DESIGN AND DEVELOPMENT OF AN INTEGRATED FRAMEWORK FOR PRONOMINAL ANAPHORA RESOLUTION IN MALAYALAM

Thesis submitted to

Cochin University of Science and Technology

in partial fulfillment of the requirements
for the award of the degree of

DOCTOR OF PHILOSOPHY

by

Ajees A P
Department of Computer Science
Cochin University of Science and Technology
Kochi - 682022

September 2019

Design and Development of an Integrated Framework for Pronominal Anaphora Resolution in Malayalam

PhD thesis in the field of Natural Language Processing

Author

Ajees A P
Department of Computer Science
Cochin University of Science and Technology
Kochi - 22
ajeesap87@gmail.com

Research Supervisor

Prof. Sumam Mary Idicula
Department of Computer Science
Cochin University of Science and Technology
Kochi - 22
sumam@cusat.ac.in

Design and Development of an Integrated Framework for Pronominal Anaphora Resolution in Malayalam

PhD thesis in the field of Natural Language Processing

Author

Ajees A P
Department of Computer Science
Cochin University of Science and Technology
Kochi - 22
ajeesap87@gmail.com

Research Supervisor

Prof. Sumam Mary Idicula
Department of Computer Science
Cochin University of Science and Technology
Kochi - 22
sumam@cusat.ac.in

CERTIFICATE

Certified that the work presented in this thesis is a bonafide research work done by Mr. Ajees A P under my guidance in the Department of Computer Science, Cochin University of Science and Technology, Kochi, India - 682022, and has not been included in any other thesis submitted previously for the award of any degree.

Kochi-22
September, 2019

Prof. Sumam Mary Idicula
(Supervising Guide)

CERTIFICATE

This is to certify that all the relevant corrections and modifications suggested by the audience during the Pre-Synopsis Seminar and recommendations by the Doctoral Committee of the candidate have been incorporated in the thesis entitled “Design and Development of an Integrated Framework for Pronominal Anaphora Resolution in Malayalam” by Mr. Ajees A P.

Kochi-22
September, 2019

Prof. Sumam Mary Idicula.
(Supervising Guide)

DECLARATION

I hereby declare that the work presented in this thesis is based on the original research work done by me under the guidance of Prof. Sumam Mary Idicula, Department of Computer Science, Cochin University of Science and Technology, Kochi, India - 682022, and has not been included in any other thesis submitted previously for the award of any degree.

Kochi-22

September, 2019 Ajees A P.

Acknowledgements

All praise and thanks to God Almighty for all the blessings showered on me from time to time.

First and foremost, I owe my profound sense of respect to my supervisor and guide, Dr. Sumam Mary Idicula, for her patience, enthusiasm, immense knowledge and motivating nature and endless support, which paved the way for successful completion of my doctoral dissertation. I have been extremely fortunate to have a supervisor who cared so much about my work, and I cherish the rich experience of working with her.

I am extremely thankful to Dr. G Santhosh Kumar, Head of the Department, Department of Computer Science, Cochin University of Science and Technology for providing me with the facility, support and encouragement to pursue the Ph D work in the department. My special thanks to K B Muralidharan, Assistant Professor, Cochin University of Science and Technology for the constructive comments and meaningful interactions throughout the work. I gratefully acknowledge all the teaching and non-teaching staff of Department of Computer Science, CUSAT who have been very forthcoming to offer advice and help in their respective roles.

I would like to extend my sincere gratitude to Prof. M Sreenathan, Department of Linguistics, Thunchath Ezhuthachan Malayalam University, for

providing the linguistic expertise required for this study. Along with the faculty, I would also like to thank his Ph D student, K J Abrar from the same University for his contributions towards this thesis.

Throughout this journey, I have had the privilege of interacting with my colleagues from the same lab. In particular, I remember with gratitude the timely support and encouragement from Ms. Namitha K, Ms. Sreela S R, Mr. Graham G, Mr. Anees V, Ms. Manju K, Ms. Jayasree, Ms. Athira M and Mr. Anil Raj. Apart from the exciting research discussions that we had, it was a pleasure for me to discuss with them many other social and personal matters. I am also grateful to all my friends from the department, who are not mentioned above, for their support and prayers.

I am deeply indebted to my parents A P Pareed and M M Fathima, who always have been a source of inspiration for the last three decades of my academic life. Without their unconditional support and prayers, I could never have reached this point. I would also like to thank my brother, sister and their families for their endless support to pursue my dreams.

Finally and most importantly, words are too short to express my deep sense of gratitude towards my beloved wife Marsana M Shereef and my loving children Faiha Mariyam and Aman Ahsan for their understanding, encouragement, patience and unwavering love. I vouch that this journey would not have been possible without their priceless and perpetual support, invaluable help and inspiration.

Ajees A P.

Abstract

Language is the way in which humans make utterances and sounds that can be understandable to the members of the human community. The functions of language include communication, the expression of identity, emotional release, and imaginative expression. It can be expressed in two forms; spoken and written forms. The spoken form of the language is generated using articulate sounds, while the written form of the language is generated using a circumscribed set of symbols. The amount of electronic text data is increasing day by day. Lot of texts and images are added to the Internet every second. But this text is stored in an unstructured way. Extracting the relevant information from this unstructured data is a demanding task that invites the focus of NLP researchers.

Anaphora resolution is one of the potential problems in natural language processing. It is the process of identifying the antecedent of an anaphoric expression in a natural language text. Most of the NLP applications such as information, text summarization, machine translation, question answering, etc. require successful resolution of anaphors. In this thesis, a methodology for the resolution of pronominal anaphors present in Malayalam text is proposed. Important preprocessing tools required for the resolution of anaphors were also developed as part of the research work and the development details are discussed in detail. Resolving the correct antecedent of anaphoric expressions is a laborious task which demands the clever utilization of NLP techniques.

The pronominal anaphora Resolution system proposed in this thesis comprises of different phases like POS tagging, Named Entity Recognition, Deep level tagging and Anaphora Resolution. The POS tagging phase experimented three techniques. One is using Conditional Random Fields, another is using neural networks and the third one is using deep neural networks. Different features like word-level features, character level features and affix level features were considered in the study. The word-level features were extracted using pre-trained word embeddings, while the affix-level and character-level features were identified using feature inferring neural networks and handcrafted rules.

The named entities present in the POS tagged text were identified in the second phase. The POS tag information obtained from the POS tagging phase was used as additional information (feature) towards entity extraction. Here also, different techniques like neural networks, CRFs and deep neural networks were experimented to generate the entity tagging model. Effect of various features like word-level, sub-word-level, character-level and POS tag information on the performance of the entity recognition task was analyzed. The model with maximum performance was selected as the entity tagging module of the overall system.

Although the entity extraction phase gave information about the named entities present in the text document, semantic information like gender, number and person (PNG information) were not available at that phase. The deep level tagging phase, which identified the number and gender information of person entities, helped in this regard. In order to extract the number and gen-

der information of person entities, various classification techniques like Naive Bayes, kNN, SVM, Random forest, MLP, etc. were tried. A combination of morphological features was applied to improve the classification accuracy. The classifier with maximum performance constituted the deep level tagging model for the architecture.

Finally, the anaphora resolution phase identified the actual antecedents corresponding to each pronominal anaphor present in the text. The deep level tagged text from the deep level tagging phase was given as the input of the anaphora resolution phase. Here the major focus was on two things; extraction of feasible candidates and selection of the most probable candidate corresponding to each anaphor. The feasible candidates were shortlisted based on the number and gender agreement with the anaphor. The selection of the maximum possible candidate was carried out using an algorithm which effectively utilized the linguistic properties of Malayalam language. The performance of each phase of the overall system was measured using various evaluation metrics.

Contents

List of Figures

List of Tables

Abbreviations

NLP	Natural Language Processing
IE	Information Extraction
POS	Parts of Speech
SVO	Subject Verb Object
ISCII	Indian Standard Code for Information Interchange
ASCII	American Standard Code for Information Interchange
ML	Machine Learning
NER	Named Entity Recognition
HMM	Hidden Markov Model
SVM	Support Vector Machine
CRF	Conditional Random Field
MXENTK	Maximum Entropy Model Toolkit
CIIL	Central Institute of Indian Languages
PNG	Person Number Gender
TnT	TrigramsnTags
MBLP	Memory Based Language Processing
MUC	Message Understanding Conference
MEMM	Maximum Entropy Markov Model
CoNLL	Conference on Computational Natural Language Learning
BIO	Begin Inside Outside
WCL	Word Codebook Learning
DNN	Deep neural network
PMI	Pointwise Mutual Information
FIRE	Forum for Information Retrieval Evaluation
TAM	Tense Aspect Modality
k-NN	k-Nearest Neighbors
MLP	Multi Layer Perceptron
OCR	Optical Character Recognition

OOV	Out Of Vocabulary
BIS	Bureau of Indian Standards
FNR	False negative rate
FPR	False positive rate
ANN	Artificial Neural Networks
ReLU	Rectified Linear Unit
API	Application Program Interface
SGD	Stochastic Gradient Descent
CBOW	Continuous Bag of words
CNN	Convolutional neural network
LSTM	Long Short-Term Memory
BiLSTM	Bidirectional Long Short-Term Memory
RNN	Recurrent Neural Network
NLTK	Natural Language Toolkit
IECSIL	Information Extraction for Conversational Systems in Indian Languages
CE	Common Era
PCA	Principal component analysis
ROC	Receiver operating characteristics
SVM	Support vector machine
TNR	True negative rate
TPR	True positive rate

1

Introduction

1.1 overview

Language is the conventional method of communication between the members of the human community. It is a set of symbols or signs arranged in a particular order to express the message. The origin of language goes back to the beginning of human species on earth. Various languages are available over the world, which can express thoughts on an endless number of topics. Every language consists of tens of thousands of words which can be used to construct an infinite number of phrases and sentences. Those sentences can be used not just to convey the information but to express our feelings. There are two forms in which languages can be expressed- spoken and written forms. The spoken form of the language is generated using articulate sounds. While the written form of the language is generated using a set of signs and symbols. The spoken form of the language is generated using a set of signs or symbols. In the spoken form of the language, much of the meaning is indicated by the context of the speaker. But in the written form of the language, most of the context should be explicitly provided by the text. Similarly, the spoken form of the language

prefers to deliver subjective information, whereas the written form tends to transmit objective information. Moreover, written texts can be communicated across time, which is not possible in the case of the spoken form of the language. According to Henry sweet, ”Language may be defined as the expression of thought by means of speech sounds” [1]. Each language has its own dialect, character set, syntax, phonetics, and morphology. Within a language itself, there are varieties in dialects based on geographical distance or social factors. The number of consonants and vowels may vary from language to language. For example, the language English has 24 consonants and 18 vowels (depending on the dialect). While Rotokas, a Bougainville language has only five vowels and six consonants [2]. Syntax, the set of rules which defines the combination of words in a language also changes from language to language. For example, in English, we say ”there is a white ball”, whereas, in French, the structure would be ”il y a une ballon blanc”. Here the position of adjectives in both the sentences is different. Furthermore, there are differences among languages in the number of meanings for words. Languages like English are more concise in nature, while those like Arabic may have tens of meanings.

