the root node that may be present in the parse-tree of the complete sentence.

Two heads of the same lexical category are not allowed inside a chunk. Consequently, "Johns house" in English will have two chunks, [Johns] and [house] since both "John" and "House" are Nouns. Similarly, verb complements are not grouped inside the verb chunk and they form separate chunks. The English sentence "*The bald man was sitting on his suitcase*' can be grouped into four chunks [The bald man], [was sitting] and [on]PP [his suitcase].

Concept of chunks for Indian languages[6] has been defined as "*A* minimal (non recursive) phrase(partial structure) consisting of correlated, inseparable words/entities, such that the intra-chunk dependencies are not distorted.

5.3 Previous works

In the literature, one can find various methods applied to perform the task of chunking such as Finite-State-Transducers, Transformation-based Learning(TBL), Memory-based Learning(MBL), Hidden Markov Models(HMM), Maximum Entropy(MEMM), Support Vector Machines(SVM), Conditional Random Fields(CRF) etc.

The first attempt for shallow parsing is by Abney[1]using hand-crafted Finite-State-Transducers. Ramshow and Marcus[47] used transformation based learning for NP chunking, where the problem of chunking has been treated as a tagging problem. Memory Based Learning(MBL) techniques also showed promising results in chunking [10]. It is a learning method where it stores the entire training data with its respective probabilities for extrapolation. But MBLs are not capable of handling irrelevant features which effects the performance [49]. CoNLL shared task gave a big boost to the research in the area of shallow parsing. Various experiments considered chunking as a sequence labeling problem which is still being followed. Molina and Pla [34] presented a new method for shallow parsing using specialized HMM. This method gave better results than MBL. Generalization of Winnow algorithm [62] was one of the other methods applied for shallow parsing. Winnow algorithms are known for their robustness to irrelevant features and efficiency in performing linearly separable tasks. Since chunking is not a linearly separable task, they modified the algorithm to perform on both linearly separable and non-separable data. This method was one of the best methods that featured in CoNLL 2000. The AL-

Lis system [15] is a rule induction system where initial system is refined with contextualization and lexicalisation operators. This incorporates linguistically motivated prior knowledge which showed a significant improvement in the performance. Shallow parser based on the Conditional Random Fields [51] outperformed all the previous methods.

5.4 Chunkers for Indian Languages

Various methods have been tried for chunking in Indian languages in last decade. The first work in the direction of using machine learning for Chunking is by using HMM [53]. They claim that for certain chunk categories, using only POS tag as feature gives better performance than using both the word and POS tag as the feature. Aniket [11] presented a Chunker based on Maximum Entropy Models. A rule based method proposed by Smriti Singh [55] identifies noun groups and verb groups by morphological and POS tag features.

Only one work on Chunking has been reported in Malayalam [46]. This used TnT tagger for chunking. Since this chunker is not available, we propose our own chunker.

5.5 Chunking in Malayalam

In Malayalam, case markers and auxiliary verbs come together with nouns and verbs respectively because of highly agglutinative nature of the language. As a result, most of the chunks will contain only one word. Even though there are many instances where modifiers occur separately.

- Noun compounds.
  vyAja vArwwa(fake news) → [vyAja vArZtha]NP

- Noun modifiers like pure adjectives, quantifiers, ordinals, cardinals are coming before nouns
  nalla kutti(good child) → [nalla kutti]NP

- post positions coming after.
  aviteV ninn(from there) → [aviteV ninn]NP

- Particles post verbs and nouns.
  pokAnZ wnneV wirumAniccu.(decided just to go) → [pokAnZ wnneV]VGNF [wirumAniccu]VGF

In order to group the above given types, chunking is needed in Malayalam although most of the chunks will be with a single word.

5.6 Data

70k data annotated for Parts-Of-Speech has been annotated with chunk information using IIIT-Chunk-tagset. This tagset contains 10 tags. Chunk annotation has been done manually as well as by validating a the output of the chunker with the help of an Annotation tool Sanchay.

5.7 Current Approach

Empirical experiments show that Conditional Random Fields(CRF) gives better results than any other supervised machine learning methods. Manish Agarwal[2] showed that for Hindi and Telugu, CRF consistently gave better results than MEMM and HMM for Chunking. Hence all the experiments have been done CRF only.