The amount of written form of the language is increasing day to day. Lot of texts and images are added to the web every second. But this text is stored in an unstructured way. Mining the important information from this unstructured data is a challenging task that invites the focus of NLP researchers. Information extraction, a subdomain of artificial intelligence, handles this challenge. Unstructured text can be converted into a structured form using IE techniques. Each of our work presented in this

thesis comes under the domain of information extraction. All of them can be effectively utilized for the computational analysis of natural language text. In our study, we have considered Malayalam as the major focus of attention, since it is the native and official language of Kerala. We were also motivated to solve the problem of resource-scarcity in Indian languages to a particular extent.

Anaphora resolution is one of the old problems in natural language processing. It is the process of identifying the true referent of an anaphoric expression in a natural language text. Most of the NLP applications such as text summarization, machine translation, question answering, etc. require successful resolution of anaphors. In this thesis, we propose a methodology for the successful resolution of anaphors present in Malayalam text. The series of preprocessing tools required for the resolution of anaphors are also developed and discussed in detail. Anaphora is a Greek word that originated in the 16th century [3]. It is comprised of 'ana' and 'phora', where 'ana' means back, and 'phora' means carrying. Hence anaphora stands for the act of carrying back. Anaphors help to avoid repetition of words in natural language text. Resolving the correct antecedent of anaphoric expressions is a laborious task which requires the clever utilization of NLP techniques.

Most of the studies in anaphora resolution are based on documents in English, French, Chinese, Persian, Arabic and related languages. Using the distinctive characteristics of Indian languages, the works on anaphora resolution are yet to be explored. In most of the Indian languages, anaphora resolution is in its in-

fancy stage. Malayalam, a morphologically rich resource-poor language, deserves its own merits in all the language processing tasks. Detailed analysis of elementary units (words) in a text document for the resolution of pronominal anaphors in Malayalam is a novel approach. Along with an algorithm for anaphora resolution, a set of preprocessing tools such as POS tagger, named entity tagger, and deep level tagger were also developed for the computational analysis of Malayalam language.

There is a huge amount of text data available online in Malayalam. Malayalam Wikipedia itself contains more than 30,000 articles. This warrants us to develop tools that can be used to explore digital information present in Malayalam and other native languages. Anaphora resolution is one such tool that helps in determining the coherence of text segments. It is also a necessity in term-based summarization, where the score of a sentence is calculated by adding the weights of words present in it. Resolving the pronouns to their actual antecedents help in measuring the importance of a sentence using statistical techniques. Automatic text classification is another application which finds the importance of anaphora resolution. The quality of text classification shows significant variation between the texts that are pronoun resolved and non-pronoun resolved. Furthermore, anaphora resolution also finds its applications in areas such as question answering, machine translation, information retrieval, etc.

1.2 Characteristics of Malayalam language.

Malayalam is one of the 22 scheduled languages in India and bears the status of official language in Kerala, Lakshadweep, and Puducherry [4]. More than 38 million people speak it over the world. In Kerala and Lakshadweep, Malayalam is used in government, commerce, education systems, and in mass communication. It belongs to the family of Dravidian languages. Even though Malayalam is closely related to Tamil, its origin is more influenced by Sanskrit, the language of Vedas. The influence of Sanskrit is clear throughout the development of Malayalam language. 'Lilatilakam', the first book of Malayalam grammar was written using Sanskrit script. The first record of Malayalam language is an inscription dated to 831 BC [5]. Vattezhuthu, which contains an alphabet set of 30, was the early script used to write Malayalam [6]. Another script of Malayalam called 'kolezhuthu' was derived from 'vattezhuthu'. However, Brahmi script, a descendant of both the above scripts is the current script used in Malayalam documents. Malayalam script has the most number of letters among Dravidian languages [7].

Malayalam is characterized by its morphological richness and agglutinative nature. Morphology can be defined as the study of the formation of words from smaller meaning-bearing units called morphemes. Morphemes are of two kinds- stems and affixes. Stems carry the basic meaning of word and affixes append additional meaning to stems. Affixes can be either prefix, infix or suffix depending upon the stem. The morphological richness of Malayalam allows its creation of complex words which are rich in information. It is not equally promi-

nent in different Indian languages. Some words in Malayalam contains meaning which can only be expressed by four or more English words. For example, the word 'cheyyendiyirunnilla'in Malayalam can only be expressed by using 5 English words as 'should not have been done'. Another characteristic of Malayalam language is its inflectional and derivational morphology. Inflectional morphology deals with the generation of variants of a word without changing its word category. While derivational morphology results change in the word category. For example, a noun may change to verb, adjective or adverb. The languages that doesn't permit inflection are called analytic languages (ex: Chinese) [8]. New words in Malayalam are originated through inflection and derivation.

Most of the western languages have a specific structure for their sentences. For example, the sentences in English and German always follows SVO (Subject, Verb, and Object) pattern. However, Malayalam does not follow such common patterns and is a free word order language. Based on the grammatical category of words, Malayalam words can be mainly classified into two, namely- Vachakam and Dyothakam. Word categories like nouns, verbs and adjectives are included in Vachakam, whereas word categories like prepositions, conjunctions and interjections are included in Dyothakam.

1.3 Challenges in Malayalam language processing

There are many challenges that exist in interpreting Indic languages, including Malayalam. Indic languages use alphabets

derived from Brahmi scripts instead of Latin alphabets. Brahmi is a syllabic alphabet, where each sign can be a simple consonant or a syllable with the consonant and the inherent vowel [9]. Lack of standard publicly available resources is the major challenge towards Indic language computing. These resources include wordnets, tagged datasets, parallel corpus, dictionaries, etc. Moreover, the available datasets are very small in size as compared to western languages. When it comes to Malayalam, the different challenges in Malayalam computing are as follows.

1. Agglutinative and inflective nature of the language:

   The primary challenge in Malayalam computing is its inflective and agglutinative nature. Two or more words can be united to form another single word based on a set of predefined rules. Similarly, the root of a word can be inflected to create a new word based on a set of rules. These rules may vary from languages to language. Both the above features of the language pause a big challenge in Malayalam language computing.

2. Lack of morphological analyzer:

   Morphological analysis is the process of analyzing individual words into their components. Even though various attempts are made by different researchers in this field, Malayalam does not have a full-fledged system for morphological analysis.

3. Lack of labelled datasets:

   Labelled datasets are the most critical resources in building machine learning/statistical models. Statistical/ ma-

chine learning models are the best performing tools in various fields of NLP, such as POS tagging, entity extraction, sentiment analysis, etc. Building the tagged dataset itself is a laborious task which demands language expertise.

4. Lack of standard encoding:

   Unlike most of the Europian languages, Indian languages do not follow standard encoding format. Different websites are using different encoding formats. Even though Unicode is widely accepted among the NLP community, some of the available Malayalam digital content is still available in ISCII (Indian Standard Code for Information Interchange) format. ISCII is an 8-bit encoding scheme specific to Indian scripts and an extension of 7-bit ASCII. But Unicode is a variable bit encoding scheme which is accepted worldwide and more dynamic.

5. Constrained wordnet:

   Wordnet is the collection of lexical items in any language. It contains definitions, synonyms, antonyms, and usage examples for each word in a language. Malayalam does not have a full-fledged wordnet available online.

6. Lack of online dictionaries:

   Dictionaries are electronic resources that record the words in a language along with its meaning and pronunciation. Attempts to build a complete online Malayalam dictionary are ongoing at Malayalam University, Tirur. Availability of online dictionaries is essential in word sense disambiguation problems.

7. Slow content creation:

English dominates most of the web data when compared with other languages. Nevertheless, recently, there is a growth in digital Malayalam literature. Difficulty in inputting Malayalam through the keyboards is one of the reasons behind this limited growth. Lack of spell checkers, grammar checkers, thesaurus, etc. are also reasons for this dilemma.

1.4 Problem statement

From the research gaps identified, the problem can be formulated as ”Developing an integrated framework for the resolution of pronominal anaphora in Malayalam, which uses machine learning as well as rule-based techniques”. The complete system consists of machine learning as well as rule-based modules. Hence, we call it an integrated system for anaphora resolution.

1.5 Objectives

Resolution of pronominal anaphors present in Malayalam documents using machine learning as well as rule-based techniques is an interesting NLP problem with direct applications in different tasks like question answering, text summarization, machine translation, etc. From the literature, it has been observed that anaphora resolution can only be accomplished after a series of pre-processing steps. Since Malayalam does not have publicly available standard pre-processing tools, tools like POS tagger, named entity tagger and deep level tagger were also developed

as part of this work. To achieve our final goal, the objectives of this research work were identified as follows.

1. Construct and validate a POS tagged corpus for Malayalam language.
2. Develop an efficient mechanism for POS tagging in Malayalam.
3. Compare the proposed algorithms with the state-of-the-art algorithms and techniques employed for POS tagging.
4. Construct and validate a Named Entity tagged corpus for Malayalam language.
5. Develop an efficient mechanism for NE tagging in Malayalam.
6. Compare the proposed algorithms with the state-of-the-art algorithms and techniques employed for Named Entity tagging.
7. Develop an efficient mechanism for Deep level Tagging in Malayalam
8. Develop an efficient mechanism for pronominal anaphora resolution in Malayalam and validate the same.

1.6 Contributions of the Thesis

The major contributions of this thesis to Malayalam NLP can be summarized as given below.

- A POS tagged corpus for the Malayalam language was constructed, evaluated and made publicly available. Each word in the corpus was labeled with Bureau of Indian Standard tag set, which consists of 36 tags corresponding to different categories. The corpus was evaluated using Fleiss Kappa co-efficient, which measured the inter-rater agreement. This corpus constituted the first publicly available dataset for POS tagging in Malayalam.

- Different schemes for POS tagging in Malayalam were designed, implemented and tested. Each of the schemes involved different phases like pre-processing, feature set construction, training and testing. Impacts of different features on the performance of the developed systems were experimented. The power of word embeddings were also utilized for Malayalam POS tagging. The proposed system beat all the existing POS tagging results in the Malayalam language.

- A named entity tagged corpus for the Malayalam language was constructed and evaluated. Four major types of entities, namely person, location, organization, and miscellaneous, were considered for tagging. Evaluation of the dataset was performed using Fleiss Kappa co-efficient.

- Different schemes for named entity recognition in Malayalam were tried. The effects of various language level features on the performance of entity recognition were experimented. Word embeddings were also utilized for the effective representation of words. The proposed system outperformed all the entity tagging algorithms currently available in Malayalam language.

- A deep level tagging scheme for the Malayalam language was designed, implemented and tested. The developed system generated deep level information about nouns and verbs in a text document. The system consisted of different phases like POS tagging, animate noun identification and deep level tagging. The morphological richness of Malayalam language was effectively utilized for the deep level analysis of nouns and verbs. This approach is a novel thought towards the semantic analysis of natural language text.

- An intelligent framework for the resolution of pronominal anaphors in Malayalam was designed, developed and tested. The proposed framework was evaluated using two small datasets from short stories and news articles domain. This system is a hybrid of machine learning and rule-based approaches for anaphora resolution in Malayalam.

1.7 Outline of the Thesis

The complete research work is presented in eight chapters and the outline of the thesis organization is as follows:

Chapter 1: This chapter discusses the motivation behind the work, states the research problem, enumerates the objectives of the research work and gives an overview of the structure of the thesis. The chapter also discusses the challenges involved in Indian language computing along with a special emphasis on the Malayalam language.

Chapter 2: A systematic survey on existing algorithms used for POS tagging, Named Entity Recognition and Anaphora Resolution in Malayalam along with other Indian as well as non-Indian languages is discussed in this chapter. More emphasis is given to works from Indian languages since they hold similar characteristics to Malayalam.

Chapter 3: This chapter gives an overview of the complete architecture proposed for the resolution of pronominal anaphora present in Malayalam text. A brief explanation of the metrics used to evaluate the performance of each module present in the overall architecture is also given.

Chapter 4: Different algorithms used for Parts of Speech Tagging of Malayalam text are explained here. They include algorithms such as CRF, neural networks and deep neural networks. The results are compared with the state-of-the-art systems for the same. A novel word representation which combines rule-based as well as machine learning techniques for deep learning-based POS tagging is also discussed in detail. The details involved in the construction of the dataset for POS tagging in Malayalam are also given there.

Chapter 5: This chapter presents the application of three different algorithms on Named Entity Recognition for Malayalam. The results obtained are compared with the state-of-the-art systems for NER in Malayalam. An enhanced word representation for deep learning-based NER in Indian languages is proposed and discussed in detail. The details involved in the construction of a publicly available dataset for NER in Malayalam are also given.

Chapter 6: Deep level tagging of Malayalam text is discussed in this chapter. The analysis of nouns includes identifying the number, gender and case details, whereas the analysis of verbs includes determining tense, aspect and modality details associated with them. The linguistic features employed for the tasks are also explained in detail.

Chapter 7: Anaphora Resolution is the process of determining the actual antecedent of an anaphoric expression present in natural language text. Most of the NLP applications such as information extraction, question answering, text summarization, etc. require successful resolution of anaphors. This chapter on Pronominal Anaphora Resolution for Malayalam presents a novel approach towards resolution of pronominal anaphora references present in Malayalam text.

Chapter 8: This chapter concludes the thesis by summarizing the major contributions of the work and discusses the future directions for extending the research work.

2

Literature Survey

This chapter briefly reviews the literature on tasks such as POS tagging, entity recognition, and anaphora resolution. Works in different Europian and Indian languages are reviewed in detail. The main emphasis is given to works in Indian languages since Malayalam is one among the classical Indian languages. POS tagging and Entity extraction in different Indian languages such as Tamil, Telugu, Malayalam, etc. are discussed in detail. Different features adapted for POS tagging and NER in Indian languages are also explored. An overview of different anaphora resolution schemes available in Hindi, Tamil, Telugu, etc. is also presented.

2.1 Introduction

The growth of digital content and the need for information extraction owes greatly to one of the highly challenging problems in NLP called anaphora resolution. Identifying the antecedent of an anaphoric expression is essential for the semantic processing of natural language text. The applications of anaphora resolution can be seen in wide areas such as text summariza-

tion, information extraction, question answering, etc. Additionally, POS tagging is a preprocessing step for most of the NLP applications. However, entity recognition is required in situations where information extraction is the major goal. Most of the high-level applications in Natural Language Processing require a combination of these tools to accomplish the task. To a certain extent, the development and performance of these tools contribute to the growth of technological improvement in the NLP community.

It is evident that the significance of anaphora resolution is increasing day by day. The need for automatic information extraction mainly contributes to this significance. Hence, the number of researchers working in this area is also increasing. There are more than 7000 spoken languages existing in the world today [10]. Many of them have very few numbers of speakers as compared to others. However, each language attributes different threats to the problem of anaphora resolution depending on the language characteristics. Languages belonging to the same family may have some common characteristics. Except that, each language demands a language-specific treatment towards the anaphora resolution problem.

The structure of this chapter is as follows. Section 2.2 discusses the state of the art in POS tagging of Indian and non-Indian languages. Similarly, section 2.3 reviews state of the art in Named Entity Recognition. Section 2.4 presents state of the art in anaphora resolution. Finally, section 2.5 concludes the chapter with an overall summary.

2.2 POS tagging

POS tagging is the process of distinguishing the correct linguistic classification of words based on their meaning and context in a text document. Most of the available works in POS tagging uses rule-based and stochastic techniques. One of the first and most widely used POS tagger in English is Brill's tagger, which used a rule-based algorithm [11]. ENGTWOL tagger, Btoush tagger are also examples of POS taggers that employed rule-based techniques [12, 13]. Rule-based taggers contain a two-phase architecture where the first stage makes use of a dictionary to assign potential tags for each word and the second phase utilizes a set of hand-written rules to disambiguate the tag. 'ENGTWOL' tagger is a well-known example of two-stage architecture. HMM-based POS taggers employ stochastic methods for tagging [14–17]. One of the disadvantages of such methods is the need for large sized training data. Stochastic methods consider the frequency count of elementary units (words) and their associated tags in the training data to assess the probability of each tag for each word. For example, HMM taggers pick the tag of a word based on the following equation.

$$P(tag|word) = P(word|tag) * P(tag|previous\ n\ tags) \quad (2.1)$$

Works such as [18–20] are examples of HMM-based tagging algorithms. Along with HMM and its variants [21, 22], transformation based learning [23], maximum entropy model [24, 25], memory based learning, Decision trees [22], support vector machines [26], etc. were also experimented for English language. For some languages such as German and English, the POS tagging accuracy has almost reached 98% [27]. Hunpose, a reim-

plementation of Trigrams'n'Tags (TnT) tagger works well for morphologically rich languages. For languages such as English and Hungarian, it works with an accuracy of 96.58% and 98.24% respectively.

Indian languages are far behind in POS tagging accuracy as compared with other languages. Lack of language resources like tagged corpus, guidelines, language technology tools, etc. are the various reasons behind this problem. Among the Indian languages, the largest number of works in POS tagging are reported from Hindi and Tamil. The first work in Hindi was reported in 2006 by Smrithi Singh [28]. Her tagger was based on morphological rules without any tag disambiguation process. A tagged corpus of 15562 words along with a decision tree based learning algorithm was used to carry out the tagging process. The reported accuracy of the tagger was 93.45%. The second work was based on a maximum entropy model by Aniket Dalal and Kumar Nagaraj. They extracted feature functions from the training data. A feature function is a boolean function which catches some aspect of the language relevant to sequence labelling task. They achieved an overall accuracy of 88.4%. CRFs, probabilistic graphical models for sequence labelling was experimented by Himanshu and Amni Anirudh [29]. Hindi morph analyzer was used in their study to produce root words. Different features like word length, suffixes and special characters were used for training with the help of CRF++. Training was conducted on a corpus of size 150000 and reported accuracy is 82.67%. The fourth work in Hindi was an HMM-based methodology which utilized the morphological features of the language [14]. This work tried to minimize the

number of unique types encountered during learning. Manish Srivastava and Pushpak Bhattacharya used a naive stemmer as a pre-processor to the HMM-based tagger [30]. They got considerable improvement in performance above all the other works.

Bengali is an Indo-Aryan language primarily spoken by the people of Bangladesh. First work in Bengali POS tagging was reported in 2007 [15]. A set of supervised, semi-supervised and maximum entropy based models were developed in the first attempt. Training was performed on a corpus of 40000 words over a tagset of 30 tags. It was observed that the supervised models outperform all the other models. The second work was based on conditional random fields over a tagset of 26 tags, where the feature selection played a vital role in its performance [31]. This method utilized contextual information along with other word-level features. Training was performed on a corpus of size 72000 with an accuracy of 90.3%. The third reported work on Bengali was based on SVM [32]. The CRF based corpus used in the previous work was used as the training data and achieved an accuracy of 86.4%. Another work in Bengali is reported in 2010 by Hammad Ali [33]. His tagger was based on beam-welch trained HMM approach. The system consisted of a four-layer architecture with each layer dedicated to specific tasks. The first level assigned words to one of the 12 universal categories. A set of disambiguation rules along with detailed morphological information was used in the second level. The third and fourth level will perform local word grouping and multiword verb tagging. Last work in Bengali (to our knowledge) was reported by Kamal Sarkar of Jadavpur University [34]. He developed a tri-gram based POS tagger for Bengali

by constructing the bi-gram based POS tagger as a baseline.

POS tagging in Telugu has reached a much better position as compared with other south Indian languages. Mainly three works are reported in Telugu. First work was based on transformation based learning, which recorded an accuracy of 96% on test data [35]. Second work was based on a rule-based approach which made use of different functional modules such as tokenizer, morph analyzer, morph to POS translator, POS disambiguator, unigram and bigram rules and annotator [36]. Tokenizer converted input text into tokens and morph-analyzer produced morphological information about these tokens. Morph to POS translator used this morphological information to tag each word in the text. POS disambiguator was used to resolve the tag ambiguity problem. Unigram and bigram rules were used to control the ambiguity in the POS tagger. Finally, annotator produced tagged words. The system reported an accuracy of 98% on a limited tagset. The third work in Telugu [37] was based on a maximum entropy-based model implemented using publicly available maximum entropy model toolkit [MXENTK]. This system reported an accuracy of 81.78%.

Tamil, the official language of Singapore and Srilanka stands second in terms of the number of works in Indian languages. The first work in Tamil was reported by Vasu Ranganathan in 2001 [38]. His work was based on the lexical, phonological approach and named as 'tagtamil'. Morphological features of verbs were processed using an index method. Tagtamil was capable of performing tagging and generation. The second work was reported by Ganeshan, who utilized the CIIL corpus [39].

A portion of CIIL corpus was tagged using a dictionary and morphological analyzer. Later he trained the system until maximum precision was reached. Another work was reported by Kathambam, who made use of heuristic rules for POS tagging [40]. No morphological analyzer was used in this work. A set of 12 heuristic rules were used, and tagging was performed using PNG, tense and case markings. Unknown words were handled using a bigram approach. Selvam and Natarajan employed statistical techniques for Tamil POS tagging [41]. According to them, the derivational and inflectional word forms can't be handled by rule-based techniques. They were able to make some improvements over the existing systems and reported an accuracy of 85%. Rajendran and a team of Amritha University used a linear programming based SVM approach towards POS tagging [42]. They have developed their tagset of 32 tags and tagged a corpus of 35000 sentences. The system reported good performance improvements over the existing systems.