5.8 Experiments

Standard feature template used for shallow parsing has been tried for the first experiment, where two preceding and succeeding words with their POS tags have been taken as features along with its combination. This gave an accuracy of 94.30%. In the second experiment, the same feature template has been used without the combination features, which resulted in a marginal increase in precision, recall and accuracy. Third and fourth experiments were to check the dependence of chunks over the number of words preceding the succeeding words. It showed that the context window less than or more than 2 result in decrease in the overall performance.

5.9 Results & Error Analysis

From all the results provided, it is visible that there are only very slight variations in the accuracy, which can be disregarded. But there are few common errors which are not being corrected by the features experimented. NP chunks are the ones with less accuracy in terms of precision and recall. Errors are mainly due to separated noun compounds. According to the guidelines for chunking, two words from

No	Feature	Precision	Recall	F-Measure	Accuracy
1	2w+2w+pos+combination	91.2	93.03	92.10	94.30
2	2w+2w+pos	91.05	93.18	92.10	94.33
3	3w+3w+pos	90.03	92.88	91.43	94.03
4	1w+1w+pos	90.24	93.08	91.63	94.19

Table 5.1: Accuracy of chunker with various feature templates

Tag	Precision	Recall	F-Measure
BLK	99.682	98.724	99.198
CCP	95.65	98.4	96.996
JJP	25	10.614	14.22
NP	86.36	89.518	87.91
RBP	96.55	96.896	96.722
VGF	96.548	97.852	97.196
VGNF	96.048	97.876	96.95
VGINF	94.21	96.08	95.072
FRAGP	0	0	0
NEGP	0	0	0

Table 5.2: Tag-wise scores for chunker

the same open class should not be chunked together. Instances of single nouns with its modifiers are very high in the corpus. Whereas in the case of compound nouns, which are separated, noun should be chunked together because they together give the meaning. Hence such instances are always confusing for the chunker. Number of instances of FRAGP, NEGP, are less than 5 in the whole corpus. hence this cannot be learned. Tag for Adjectival Phrases, ie JJPs is given only when the adjectives are coming post noun. Such instances are also very less in the whole corpus, hence this also will be tough for the chunker.

5.10 Conclusion

Various experiments with different features for creating chunker for Malayalam have been explained in this chapter. The template with 2 words before and after the current word along with their POS tags as features gave the highest accuracy 94.33%. Experiments show that this template is the optimal one since accuracy decreases when there is an increase or decrease in the number of contexts. When compared

to Parts-Of-Speech Tagging, chunking is relatively simple, since the number of classes to be learned is very less and the model mainly depends only on the word contexts.

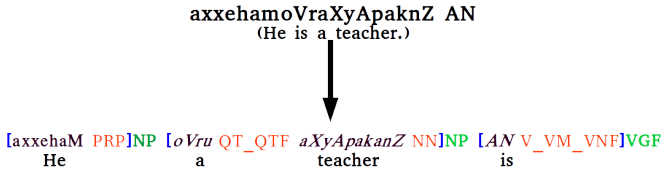
Chapter 6

Shallow Parser

6.1 Shallow Parsing

Shallow parsing is the task of identifying the non recursive phrases from a given sentence. A non recursive phrase contains head word and its modifiers. Shallow parsing is a collective process in an order by Tokeniser, Sandhi splitter, POS tagger and Chunker. A raw text will be given as the input and the system tokenises, identifies individual words using Sandhi splitter, assigns POS tags to each word and groups them to non recursive phrases and outputs.

To the best of our knowledge, no previous works have been reported for shallow parsing in Malayalam.



6.2 Architecture of the Shallow Parser

'Current shallow parser has 4 main modules in its architecture namely Tokeniser, Sandhi Splitter, Part-Of-Speech Tagger and Chunker. Input to the Shallow Parser is a raw text and the output is a chunked text with its POS and Chunk information of every word present in it. The diagram of the architecture is given below in Figure 6.1 which is given earlier in the Introduction.

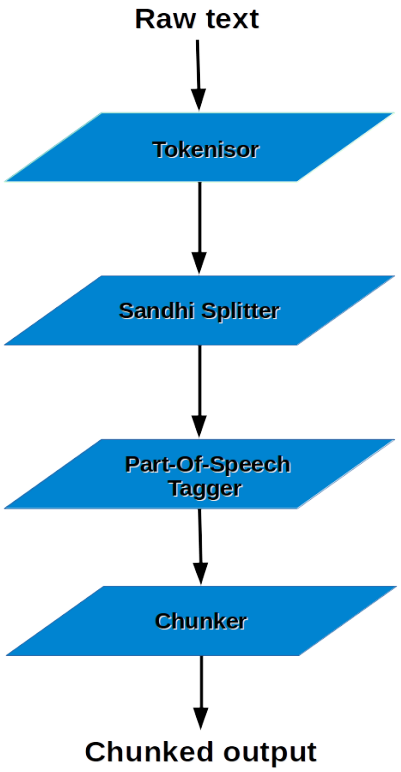


Figure 6.1: Pipeline Architecture

In this architecture, primarily the raw text will be taken as the input and each line will be split into individual tokens including punctuation marks using a tokeniser. The tokeniser for Malayalam is a rule based system. Output of the tokeniser will be passed to the Sandhi Splitter as its input.