Parts of speech tagging in Malayalam is not a new area of research. However, only a few numbers of works are reported until now. The first work in Malayalam POS tagging was reported by Manju K and Soumya S in 2009 [43]. They used a stochastic HMM-based methodology for tagging. The major bottleneck of their work was the limited size of the training corpus. Only 1400 words were used for training. Another work using TnT was reported in the same year by Rajeev et al. [44]. They used a training corpus of 15,245 words and tested using a corpus of 200 words. The system reported an overall accuracy of 90.5% on IIIT-Hyderabad tagset. The third work was reported in 2010 by Amritha University, Coimbatore

[45]. A training corpus of 100000 words tagged using a tagset of 29 tags (their own) was used for training. SVM was chosen as the training algorithm. The fourth work was reported by Robert Jesaraj in 2013 [46]. He used a memory-based language processing (MBLP) approach, which utilized the power of efficient storage of solved examples and similarity-based reasoning. A hybrid approach for POS tagging in Malayalam was reported in 2014 [47]. A combination of rule-based and statistical approaches were employed in this approach. The rule-based module made use of lexical analyzer, morphological analyzer and syntax analyzer to tag the words in the input text. The statistical module made use of a CRF based tagger to tag the remaining words. Another hybrid approach for POS tagging in Malayalam was reported in 2015 by Noorul Mubarak and Sareesh Madhu [48]. The proposed approach made use of dictionary entries along with context tag information for generating the tag sequence. Another reported work in Malayalam POS tagging was by Sachin et al. using an EPIC framework [49]. The framework used conditional random fields to generate the tagging model. The training data contained text from different domains, such as health, tourism, news articles, etc.

2.3 Named Entity Recognition

Named entity recognition (NER) is an important task in natural language processing. NER has made a lot of progress in the past 20 years. Even though different solutions are reported for the problem, NER still remains as an unsolved problem. The named entity is a word or phrase which clearly isolate an item

from a set of items having similar attributes. NER can be defined as the problem of locating a word or phrase that refers to a particular entity within a text. The term named entity is coined for the first time in the sixth message understanding conference (MUC-6) [50]. Typically, three kinds of approaches were applied towards NER, namely- supervised, semi-supervised and unsupervised. We will go through each of this approach in detail.

2.3.1 Supervised methods

Supervised methods make use of the paradigm called learning through examples. Different supervised learning algorithms such as HMM, SVM, decision trees, maximum entropy models, conditional random fields, etc. were employed for the problem. Typically, the supervised learning algorithms either learn the parameters of supposed distribution that maximizes the likelihood or learn the disambiguation rules. HMM, a generative model for sequence labelling was the earliest model applied for NER by Bikel et al. in 1999 [51]. He named his system as 'identifier', which assigns the desired label to each word in a document. According to him, each word can be assigned with a single label in any context. The label can be either one of the desired class or none of the desired class (not an entity). Hidden Markov Models try to generate the data, given the sequences of words and labels from the parameter distribution. Viterbi algorithm is used to find the best label sequence from the entire space of label sequences. In 2002, [52] used an HMM-based NER system for English and reported 96.6% accuracy on

MUC-6 data. They employed different orthographic features, trigger words and words from gazetteers to identify the named entities. A comparative study between HMM and MEMM has been performed by Malouf [53]. He experimented the impact of different features like capitalization, word position, etc. on NER. His system reported an F-score of 73.66% on Spanish CoNLL 2002 data set.

Unlike hidden Markov models, maximum entropy models are discriminative models for sequence labelling. Here the model tries to learn the weight for discriminative features from the given set of features and training data. Each feature is connected with a parameter λ_i. Maximum entropy models try to maximize entropy, which in turn ensures that for every feature q_i, the expected value of q_i will be equal to the empirical expectation of q_i in the training data. Similar to the HMM model, the Viterbi algorithm was used to find the best possible label sequence. The MENE system from Borthwick is an example of a maximum entropy-based NER model [54]. He experimented his methodology on MUC-7 datasets and got an accuracy of 88.8%. Similarly, Carran and Clerk also applied a maximum entropy-based model to the problem of NER [55]. They employed a softmax technique to formulate the probability and achieved an overall accuracy of 84.89% on CoNLL-2003 data.

Mcname and Mayfield were the first to use support vector machines for entity recognition [56]. They have considered 8-tags for the task including person, location, organization and miscellaneous. Eight binary classifiers were used for training. Features like language-related features, orthographic features

and punctuation features were experimented. Each word was tagged with the set of all possible tags. If that set is empty 'other' tag was assigned else the most frequent tag was assigned. They could achieve an accuracy of 60.97% for the Spanish language on CoNLL-2002 dataset.

CRFs were introduced to the problem of NER by McCallum and Li in 2003 [57]. They proposed a feature induction based statistical modelling technique to the problem of NER. The conditional probability of a state sequence given an input sequence is calculated as given below.

$$P(s \mid o) = \frac{1}{Z}\exp\left(\sum_{t=1}^{T}\sum_{t=1}^{T}\lambda_k f_k(s_{t-1}, s_t, o, t)\right) \tag{2.2}$$

Here 'Z' is called the normalization factor and F_k is an arbitrary feature function. λ_k is the weight factor for each feature function. The state transition probabilities were learned using a dynamic programming approach. And the most likely tag sequence was identified using the Viterbi algorithm. They were able to obtain an overall accuracy of 84.04% for the English language on CoNLL-2003 dataset.

2.3.2 Semisupervised methods

Semi-supervised learning algorithms make use of a small amount of labelled data set and create more hypothesis by utilizing a lot of unlabelled data. The major motivation behind the semi-supervised type approach is the lack of enough labelled data, and data sparsity problem. A small amount of labelled data, a

large amount of unlabeled data, and an initial hypothesis are the pre-requisites for semi-supervised learning algorithms. More amount of annotated data are generated iteratively until a pre-defined threshold occurs. The AdaBoost algorithm developed by Carreras et al. in 2002 was the first example of such an approach in NER [58]. They used three binary classifiers for the labelling task, each corresponding to each label. BIO tagging scheme was used to label the data. Semantic and orthographic features of words were evaluated over a shifting window through simple binary propositional features. They were able to achieve a performance of 79.28% on CoNLL-2002 Spanish corpus.

Liao et al. [59] proposed a semi-supervised learning approach using conditional random fields. They could exploit the evidence that is self-sustained from the features used for a classifier. They were able to achieve an improvement of 12% in recall as compared with the supervised classifiers at that time. Mishra et al. introduced a semi-supervised approach for entity recognition in noisy-text [60]. They used linear-chain conditional random fields for up-sampling the training data. Features like word clusters, pre-trained distributed word representations, updated gazetteer features, etc. were used for ingesting the meaning of words. Kuksa et al. [61] introduced a novel semi-supervised technique called Word-Codebook Learning (WCL) and they used it for biomedical named entity recognition. WCL try to learn a class of word-level feature embeddings to capture the semantic meaning of words from a large un-labelled corpus. They could achieve state-of-the-art performance in bio-medical NER.

2.3.3 Unsupervised methods

The most costly thing in building the supervised models is the preparation of annotated data. Identifying the robust set of features is also a burden when dealing with supervised algorithms. Moreover, many resource-poor languages like Malayalam, Tamil, etc. don't have a large set of annotated corpus available for experimentation. These factors led to the development of unsupervised methods for entity extraction. KNOWITALL, a domain-independent system for NER was the first one reported in this manner [62]. Here the candidate facts are generated using eight domain independent extraction patterns. These candidate facts are evaluated using pointwise mutual information (PMI) computed using huge web content. Later, a probability was assigned to each extracted fact, which eventually helps to manage the trade-off between precision and recall. Munro and Manning proposed another work on Unsupervised NER in 2012 [63]. They have developed a language-independent cross-domain entity recognition system that generated seed candidates via local, cross-language edit likelihood. No external resources were used for the system. Parallel text that may or may not be aligned was the only resource employed for the problem. However, they could obtain an F-score of 85% on purely unsupervised named entity recognition across languages [64].

2.3.4 NER in Malayalam

NER in Malayalam is not a novel area of research. But only a few numbers of works were reported until now. Lack of

standard data sets, pre-processing tools, the complexity of the language, etc. are the various reasons for this dilemma. Among the Indian languages, Hindi and Tamil are the leading ones in NER. The first work in Malayalam NER was reported by Bindu et al. in 2012 [50]. They employed a hybrid approach for entity extraction. A combination of linguistic principles and statistical methods were employed in their study. [65] reported a statistical approach using Trigrams'n'Tags. TnT is an open source statistical tagger which can be used for any morphologically rich language. The only limitation to their work was the limited size of training data. In 2014, a comparative study on the performance of different entity tagging algorithms was conducted by Amrita University, Coimbatore [66]. Conditional Random Field (CRF) was used for English and Support Vector Machine (SVM) for other languages. They were able to achieve comparable performance in Indian languages. A combination of TnT and Maximum Entropy Markov Model (MEMM) were tested by Shruthi et al. in 2016 [67]. They too faced the problem of limited sized training data, which unveils the bottleneck of their work. Another reported work on Malayalam NER is from Remmiya Devi et al. in 2016 [68]. They employed skip-gram based word embedding features for the identification of named entities. The tagged social media text shared as part of FIRE-2015 was used for training. They were able to obtain an F-score of 81.55% on test data.

2.4 Anaphora Resolution

Anaphora Resolution is the process of identifying the correct antecedent of an anaphoric expression present in natural language text. It affects the performance of most of the NLP applications, including text summarization, machine translation, information retrieval, etc. It is the presence of anaphors, which makes the natural language text interesting to read. Even though different solutions were reported for the problem of anaphora resolution, it remains an unsolved problem to a significant extent. Different researchers across the world have analyzed anaphors at various levels such as syntactic, semantic and discourse levels. The proposed approaches vary from traditional rule-based and syntactic methods to modern statistical and semantic methods. Works in anaphora resolution started in the early nineteen fifties. Depending on the linkage between phrases and sentences in a paragraph, anaphors can be mainly classified into three. They are

- pronominal anaphora
- Definite noun phrase anaphora
- One anaphora

pronominal anaphora Pronominal anaphora is the most commonly used one in natural language texts [69]. They are realized using pronouns, short words that refer to some entity mentioned in the prior discourse. Pronouns include personal pronouns, possessive pronouns, reflexive pronouns, demonstrative pronouns

and relative pronouns. All of them need not be anaphoric.

Example: Raju went to collect *his* mark list.

Definite noun phrase anaphora Here the antecedents are referred by a definite noun phrase which either points to the same concept or semantically close concept.

Example: Donald Trump visited India. *The American president* utilized his opportunity.

One anaphora In one anaphora, the anaphoric expressions within the text are realized using the keyword 'one'.

Example: There are six flowers on the table. But, I like the *one* with red colour.

2.4.1 Anaphora resolution in non-Indian languages

Anaphora resolution gained attention with the work of Hobbs in 1976 [70]. He developed a parse tree based searching algorithm where the noun phrase upon which the search terminates was considered as the probable antecedent of the pronoun. There onwards many knowledge-intensive approaches, including Carter (1979), Carbonell and Brown (1988), Rich and Luperfloy (1988), etc. were also reported [71–73]. All these approaches were able to make use of syntactic, semantic and world knowledge information contained in the text documents. Centering theory, a theory related to focus of attention based

on discourse knowledge, was reported by Grosz (1977), Joshi (1979) and Strube and Hahn (1999) [74–76]. Such methods were intended to identify the centre of attention (who is being talked about) in a discourse. Centres are the semantic entities that contribute to the discourse model. Further thoughts towards anaphora resolution were identifying the salience features for antecedents. Appropriate weights were assigned to each feature, and the antecedent with the highest salience weight is selected as the actual antecedent. Works from Lappin and Leass (1994), Kennedy Boguraev (1996), and Sobha et al. (2000) belongs to this category [77–79].

A two-engine approach by Mitkov was successful to a particular extent in anaphora resolution [80]. It was based on the interactivity of two engines, where the first engine incorporated the constraints and preferences and the second engine followed the principles of uncertainty reasoning. Later (1997) he also presented an indicator based resolution method, where the possible candidates were assigned scores by a set of indicators and the candidate with the highest indicator score is selected as the actual antecedent [81]. The system was able to achieve a success rate of 89.7%, which was better than the comparable systems at that time. Machine learning was introduced into this field with the work of Dagan Hai (1990) [82]. He employed an unsupervised approach based on co-occurrence patterns in a large corpus. These statistics reflected the semantic constraints and were used to disambiguate the antecedents. Dagan Itai's work was followed by many supervised and unsupervised approaches including Aone and Bennet (1995), McCarty and Lahnert (1995), Ng and Cardia (2002), Daelman and Van De Bosh

(2005), Hendrickx (2008), Recasen (2009), etc. Different algorithms such as Decision Tree, CRFs, Expectation Maximization, etc. were successfully demonstrated for the task.

2.4.2 Anaphora Resolution in Indian languages

Most of the Indian languages are morphologically rich in nature and are verb-ending languages. They are also well known for their free word ordering property. Indian languages come under Indo-Aryan, Dravidian, and Tibeto-Burman families of languages. Among them, Malayalam belongs to the family of Dravidian languages. Dravidian languages have rich, productive suffixation and are more agglutinative in nature. Nouns are affixed with number, gender and case markers. Similarly, tense, aspect and mood markers are affixed with verbs. But, in languages like Hindi (Indo-Aryan) case markers take place as post positions to the nouns. There it should be handled in the preprocessing stage to understand the proper morphology of nouns. When it comes to the distinction in number and gender of pronouns, most of the Indian languages agree in number distinction. However, the scenario is different in gender distinction. Some of the languages like Sanskrit, Malayalam, Kannada, etc. allows gender distinction in their pronouns whereas languages like Hindi, Gujarathi, Assamese, Punjabi, etc. do not provide any linguistic clue for the classification of gender in their pronouns [83].

In Indian languages, works in anaphora resolution were reported only from a limited set. This includes languages such as Hindi, Tamil, Malayalam and Bengali. This situation was con-

stituted by the scarcity of resources like preprocessing tools, standard datasets and parsers. Vasisth, a rule-based system for anaphora resolution was the earliest work reported by Sobha and Patnaik (1998) [79]. They proposed a rule-based system for anaphora resolution by exploiting the morphological richness of languages such as Malayalam and Hindi. It was a language-independent system for anaphora resolution, which was primarily focussed on Hindi and Malayalam. Later, centering theory-based approaches were reported by Prasad and Strube (2000) [76], Upalapu et al. (2009) [84] and Dekwale et al. (2013). According to them, it was the grammatical role and not the word order that determines salience in a discourse. Hobb's algorithm was attempted by Dutta et al. in 2008 [85]. They had assessed the roles of subject and object towards the resolution of reflexive and possessive pronouns in the Hindi language. A comparative study on the performance of Tamil anaphora resolution using multi-linear regression and salience score based approach was presented by Murthy et al. in 2007. [86].

CRF was applied for anaphora resolution by Akilandeswari et al. in 2013 [87]. CRFs were trained on linguistically motivated features to boost the performance of the system. They were able to obtain an average accuracy of 64.83% across texts from different Tamil novels. Balaji et al. presented a two-stage bootstrapping approach for the resolution of anaphors in Tamil text. The first stage of his architecture identified anaphora and its possible candidates. While the second stage dealt with the resolution of anaphors. Another work for the same language (Tamil) was reported by Ram et al. in 2013 [88]. They used tree CRF for the resolution of anaphors by utilizing the fea-

tures from dependency parsed text. GUITAR, a general tool for anaphora resolution, was customized by Senapati et al. in 2013 [89]. They fine-tuned the system for Bengali anaphora resolution. Similarly, BART tool was customized by Sikdar et al. for Bengali anaphora resolution [90]. Both the above works were inspired by the works in ICON-2011.

Similar to SemEval 2010, a tool contest for multilingual anaphora resolution was conducted as part of ICON-2011. The contest had three languages including Hindi, Tamil and Bengali. Four teams participated in the contest. All four participants were able to submit their results for Bengali, while Tamil and Hindi had two participants. All the submitted systems included a language dependent module specific to each language. Later, Sobha et al. came up with a generic anaphora resolution engine for Indian languages [83]. The system was free from any language-specific modules. The PNG information associated with nouns was identified using in-depth morphological analysis and PNG agreement heuristic rules. The possible candidates having the PNG agreement was passed through a CRF module which identified the correct antecedent of the anaphor.

2.4.3 Anaphora Resolution in Malayalam

Malayalam is a morphologically rich agglutinative language, where the syntactic and semantic roles played by the nominal expressions are expressed by the case suffixes and postpositions rather than by word order. It is a verb-final language that allows scrambling. Most commonly, head nouns are preceded by adjectives, participial adjectives and free relatives. Postposi-

tions carry semantic information about the association between nouns and verbs. Sometimes they function as case endings with a usual occurrence after nominals. Prepositions and prefixes are unattested in the language. Case suffixes in nouns influence the grammatical relation between words in a sentence which is a common feature to all Dravidian languages. The first reported work in Malayalam was Vasisth, which was a multilingual system for anaphora resolution primarily focussed on Malayalam and Hindi [79]. It used limited parsing where the parser required limited information such as POS, clause identification and subject of the clauses. The system could resolve all the referentially dependent elements. The second and the next reported work in Malayalam Anaphora Resolution was from Athira et al. in 2014 [91]. They proposed an algorithm to resolve the pronominal anaphora in Malayalam text using a hybrid approach. The system computed the salience value for each possible candidate and returned the one with the highest salience score. The system was evaluated using precision and recall measures with satisfactory results.

2.5 Summary of the chapter

Literature survey has enabled us to survey a wide variety of techniques available for various NLP tasks in low-resource languages. In Indian languages, the non availability of efficient computational tools such as POS tagger, NE tagger, etc. have adversely affected NLP operations. Complex nature of the languages is also a problem in most of the cases. Among the Indian languages, Hindi stands first in terms of the number of

works and available linguistic tools. The most number of public datasets for various language processing tasks are also available in Hindi. Tamil takes second place in this context.

Various techniques available for POS tagging in different Indian languages are studied in detail. Also, studies were carried out in Europian languages like English and French. The shift of technology from rule-based systems to deep learning based technology is also observed. Currently, there are very few deep learning based systems available for POS tagging in Dravidian languages. In the preliminary analysis, various mechanisms applied to improve the accuracy of POS tagging were explored. Later, the focus was shifted to the effective representation of words in agglutinative languages. Combinations of different word-level and contextual level features were carried out to improve the POS tagging accuracy in various languages. Studies show that the features that are effective in one family of languages may not be effective in another family of languages. From this, we realized that the features must be selected based on the characteristic features of individual languages.

In named entity recognition, the survey was more focussed on the recent advancements in entity extraction. In the earlier days of entity recognition, more emphasis was given to the methodology opted to solve the problem. Later, it was found that the focus is shifted to the effective representation of words concerning each language. With the introduction of deep learning technology, the performance of entity extraction systems increased predominantly. The need for huge datasets was also increased since the deep- learning-based systems demand large

datasets to train and test. Studies were also carried out to investigate the impact of deep learning technology in Indian language NER. It was found that the introduction of word embedding and deep learning based technologies not only improved the performance in Europian languages but also changed the scenario in Indian languages. Our observation reveals the point that ”still, there is a lot to do with Indian language NER”.

Studies were also carried out to analyse the progress of anaphora resolution in Indian languages along with other family of languages. In Indian languages, only very few works were reported in anaphora resolution and that too in limited datasets. Most of the works were rule-based and language-specific, utilizing the morphological features of the language. From the literature survey, we understand that the task of anaphora resolution in Indian languages is still in its infancy.

3

System Architecture

This chapter describes the architecture of the proposed system. The main goal of this research work is to resolve the pronominal anaphors present in Malayalam text. Different words that help in resolving the antecedents of anaphors are analyzed in detail. In order to form the list of possible candidates for each anaphor, a detailed study regarding the nouns in preceding sentences is also conducted. A combination of rule-based and machine learning based algorithms constitutes the complete solution.

Section 3.1 of this chapter gives a brief introduction about the methodology opted to solve the problem. The necessity of such a system for anaphora resolution in Malayalam is also discussed here. Section 3.2 presents a detailed architecture of the proposed system. Furthermore, each module included in the overall architecture is discussed in detail. Section 3.3 briefs about the metrics used to evaluate the performance of the proposed system. Finally, the chapter is concluded in section 3.4, with an overall summary of the chapter.

3.1 Introduction

The methodology proposed for the resolution of pronominal anaphors is addressed in this chapter. Since there is no publicly available system for anaphora resolution in Malayalam, the need for such a system is one of the demanding areas of research in the NLP community. In an anaphora resolution (AR) system, identifying and selecting the proper antecedent for each anaphoric expression is a challenging task. The appropriate candidate for each anaphor depends on many factors, including the type of anaphor, nature of the text, the context of the anaphor, etc. The focus of this research is on the resolution of pronominal anaphors (the most frequent type of anaphors) present in Malayalam text.

Anaphora resolution is the process of finding the correct antecedent of an anaphor from the set of all possible antecedents. It is a complex problem in NLP, which is yet to be explored in its required dimension. Many definitions have been given to the problem by different researchers in the field. According to Halliday and Hassan, anaphora is ”the cohesion which points back to some previous item” [92]. As reported by Hirst, ”anaphora is a device for making an abbreviated reference to some entity in the expectation that the receiver of the discourse will be able to dis-abbreviate the reference and, thereby, determine the identity of the entity” [93]. AR is required in information extraction systems to summarize a document or to find the answer to a question. A large amount of text data are available online in Malayalam. Proper understanding of such data demands anaphora resolution. Hence, anaphora resolution is an area of

research holding social relevance.