Sandhi Splitter identifies the individual words present(if any) in each token and splits them into separate tokens. We implemented a statistical sandhi splitter using the character level patterns which identifies the sandhi splits with 87% accuracy. Output of the Sandhi splitter will be given as the input for Part-Of-Speech tagger.

POS tagger gives grammatical tags for each token present in the input based on various word internal and word external features.Output of the POS tagger will betaken as the input for Chunking.

Chunker identifies the non-recursive phrases from given input sentences. Experiments have been

conducted with chunker by varying the training features. Output of the chunker will be the final output of this system.

6.3 Data for Experiments

Manually annotated 70k POS and Chunk annotated corpus which we used for POS tagging and Chunking(4, 5) has been used for conducting various experiments for shallow parsing. Out of 70k data, 8k data has been taken as test data and remaining 62k as training data for POS tagging and Chunking. For Sandhi splitter, training data is 2k as mentioned in chapter 3 and test data will be the same 8k data which were employed for POS tagging and Chunking.

6.4 Experiments

Two types of experiments have been conducted to evaluate the error propagation rate of the Malayalam shallow parser pipeline. In the first type of experiments, individual modules in the pipeline are considered as independent of the output of previous modules. In the second type of experiment individual modules are considered as dependent on the output of previous modules.

6.4.1 Experiment Type - 1

In these experiments, input to each module namely sandhi splitter, POS tagger and chunker is not be affected by the performance of previous modules. This experiment evaluates the performance of individual modules with respect to the current train and test data.

6.4.1.1 Results

feature	Precision	Recall	F-Measure	Accuracy
6-1	91.77	62.95	74.68	88.46

Table 6.1: Result of Sandhi Splitter

No	Tag	Precision	Recall	F-Measure	Accuracy
2	1w+4p+6s	90.45	90.49	90.47	90.45

Table 6.2: Result of POS tagger

No	Tag	Precision	Recall	F-Measure	Accuracy
1	2w+2w+pos	88.47	91.55	89.98	92.92

Table 6.3: Result of Chunker

6.4.2 Experiment Type - 2

In these experiments, output of one module will be given as input to the next module, hence the performance of the previous module effects the next module. These experiments are to evaluate the rate of error propagation from each module which eventually affects the final output.

6.4.2.1 Method of Evaluation of Pipeline

Evaluation of Sandhi splitter and POS taggers are based on words. If the number of words in the test data and gold data does not match, evaluation of the system becomes difficult. Sandhi splitter is an important module in this pipeline. Sandhi splitter splits a token into individual words if the token has more than one word present in it. Sandhi splitter employed in this pipeline is a hybrid system which gives 88% accuracy. This system can produce 2 major errors which becomes a hurdle in the process of evaluation of all the subsequent modules.

- Not splitting a token which has to be split into words.
- Splitting a token which should not have been split.

By these errors, number of words in the test data and gold data does not match which makes the usual evaluation process impossible. In order to make the evaluation possible, the data representation of all the output of the Sandhi splitter as well as the gold data have been stored in such a way that it will be able to represent whether the input got split or not.

$$SandhiedString = Sandhied/P_{tag}/C_{tag} + string/P_{tag}/C_{tag}$$

P_{tag} = POS-tag, C_{tag} = Chunk-tag

This representation brings both the system output and gold data in the same format which enables the one-to-one comparison of strings. Accuracy of the pipeline is calculated based on number of words that got correct surface form (in sandhi splitting), POS tag. The final accuracy is based on the number of chunks correctly identified with correct number of proper words and POS tags.

6.4.2.2 Results

Shallow Parser pipeline evaluation scores are given below.

tokeniser+sandhi	tokeniser+sandhi+pos	tokeniser+sandhi+pos+chunk
88.46	79.87	71.38

Table 6.4: Pipeline Accuracies

6.5 Error Analysis

Accuracy of the sandhi splitter is 88%. This Sandhi splitter creates 2 types of errors as mentioned in 6.4.2.1.

1. Not splitting a token which has to be split into words.

2. Splitting a token which should not have been split.

In this experiment, error 1 is more prevalent than error 2. For example, "*arYivilla*(no knowledge)" should have been split into "*arYiv*(knowledge) and *illa*(no)". But the system failed to do so. This problem is due to the lack of diverse patterns in training data. When it comes to error 2, split occurs either between a root and its suffix or just splits in common sandhi split points like "*ya, va* or *ma*". The word "*aticcamarZwwi*(suppressed)" got split into *"aticcaM"*and *"arZwwi"* which is meaningless. This problem is also due to the lack of diverse patterns in training data. Another cause of errors are rules employed in Sandhi Splitter for inducing morpho-phonemic changes after split. Though the system correctly identified "n" as split point for "*AIDSnulYYa(which is for AIDS)*", but when the rules got applied, this bacame "*AIDSnZ*"+ "*ulla*", where it should have been simply "*n*" which represents a dative case suffix. Whereas "*AIDSnZ*" in which it is meaningless.

The errors in sandhi splitting will eventually effect the performance of Parts-Of-Speech Tagger in

two ways along with sole errors created by POS tagger. Which means 12% of errors from Sandhi Splitter has been propagated to POS Tagger, along with 8% of errors from POS Tagger. Since it is in a pipeline, wrongly split tokens that POS tagger get have unknown patterns which make the system unable to predict the tag accurately and subsequently, this will affect the word level context as well. Though such cases are very rare, one example would be "*rAjAvAN*" (is king) got split into "*rAjA*" and "*AN*", where it should have been "*rAjAv*" and "*AN*". Here "*rAjA*" got tagged as JJ and "*AN*(is)" ideally a verb but got tagged as NN, since the previous word got tagged as JJ.

Errors from both Sandhi splitter and POS Tagger effect the performance of Chunker. 20% of errors together from POS Tagger and Chunker have been propagated to Chunker. A chunk is tagged as incorrect when the words and number of words along with their respective POS tags are not correct. Many instances have 2 or more words per chunk and the chunk tag is decided based on POS tags of words. Since it is in a pipeline, two types of errors can propagate,

- Errors due to unidentified or wrongly identified words from Sandhi splitter.
- Errors from POS tagger, which was effected or unaffected by the errors from sandhi splitter.

There are many instances where snadhi splitter could not identify individual words from a token like "*arYivilla*(no knowledge)". Ideally "*arYiv* and "*illa*", where the first word is a Noun and the other is a Verb. Hence there should be a noun chunk(NP) and a verb chunk(VGF). Since individual words are not available, POS tags and chunk tags will be wrongly identified. Similar would be the case of wrongly identified words.

6.6 Summary

Shallow Parser pipeline for Malayalam has been explained in this chapter. Pipeline includes 4 modules; tokeniser, Sandhi Splitter, POS tagger and Chunker. Results of Individual modules and pipeline have been presented along with the method of pipeline evaluation. Results from the pipeline experiment show the need of a highly accurate Sandhi Splitter. Error propagation due to the the performance of Sandhi Splitter is very high when compared to other modules. Accuracy of the POS Tagger, came down to 79% from 90% due to the errors caused by Sandhi Splitter and further this brought down the accuracy of chunker to 77% from 92%.

Chapter 7

Conclusions

In this thesis we have discussed about experiments conducted to create a Shallow Parser for Malayalam. *Sandhi* is the main bottleneck to process Malayalam computationally. That problem has been addressed by creating a hybrid sandhi splitter which identifies the split point at character level using Naive Bayes classifier and uses hand-crafted character level rules to induce morpho-phonemic changes. Overall accuracy of the system is 87%. This experiment proved that there exists a character level pattern for identifying the split point.

When it comes to Parts-Of-Speech Tagger, CRF has been used as the main classifier. Out of various feature templates used, 3 feature templates tell us more about the nature of the language. When the feature was only word level contexts, accuracy is 79%, whereas when word internal features like prefix and suffix information were added, accuracy improved to 91%. In the third main experiment, the feature was only prefix and suffix information and the accuracy we got is 89%. These experiments validate that Malayalam has more dependency over word internal features like suffix and prefix information than word external information like word contexts in deciding the grammatical category of a word.

Chunker for Malayalam has been built using CRF. Results where high when the feature is 2 words before and after current word along with their POS tags. Experiments showed that reducing or increasing the number of words in context as features reducing the accuracy slightly. Overall accuracy of the best feature template is 94.33%.