Natural languages across the globe may not be similar. The structure and syntax of one language may not be the same as that of the other language. Moreover, anaphora resolution is a difficult task that demands proficiency in various domains of language processing, such as syntax analysis, semantic analysis, discourse analysis, etc. In this study, the resolution of pronominal anaphors present in Malayalam text document is considered. As far as Malayalam is concerned, the key challenges associated with anaphora resolution are

- Lack of standard preprocessing tools
- Lack of standard datasets
- Inflectionally rich nature of the language
- Free word ordering nature of the language
- Influence of case information
- Nonuniform encoding formats, etc.

3.2 System Overview

Figure 3.1 illustrates the general architecture of the proposed system. The main steps involved in the development of the proposed anaphora resolution system are as follows.

- Preprocessing

- POS tagging
- Named Entity Recognition
- Deep level tagging
- Anaphora resolution

In the proposed system, the major objective is to improve the performance of pronominal anaphora resolution system in Malayalam, by incorporating syntactic, semantic and discourse level information. The language level features considered for this task are the POS tag information, entity tag information, number and gender information, case information, etc. The proposed architecture mainly include different subtasks, namely-preprocessing, POS tagging, Named Entity Recognition, Deep level tagging and anaphora resolution.

Preprocessing is the first phase of the architecture, which brings the input text into a form that can be easily handled by the machines (algorithms). The amount of preprocessing required for one task may not be the same for another task. Therefore it can be said that ”text preprocessing is not directly transferable across various tasks”. In the current work, it has been limited to noise removal, sentence segmentation and word tokenization. Noise removal is the process of removing unnecessary symbols/characters, that can interfere with the text data. Sentence segmentation operations carry out the task of separating the sentences and word tokenization carries out the task of separating the words in a text document.

The POS tagging phase involves identifying the grammat-

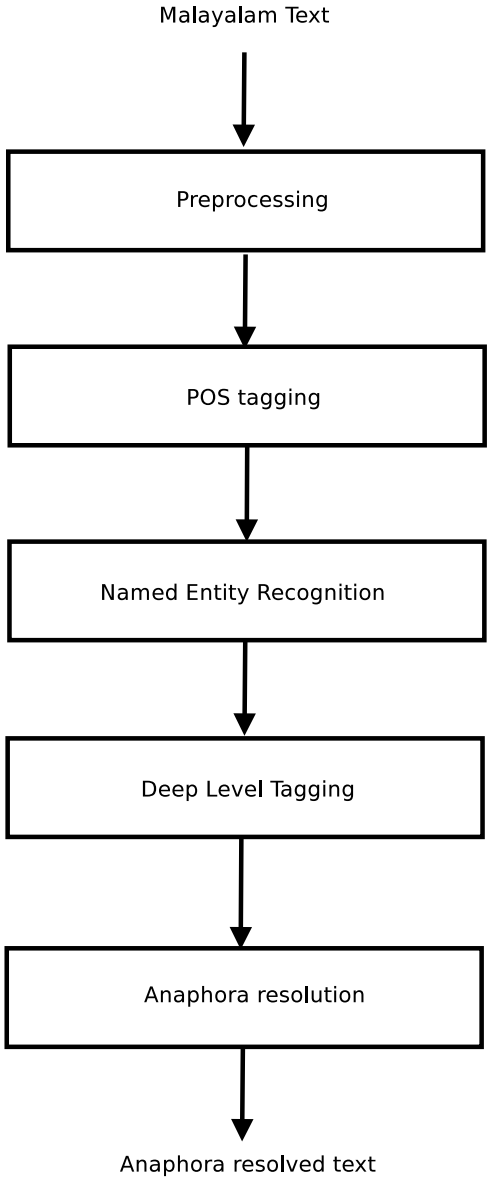


Figure 3.1: General architecture of the proposed system

ical category of each word in a text document. In order to identify the POS tag information for words, approaches such as CRFs, neural networks and deep neural networks were applied. A combination of word-level, subword-level and charac-

ter level features were employed to improve the tagging performance. Tagging model with maximum performance constituted the POS tagging module of the architecture.

The named entity recognition phase recognizes the named entities present in the POS-tagged text. It is the process of identifying the named entity mentions present in a natural language text. The POS tag information carried from the POS tagging phase act as additional information (feature) towards entity extraction. Here also, different techniques like neural networks, CRFs and deep neural networks were used to develop the entity tagger. Effect of various features like word-level, subword-level, character-level and POS tag information on the performance of the entity recognition task was experimented. The model with maximum performance constituted the entity tagging module of the architecture.

Although the entity extraction phase can give information about the named entities present in a text document, no semantic information like Person, Number and Gender is available at that phase. Deep level tagging, which identifies the number and gender information of person entities, can help in this regard. In order to extract the number and gender information of person entities, various classification techniques such as Naive Bayes, kNN, SVM, Random forest, MLP, etc. were employed. A combination of morphological features was applied to improve the classification accuracy. The classifier with maximum performance constituted the deep level tagging module of the architecture.

The anaphora resolution phase identifies the actual antecedents

corresponding to each anaphor present in the text. Deep level tagged text from the previous phase is the input of the anaphora resolution phase. Here the main focus is given to two things, namely-extraction of potential candidates and selecting the best candidate corresponding to each anaphor. The potential candidates are shortlisted based on the number and gender agreement with the anaphor. The selection of the best candidate is carried out using an algorithm which effectively utilizes the linguistic properties of Malayalam language.

3.3 Performance Evaluation

Selection of proper evaluation metrics is very important in analyzing the performance of the proposed system. Various evaluation measures such as accuracy, precision, recall, F-score, ROC-curve, etc. were used for evaluating the performance of various modules of the proposed architecture [94]. Accuracy is the most commonly used evaluation metric to measure the performance of any machine learning model. It is the ratio of the number of accurate predictions to the total number of predictions. However, it fails in situations where there are class imbalance problems in the dataset. The real problem in sticking to the accuracy measure alone is that the rate of misclassification in minor class (class with less training samples) is not well projected. Hence, it was decided to go for other measures such as precision, recall, F-score, ROC curve, etc.

Precision-a measure of exactness gives an indication about the correctness of a classifier. It is the ratio of correct positive

data points to the data points that are classified as positive. Precision gives an idea about the ability of a model to identify only the relevant data points. It is also called as positive predictive value of a classifier/algorithm. It is calculated as given in equation 3.1.

$$Precision = \frac{\text{True positives}}{\text{True positives + False positives}} \tag{3.1}$$

Recall-a measure of completeness in any classification problem. It is the ratio of correct positive data points to the data points that should have been identified as positive. It is also called as sensitivity or true positive rate of a classifier [95]. Recall gives an idea about the ability of a model to find all relevant cases within a set of data points. The recall is calculated as given in equation 3.2.

$$Recall = \frac{\text{True positives}}{\text{True positives + False negatives}} \tag{3.2}$$

F-score, measure used to find the balance between precision and recall gives the correct indication about a model's performance. It is the harmonic mean between precision and recall. F-score can give true information regarding the preciseness and robustness of a machine learning model. It can be used in situations where there are uneven class distributions within the dataset. F-score is calculated as in equation 3.3.

$$F - score = 2 * \frac{\text{Precision*Recall}}{\text{Precision+Recall}} \tag{3.3}$$

ROC curve (Receiver Operating Characteristic curve) is a probability curve used to measure the performance of a machine learning model. It is a plot between the true positive rate and false positive rate of a classifier. Here, FPR represents the X-axis and TPR represents the Y-axis. The area under the ROC curve (AUC-ROC) represents the degree of separability between different classes of a classifier. It is a measure indicating how well we can distinguish between various classes of a classifier. An excellent machine learning model has an AUC-score near to one, whereas a poor model has an AUC-score near to zero.

3.4 Summary of the chapter

This chapter puts forward a novel architecture towards pronominal anaphora resolution in Malayalam. The need for such a system for Malayalam NLP is also detailed. Various modules and their functions in the system architecture are briefly discussed.

4

POS tagging

This chapter describes novel approaches towards parts of speech tagging in Malayalam. Parts of speech tagging is the process of assigning lexical class markers to each and every term in a text document. It is also called as grammatical tagging since it finds the grammatical category of words in a document. Different approaches like HMM, CRF, neural networks, and deep neural networks are experimented. The performance of different tagging algorithms are computed and compared with the existing systems. Different features that could improve the accuracy of tagging are also considered and analyzed. Experiments are conducted to find the impact of word embeddings for the task of POS tagging.

Section 4.1 of this chapter gives a brief introduction about POS tagging. Section 4.2 deals with the dataset preparation for the same. A detailed description of the tagset used and the method of preparation of dataset is also presented here. Section 4.3 of this chapter introduces POS tagging using conditional random fields. It also discusses the basic concepts of conditional random fields. Section 4.4 discusses POS tagging using neural networks. Results obtained by different tagging algorithms on the same data set is also included in this section.

It also includes a comparison of the performance of the proposed system with the existing methods. Section 4.5 proposes a deep learning based POS tagging system for Malayalam. The power of word embeddings for Malayalam POS tagging is also demonstrated in this section. Finally, a brief overview of the chapter is given in section 4.6.

4.1 Introduction

POS tagging is an important task in most of the NLP applications. It is the process of identifying the syntactic role of a word in a phrase or sentence. POS tags can provide linguistic clues about the word within the scope of a sentence. POS tagging plays an important role in various stages of NLP, including syntactic processing, semantic processing, pragmatic processing, etc. They are also useful in distinguishing the sense of a word in a document. Syntactic patterns of words can also be inferred from POS tags. Despite of the several attempts made by different researchers, POS tagging in Malayalam still demands lot of improvement. Resolving the ambiguities of words present in a document is a challenging task [28]. In comparison with the growth of POS tagging research in Europian languages, Malayalam is far behind in terms of the number and quality of works. Lack of standard datasets, pre-processing tools and agglutinative nature of the language are the various reasons for this state.

4.2 Dataset preparation

Lack of standard datasets is the curse of Indian languages. Only very few datasets are publicly available for experimentation. That too in limited quantity. Most of the digital content available in Indian languages do not have a standard encoding format and font. Languages like English had come across a long way in areas like character representation, character display, OCR technology, etc. The focus of research in such languages have shifted towards a semantic angle in the computing world. But for languages like Malayalam, the focus of research is still in standard dataset preparation, online spell checking, OCR technology, morphological word processing, etc. Lack of development and implementation standards is another issue in Indian language computing. BIS tagset is developed by the POS tag standardization committee of the department of information technology (DIT), NewDelhi, India is used in this work.