Shallow Parsing is a collective process which includes modules in the order which do sandhi splitting, POS tagging and Chunking. Primarily, accuracies of individual modules have been calculated and the respective overall accuracies for Sandhi splitter, POS tagger and Chunker are 88.46%, 90.45% and 92.98%. Pipe line experiments have been conducted to find the rate of error propagation where the output of previous module will be the input for next module. In pipeline experiment, respective over-

all accuracies for Sandhi splitter, POS tagger and Chunker have been dropped to 88.46%, 79.87% and 71.38%. This experiment showed a drop from overall accuracy of 92% to 71%.

7.1 Future Work

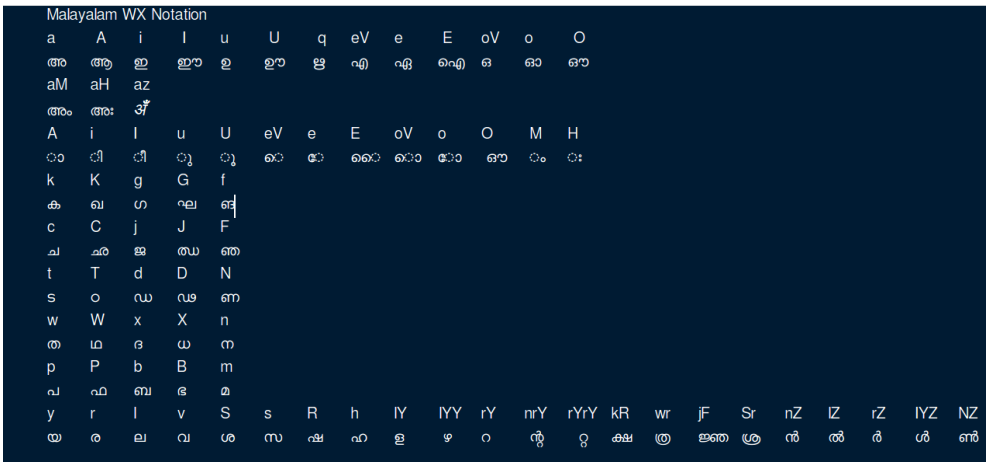
Scope and directions to elaborate this work further are as follows

- Development of a fully statistical Sandhi Splitter.
- Development of algorithms to reduce the error propagation.
- Experiments with Deep Learning techniques for creating, Sandhi Splitter, POS Tagger and Chunker.
- Creation of Full Parser for Malayalam.

Appendix

Wx Transliteration Scheme for Malayalam

Given below is the Wx transliteration scheme[1] for Malayalam.



[1] https://en.wikipedia.org/wiki/WX$_n$otation

BIS Parts-Of-Speech Tagset for Malayalam

No	Category			label	Tag
	Top level	Subtype (level 1)	Sub type (level 2)		
1	Noun			N	N
1.1		Common		NN	N_NN
1.2		Proper		NNP	N_NNP
1.3		Nloc		NST	N_NST
2	Pronoun			PR	PR
		Personal		PRP	PR_PRP
		Reflexive		PRF	PR_PRF
		Relative		PRL	PR_PRL
		Reciprocal		PRC	PR_PRC
		Wh-word		PRQ	PR_PRQ
4	Verb			V	V
4.1		Main		VM	V_VM
4.1.1			Finite	VM	V_VM_VF
4.1.2			Non-Finite	VNF	V_VM_VF
4.1.3			Infinitive	VINF	V_VM_VINF
5	Adjective			JJ	JJ
6	Adverb			RB	RB
7	Post position			PSP	
8	Conjunctions			CC	CC

8.1		co-ordinator		CCD	CC_CCD
8.2		Subordinator		CCS	CC_CCS
8.2.1			Quotative	UT	CC_CCS_UT
9	Particles			RP	RP
9.1		Default		RPD	RP_RPD
9.2		Classifier		CL	RP_CL
9.3		Interjection		INJ	RP_INJ
9.4		Intensifier		INTF	RP_INTF
10	Quantifiers			QT	QT
10.1		General		QTF	QT_QTF
10.2		Cardinals		QTC	QT_QTC
10.3		Ordinals		QTO	QT_QTO
11	Residuals			RD	RD
11.1		Foreign word		RDF	RD_RDF
11.2		Symbol		SYM	RD_SYM
11.3		Punctuation		PUNC	RD_PUNC
11.4		Unknown		UNK	RD_UNK
11.5		Echo words		ECH	RD_ECH

Chunk Tagset for Malayalam

Sl.No	Chunk Type	Tag Name	Example
1	Noun Chunk	NP	[nalla/JJ kutti/NN]NP good child
2.1	Finite Verb Chunk	VGF	[Poyirunnu/VF]VGF have gone
2.2	Non-Finite Verb Chunk	VGNF	[PoyAlZ/VNF wanne/RP]VGNF even if gone
2.3	Infinitival Verb Chunk	VGINF	[pokAnZ/VINF]VGINF to go
3	Adjectival Chunk	JJP	[KuttikalYZ/NN]NP [XarAlYYaM/QTF]JJP Many children
4	Adverbial Chunk	RBP	[VegaM/RB]RBP [poyi/VF]VGF went fast
5	Conjuncts	CCP	[poyi/VF]VGF [eVnnAlZ/CCS]CCP if gone
6	Chunk Fragments	FRAGP	[Amma/NN]NP [(/SYM rAXa/NN)/SYM]NP [yuteV/PSP]FRAGP mother(Radha)'s
7	Miscellaneous	BLK	[NjAnZ/PRP]NP [poyi/VF]VGF [./PUNC] BLK I went .
8	Negation Chunk	NEGP	[allAweV/NEG]NEGP not being

Related Publications

- **Devadath V V**, Litton J Kurisinkel, Dipti Misra Sharma, Vasudeva Varma, **A Sandhi splitter for Malayalam** in *ICON* , Goa , India, December 2014.
- **Devadath V V**, Dipti Misra Sharma **Significance of an Accurate Sandhi-Splitter in Shallow Parsing of Dravidian Languages** in *ACL-SRW*, Berlin, Germany, August 2016.

Bibliography

[1] StevenP. Abney. Parsing by chunks. In RobertC. Berwick, StevenP. Abney, and Carol Tenny, editors, *Principle-Based Parsing*, volume 44 of *Studies in Linguistics and Philosophy*, pages 257–278. Springer Netherlands, 1992.

[2] Manish Agarwal, Rahul Goutam, Ashish Jain, Sruthilaya Reddy Kesidi, Prudhvi Kosaraju, Shashikant Muktyar, Bharat Ambati, and Rajeev Sangal. Comparative analysis of the performance of crf, hmm and maxent for part-of-speech tagging, chunking and named entity recognition for a morphologically rich language. *Proc. of Pacific Association For Computational Lingustics*, pages 3–6, 2011.

[3] R Amritavalli. Separating tense and finiteness: anchoring in dravidian. *Natural Language & Linguistic Theory*, 32(1):283–306, 2014.

[4] PJ Antony, Santhanu P Mohan, and KP Soman. Svm based part of speech tagger for malayalam. In *Recent Trends in Information, Telecommunication and Computing (ITC), 2010 International Conference on*, pages 339–341. IEEE, 2010.

[5] Leonardo Badino. Chinese text word-segmentation considering semantic links among sentences. In *INTERSPEECH*, 2004.

[6] Akshar Bharati. Anncorra: Annotating corpora guidelines for pos and chunk annotation for indian languages. 2006.

[7] Akshar Bharati, Vineet Chaitanya, Rajeev Sangal, and KV Ramakrishnamacharyulu. *Natural language processing: a Paninian perspective*.

[8] S. Buchholz and W. Daelemans. Complex answers: A case study using a www question answering system. *Nat. Lang. Eng.*, 7(4):301–323, December 2001.

[9] Michael John Collins. A new statistical parser based on bigram lexical dependencies. In *Proceedings of the 34th annual meeting on Association for Computational Linguistics*, pages 184–191. Association for Computational Linguistics, 1996.

[10] Walter Daelemans, Sabine Buchholz, and Jorn Veenstra. Memory-based shallow parsing. *arXiv preprint cs/9906005*, 1999.

[11] Aniket Dalal, Kumar Nagaraj, Uma Sawant, and Sandeep Shelke. Hindi part-of-speech tagging and chunking: A maximum entropy approach. 2006.

[12] Aniket Dalal, Kumar Nagaraj, Sandeep Shelke, and Pushpak Bhattacharyya. Building feature rich pos tagger for morphologically rich languages: Experience in hindi.

[13] Sandipan Dandapat, Sudeshna Sarkar, and Anupam Basu. Automatic part-of-speech tagging for bengali: An approach for morphologically rich languages in a poor resource scenario. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, ACL '07, pages 221–224, Stroudsburg, PA, USA, 2007. Association for Computational Linguistics.