4.2.1 Data collection

As part of the dataset preparation, we have crawled lot of texts from various online newspapers, story sites and literature. All of them were converted into standard 'utf-8' encoding format. Useless symbols and abnormalities were removed using regular expressions. A set of 28755 sentences were prepared in this way and tagged with BIS tagset using the publicly available IIIT-MK POS tagger. The major drawback associated with the IIIT-MK POS tagger is its inadequate accuracy. It gives only a real-time performance of about 75% in accuracy. Therefore we

decided to go for a POS tagger with improved performance.

4.2.2 Data annotation

The preprocessed tagged text is erroneous in nature since the performance of the publicly available tagger is comparitively less. Hence, it was decided to manually correct the tags. The tagged text was formatted in such a way that each line contain a word and its tag separated by a tab. A total of 287500 words were aligned in this way. All these words were separated into 30 files such that each file contains around 1000 sentences. Each file is distributed among the M. Tech students in our department along with the description of how to identify and change the erroneous tags. The description contains details of BIS tagset along with some real examples in Malayalam. Around one month time is given to the students for manual tag correction. After the manual tag correction phase, each file is collected and integrated into a single document. The description of the BIS tagset is given in table 4.1.

4.2.3 Data Evaluation

For evaluating the authenticity of the dataset, we have used Fleiss Kappa coefficient as an evaluation metric. Fleiss Kappa coefficient is a statistical measure which measures the inter-rater agreement for categorical items [96]. Unlike Cohens Kappa, Fleiss Kappa can be employed in situations where there are more than two raters (labellers). The Fleiss kappa can be calculated as follows. $\bar{P}_e$ and $\bar{P}$ are calculated as given below.

Table 4.1: BIS Tagset and its Description

Tag	Description	Tag	Description
N_NN	Common noun	RB	Adverb
N_NNP	Proper noun	PSP	Postposition
N_NST	Locative noun	CC_CCD	Co-ordinator
V_VM	Main verb	QT_QTC	Cardinals
V_VM_VF	Finite verb	QT_QTO	Ordinals
V_VM_VNF	Non-finite verb	RD_RDF	Foreign words
V_VM_VINF	Infinite verb	RD_SYM	Symbol
V_VN	Verbal noun	RD_PUNC	Punctuation
V_VAUX	Auxiliary verb	RD_UNK	Unknown
JJ	Adjective	RD_ECH	Echo words
DM_DMD	Deictic demonstrative	RP_INTF	Intensifier particle
DM_DMR	Relative demonstrative	RP_NEG	Negation particle
DM_DMQ	Wh-word(Demonstrative)	QT_QTF	General quantifier
PR_PRP	Personal pronoun	CC_CCS	Subordinator
PR_PRF	Reflexive pronoun	CC_CCS_UT	Quotative
PR_PRL	Relative pronoun	RP_RPD	Default particle
PR_PRC	Reciprocal pronoun	RP_CL	Classifier particle
PR_PRQ	Wh-word(Pronoun)	RP_INJ	Interjection particle

$$k = \frac{\bar{P} - \bar{P}_e}{1 - \bar{P}_e} \tag{4.1}$$

where $1 - \bar{P}_e$ gives the degree of agreement that is attainable above chance and $\bar{P} - \bar{P}_e$ gives the degree of the agreement actually obtained above chance.

$$\bar{P} = \frac{1}{N} \sum_{i=1}^{N} P_i \tag{4.2}$$

$$\bar{P}_e = \sum_{j=1}^{k} P_j{}^2 \tag{4.3}$$

$$P_i = \frac{1}{n(n-1)} \sum_{j=1}^{k} n_{ij}(n_{ij} - 1) \tag{4.4}$$

$$P_j = \frac{1}{Nn} \sum_{i=1}^{N} n_{ij} \tag{4.5}$$

Here 'N' is the total number of words that are given in common to all raters and 'K' is the total number of tags considered

Table 4.2: Most common tags and their frequencies

Tag	Frequency
N_NN(Common noun)	112290
V_VM_VNF(Non finite verb)	31204
JJ(Adjective)	19897
V_VM_VF(Finite verbs)	16979
N_NNP(Proper noun)	15071
V_VAUX(Auxiliary verbs)	8891
RB(Adverb)	8364
QT_QTF(Cardinals)	5983
DM_DMD(Demonstrative)	5953
PSP(Postposition)	5039

for the study. 'n' is the total number of raters participated in the manual tagging process. The words are indexed i=1,2,3,...N and tags are indexed j=1,2,....k. We could achieve an overall Kappa score of 0.71, which shows a substantial agreement between the different raters (labellers). Table 4.2 shows the statistics of most frequent tags present in our corpus. The noun is the most frequent tag in the training corpus. On the other hand, 'wh-word' is the least frequent tag.

4.3 POS tagging using Conditional Random Fields

4.3.1 Introduction

Conditional random fields are probabilistic graphical models for sequence labelling. They are often applied in areas like pattern recognition and machine learning. In comparison with a discrete classifier, CRFs are well known for its ability to take context into account. But discrete classifiers predict outputs based on the current sample without considering the neighbour-

ing samples. CRFs are capable of predicting multiple variables that are mutually dependent. In our experiments, we have used linear chain CRF which predict the sequence of tags for the sequence of words. CRF is a type of undirected probabilistic graphical model used to encode known relationships between hidden variables and observations. CRFs are suitable for applications like POS tagging, named entity recognition, gene finding, etc. Graphical representation of CRF is shown in Figure 4.1.

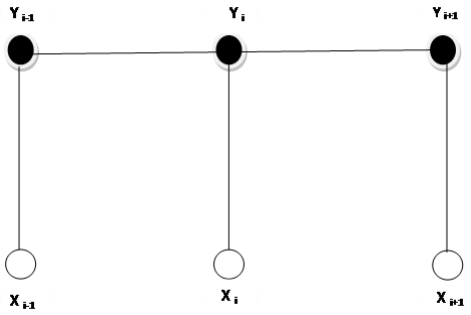


Figure 4.1: Graphical representation of CRF

4.3.2 Architecture

The CRF based system generally consists of a training phase and a testing phase. The architecture of the proposed system is given in Figure 4.2.

The architecture mainly consists of two phases- training phase and testing phase. The first phase is the training phase which takes the tagged text as input and build the model. First of all the tagged text is provided to a preprocessing module which takes the tagged text as input and transforms into a sequence

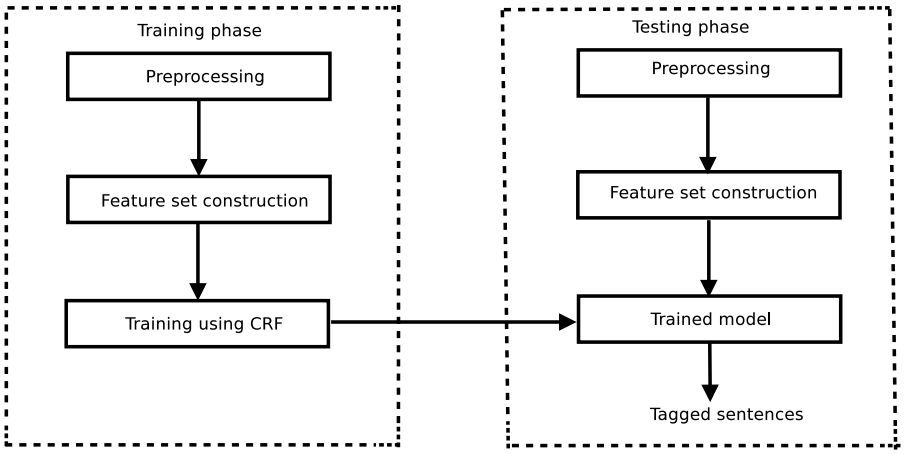


Figure 4.2: General architecture of the proposed system

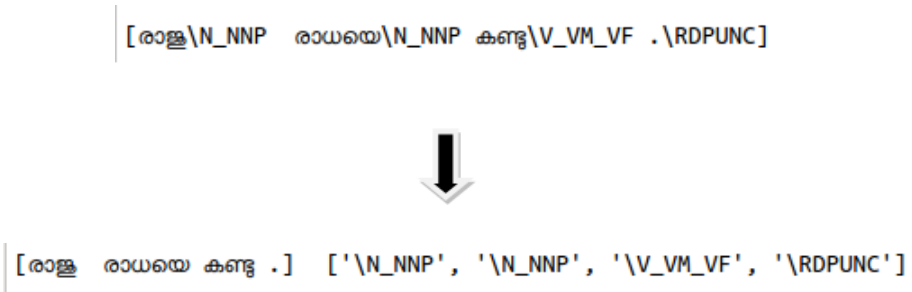


Figure 4.3: Sample text showing the input and output of the preprocessing phase

of words and sequence of tags. A sample text showing the input and output of the preprocessing phase is given in figure 4.3. Each word from the word sequence is sent through a feature preparation module. The feature preparation module replaces each word by a set of features corresponding to that word. Different features we have considered in our study are the word itself, preceding words, succeeding words, suffixes of different length, etc. A sample feature set for a simple sentence from the training data is shown in figure 4.4.

The third module of the first phase is the training module,

['word=ഇന്ത്യ', 'word[-5:]=ന്ത്യ', 'word[-3:]=ത്യ', 'word[-2:]=്യ', 'word.isdigit=False', 'BOS', 'BOS', '+1:word=എന്റെ',
'+2:word=രാജ്യമാണ്'] ['word=എന്റെ', 'word[-5:]=എന്റെ', 'word[-3:]=്റെ', 'word[-2:]=റെ', 'word.isdigit=False', 'BOS',
'-1:word=ഇന്ത്യ', '+1:word=രാജ്യമാണ്', '+2:word=.'] ['word=രാജ്യമാണ്', 'word[-5:]=യമാണ്', 'word[-3:]=ാണ്', 'word[-2:]=ണ്',
'word.isdigit=False', '-2:word=ഇന്ത്യ', '-1:word=എന്റെ', '+1:word=.', 'EOS'] ['word=.', 'word[-5:]=.', 'word[-3:]=.',
'word[-2:]=.', 'word.isdigit=False', '-2:word=എന്റെ', '-1:word=രാജ്യമാണ്', 'EOS', 'EOS']

Figure 4.4: Sample feature set for a simple sentence from the training data

where the model parameters are fine-tuned. A python-based implementation of CRF called 'pycrfsuite' is used for training [97]. After training, the model file is saved for later use. The second phase of the architecture is the testing phase, where the saved model is utilized for performance evaluation. Test sentences are also converted into sequences of feature vectors using the same method employed in the training phase. Finally, the feature sequences are provided to the saved model for tag sequence prediction. The predicted sequence is aligned with the input word sequence to produce the required output.