[14] Divya Das, Radhika K T, Rajeev R R, and Raghu Raj. Hybrid sandhi-splitter for malayalam using unicode. In *In proceedings of National Seminar on Relevance of Malayalam in Information Technology*, 2012.

[15] Hervé Déjean. Learning rules and their exceptions. *The Journal of Machine Learning Research*, 2:669–693, 2002.

[16] V Dhanalakshmi, G Shivapratap, and Rajendran S Soman Kp. Tamil pos tagging using linear programming. 2009.

[17] Quang Thang Dinh, Hong Phuong Le, Thi Minh Huyen Nguyen, Cam Tu Nguyen, Mathias Rossignol, Xuan Luong Vu, et al. Word segmentation of vietnamese texts: a comparison of approaches. In *6th international conference on Language Resources and Evaluation-LREC 2008*, 2008.

[18] Braja Gopal Patra1 Khumbar Debbarma Dipankar and Das3 Sivaji Bandyopadhyay. Part of speech (pos) tagger for kokborok. In *24th International Conference on Computational Linguistics*, page 923, 2012.

[19] A. Ekbal and S. Bandyopadhyay. Part of speech tagging in bengali using support vector machine. In *Information Technology, 2008. ICIT '08. International Conference on*, pages 106–111, Dec 2008.

[20] Navneet Garg, Vishal Goyal, and Suman Preet. Rule based hindi part of speech tagger. 2012.

[21] Huet Gérard. Lexicon-directed segmentation and tagging of sanskrit. Citeseer.

[22] James Hammerton, Miles Osborne, Susan Armstrong, and Walter Daelemans. Introduction to special issue on machine learning approaches to shallow parsing. *The Journal of Machine Learning Research*, 2:551–558, 2002.

[23] Jisha P Jayan and RR Rajeev. Parts of speech tagger and chunker for malayalam: Statistical approach. *Computer Engineering and Intelligent Systems*, 2(2):68–78, 2011.

[24] D. Jurafsky and J.H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Prentice Hall series in artificial intelligence. Pearson Prentice Hall, 2009.

[25] Dan Klein, Joseph Smarr, Huy Nguyen, and Christopher D Manning. Named entity recognition with character-level models. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 180–183. Association for Computational Linguistics, 2003.

[26] Dinesh Kumar and Gurpreet Singh Josan. Part of speech taggers for morphologically rich indian languages: a survey. *International Journal of Computer Applications (0975–8887) Volume*, pages 1–9, 2010.

[27] Sachin Kumar. Sandhi splitter and analyzer for sanskrit. Master's thesis, Special Center for Sanskrit Studies, Jawaharlal Nehru University, New Delhi, 2007.

[28] Prathyusha Kuncham, Kovida Nelakuditi, Sneha Nallani, and Radhika Mamidi. Statistical sandhi splitter for agglutinative languages. In *Computational Linguistics and Intelligent Text Processing*, pages 164–172. Springer, 2015.

[29] John Lafferty, Andrew McCallum, and Fernando CN Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. 2001.

[30] K Manju, S Soumya, and Sumam Mary Idicula. Development of a pos tagger for malayalam-an experience. In *Advances in Recent Technologies in Communication and Computing, 2009. ARTCom'09. International Conference on*, pages 709–713. IEEE, 2009.

[31] Thomas McFadden and Sandhya Sundaresan. Finiteness in south asian languages: an introduction. *Natural Language & Linguistic Theory*, 32(1):1–27, 2014.

[32] AG Menon. Malayalam. descriptive grammers series. *Indo-Iranian Journal*, 42(4):382–387, 1999.

[33] Vipul Mittal. Automatic sanskrit segmentizer using finite state transducers. In *Proceedings of the ACL 2010 Student Research Workshop*, pages 85–90. Association for Computational Linguistics, 2010.

[34] Antonio Molina and Ferran Pla. Shallow parsing using specialized hmms. *J. Mach. Learn. Res.*, 2:595–613, March 2002.

[35] FrankHenrik Mller and Tylman Ule. On the nature, annotation and use of shallow parsing structures. In Lea Cyrus, Hendrik Feddes, Frank Schumacher, and Petra Steiner, editors, *Sprache zwischen Theorie und Technologie / Language between Theory and Technology*, Sprachwissenschaft, pages 199–209. Deutscher Universittsverlag, 2003.

[36] Latha R Nair and S David Peter. Development of a rule based learning system for splitting compound words in malayalam language. In *Recent Advances in Intelligent Computational Systems (RAICS), 2011 IEEE*, pages 751–755. IEEE, 2011.

[37] Ravi Sankar S Nair. *A Grammar Of Malayalam.* Languages In India, 2012.

[38] Abhiram Natarajan and Eugene Charniak. Statistical samdhi splitting. 2011.

[39] Irina Nikolaeva. *Finiteness: Theoretical and empirical foundations*. Oxford University Press, 2007.

[40] Kemal Oflazer, Elvan Göçmen, Elvan Gocmen, and Cem Bozsahin. An outline of turkish morphology. 1994.

[41] Chirag Patel and Karthik Gali. Part-of-speech tagging for gujarati using conditional random fields. 2008.

[42] Alan Prince and Paul Smolensky. *Optimality Theory: Constraint interaction in generative grammar*. John Wiley & Sons, 2008.

[43] Shambhavi B R and Ramakanth Kumar P. Article: Kannada part-of-speech tagging with probabilistic classifiers. *International Journal of Computer Applications*, 48(17):26–30, June 2012. Full text available.

[44] Shambhavi.b. R, Ramakanth Kumar P, and Revanth G. Article: A maximum entropy approach to kannada part of speech tagging. *International Journal of Computer Applications*, 41(13):9–12, March 2012. Full text available.

[45] Lawrence R Rabiner and Biing-Hwang Juang. An introduction to hidden markov models. *ASSP Magazine, IEEE*, 3(1):4–16, 1986.

[46] RR Rajeev, Jisha P Jayan, and Elizabeth Serly. Tagging malayalam text with parts of speech-tnt and svm tagger comparison. 2010.

[47] Lance A Ramshaw and Mitchell P Marcus. Text chunking using transformation-based learning. *arXiv preprint cmp-lg/9505040*, 1995.

[48] Adwait Ratnaparkhi. *Maximum entropy models for natural language ambiguity resolution*. PhD thesis, University of Pennsylvania, 1998.

[49] Erik F Sang. Memory-based shallow parsing. *The Journal of Machine Learning Research*, 2:559–594, 2002.

[50] Paul Schachter and Timothy Shopen. Parts-of-speech systems. In Timothy Shopen, editor, *Language Typology and Syntactic Description*, volume 1, pages 1–60. Cambridge University Press, second edition, 2007. Cambridge Books Online.

[51] Fei Sha and Fernando Pereira. Shallow parsing with conditional random fields. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 134–141. Association for Computational Linguistics, 2003.

[52] Manish Shrivastava and Pushpak Bhattacharyya. Hindi pos tagger using naive stemming: harnessing morphological information without extensive linguistic knowledge. In *Proceedings of the ICON on Main conference*, 2007.

[53] Akshay Singh, Sushma Bendre, and Rajeev Sangal. Hmm based chunker for hindi. In *Proceedings of International Joint Conference on Natural Language Processing*. Association for Computational Linguistics, 2005.

[54] Anil Kumar Singh. A mechanism to provide language-encoding support and an nlp friendly.

[55] Smriti Singh, Om P Damani, and Vaijayanthi M Sarma. Noun group and verb group identification for hindi. In *Proceedingd of COLING*. Citeseer, 2012.

[56] Smriti Singh, Kuhoo Gupta, Manish Shrivastava, and Pushpak Bhattacharyya. Morphological richness offsets resource demand-experiences in constructing a pos tagger for hindi. In *Proceedings of the COLING/ACL on Main conference poster sessions*, pages 779–786. Association for Computational Linguistics, 2006.

[57] A.R.R. Varma. *Keralapanineeyam 9Th/Ed.* DC Books, 2007.

[58] W. Wahlster. *Verbmobil: Foundations of Speech-to-Speech Translation*. Artificial Intelligence. Springer Berlin Heidelberg, 2013.

[59] Casey Whitelaw and Jon Patrick. Named entity recognition using a character-based probabilistic approach. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 196–199. Association for Computational Linguistics, 2003.

[60] Wikipedia. Languages of india — wikipedia, the free encyclopedia, 2015. [Online; accessed 17-December-2015].

[61] Wikipedia. Part of speech — wikipedia, the free encyclopedia, 2015. [Online; accessed 8-July-2015].

[62] Tong Zhang, Fred Damerau, and David Johnson. Text chunking based on a generalization of winnow. *The Journal of Machine Learning Research*, 2:615–637, 2002.