4.3.3 Experiments and results

The corpus prepared for the task was used for experimentation. It contains words that are unique and ambiguous which makes the tagging difficult. The noun is the most common tag present in the training corpus. Preprocessed tagged text was used for both training and testing. 80% of the total data was used for training and rest for testing. The proposed system was trained on 23,000 sentences and tested on 5750 sentences. The model parameters were tuned to produce maximum accuracy. The coefficient of L1 penalty was set as '1.0' and L2 penalty as '1e-3'. Training was conducted for 50 epochs. The maximum accuracy obtained by the system was 91.2%. Experiments were

also conducted to assess the performance of different existing tagging algorithms on CUSAT corpus. Figure 4.6 shows the performance of our algorithm, along with different existing algorithms on CUSAT corpus. An example of the tagged Malayalam text generated using our tagger is also given in Figure 4.5.

```
[['സ്റ്റാറ്റിൻസ്\\N_NN', 'എന്നത്\\CC_CCD', 'ഹൃദയസംബന്ധിയായ\\N_NN', 'രോഗങ്ങളുള്ളവർക്ക്\\N_NN'
'സാധാരണയായി\\RB', 'നൽകിവരുന്ന\\V_VM_VNF', 'മരുന്നാണ്\\V_VAUX', '.\\RDPUNC'], ['എന്നാ
\CC_CCD', 'കഴിഞ്ഞ\\JJ', 'കുറച്ചുകാലമായി\\N_NN', 'ഇതിന്റെ\\DM_DMD', 'ഉപയോഗത്തെയും\\N_NN',
'പാർശ്വഫലങ്ങളെയും\\N_NN', 'സംബന്ധിച്ച്\\PSP', 'വിവിധങ്ങളായ\\JJ', 'വാദപ്രതിവാദങ്ങളാണ്\
\V_VAUX', 'വൈദ്യശാസ്ത്രരംഗത്ത്\u200c\\N_NN', 'നടന്നുവരുന്നത്\\V_VM_VNF', '.\\RDPUNC']]
```

Figure 4.5: An example of the tagged Malayalam text generated using CRF based POS tagger

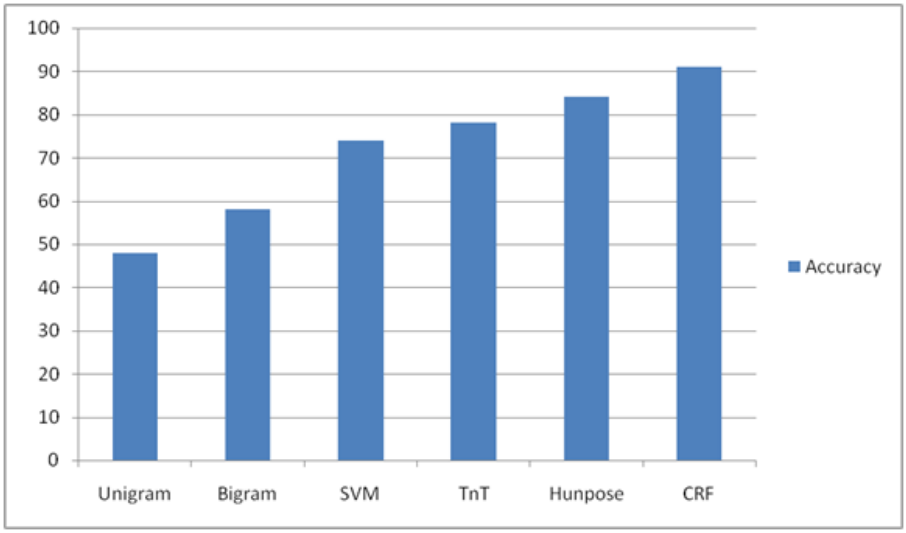


Figure 4.6: The performance of different tagging algorithms in comparison with CRF

4.4 POS tagging using neural networks

4.4.1 Introduction

Artificial neural networks are computing systems inspired by the structure and function of biological neural networks which constitute human brains. The key element in the architecture of ANNs is the structure of the information processing system. It is a framework for different machine learning algorithms to process complex data. Neural networks usually learn from examples rather than being explicitly programmed. For classification problems, the characteristics of different classes are automatically generated from the learning materials provided to them. Basic units of artificial neural networks are neurons which mimic the behaviour of biological neurons. The connections between neurons are called edges. Learning is programmed through the adjustment of weights in these edges. Neurons are set with a threshold and the aggregate signal passes through them if it crosses that threshold. Typically ANNs are constituted by a set of layers consisting of a set of neurons. Each layer will perform different kinds of transformations on the inputs received by them.

4.4.2 Architecture

In this section, we propose a methodology for POS tagging through the application of Neural Networks. A Neural Network contains three types of layers: the input layer, hidden layer, and the output layer. Here the input features go through the hidden

layers to reach the cost function, which calculates the error between real value and the predicted value. Then backpropagation algorithm is applied to minimize the error by finding the derivative of the cost function with respect to the network parameters. After the error at the output layer is minimized, it is backpropagated to previous layers for minimizing the hidden layer errors. This process is repeated in all the layers. And this cycle is repeated until the difference between the predicted value and the actual value is acceptable.

Since POS tagging can be considered as a classification task, we used Neural Network as our classifier. Neural Networks are potential tools in natural language processing. In neural networks, the number of hidden layers can be increased to the required level. The performance of the system gets improved with more amount of data. They can model complex nonlinear relationships. Our architecture contains three main modules. First is the representation module, which takes the input in some numerical form. Since words are symbolic constituents, it cannot be directly fed into neural networks. It should be converted into some numeric form. We have used word2vec for this purpose. The second module contains hidden layers. We conducted a lot of experiments with different number of hidden layers of various sizes. In our hidden layers, we have used 'Relu'activation units to introduce nonlinearity. 'Relu'provides faster convergence as compared with 'tanh'and 'sigmoid'. It also avoids the vanishing gradient problem in backpropagation algorithms. It is also known as a ramp function and is analogous to half-wave rectification in electrical circuits. Figure 4.7 illustrates the values and computations of the 'relu'function. The value of the

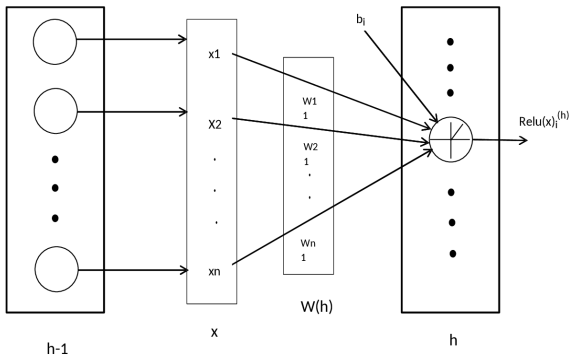


Figure 4.7: 'Relu' function illustration

'relu'function for any input 'x'is computed as in equation 4.6.

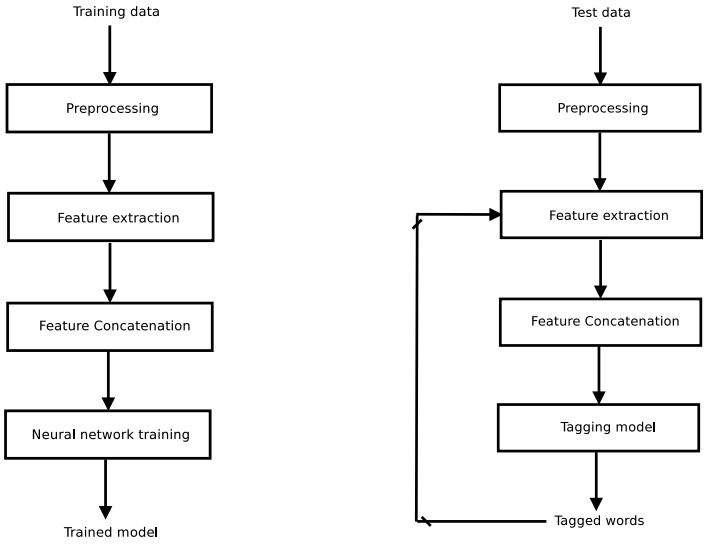
$$f(x) = max(0,x) \tag{4.6}$$

The third module is the output layer, which makes predictions. The output layers contain a set of softmax units. Softmax activations are used to represent the categorical distribution, a probability distribution over a set of different possible outcomes. The value of the softmax function for any given input y_j is calculated as in equation 4.7.

$$f(y_j) = \frac{e^{y_j}}{\sum_{k=1}^{K} e^{y_k}} \tag{4.7}$$

The architecture of the proposed system is shown in Figure 4.8. The pre-processed tagged text was used for training. The feature set for learning was selected based on the assumption that the tag for a particular word is decided by the tags of context words, morphological features of the target word and the target word itself. Hence it was decided to experiment with a

set of eight features. The selected features for experiments were word,word-1,word-2,word+1,word+2,tag-1,tag-2 and morphological information of the the target word. Words are symbolic units which will make no sense to computers when given as raw words. Hence it was decided to convert words into vectors which has some semantic meaning. Word embeddings were used for this purpose. Word embedding models converted words into vectors in a semantic space.



(a) Training module **(b)** Testing module

Figure 4.8: Architecture of the proposed system-training and testing modules

Word2Vec was used to convert words into vectors. Vectors of different sizes were created for experiments. Morphological features of the words are converted into vectors of numeric values using Word2Vec [98]. Tags of previous words were also converted into a numeric vector using both Word2Vec and one hot encoder. But later it was found that one hot encoding of tags were more accurate than Word2Vec representation, since tags were from a limited vocabulary of size 36. The vectors

of words, morphology, and tags were finally concatenated into a single vector to provide as an input to neural networks. The feature set for the first and second word in each sentence were filled with dummy vectors of zeros to make the feature set uniform in length. Training was done using Keras functional API model, and the model file was saved. Later on, this file was used for testing.

During the testing phase, the preprocessed testing data was fed to the feature extraction module, where the embedded representation of the word, embedded representation of suffix, and one hot representation of previous tags were extracted. For the first two words in the sentence, the feature vector was filled with necessary dummy values to make the feature vector uniform in length. For the rest of the words in the sentence, the one-hot representation of previous tags were provided dynamically from the final layer. Then all the features were concatenated to form the feature vector. Finally, the feature vector was provided to the trained model for tag prediction.

4.4.3 Experiments and Results

To demonstrate the merits of the proposed system, CUSAT Malayalam POS tagged corpus was used to conduct the experiments. Approximately 35% of the words were ambiguous, which made the tagging problem a challenging one. The performance of the tagger for different features and different network parameters are discussed here. After the preprocessing phase, the data set was divided into 80 % training and 20% validation sets. To make our experiments more reliable, 10 fold cross-validation

was also performed on the training data.

The functional API of Keras was used for implementation [99]. Functional model permits the addition of a linear stack of layers for feature extraction and transformation. Training was done after configuring the learning process, which was through the compile method. The compile method had three parameters, namely, optimizer, loss function, and metrics. SGD was used as the optimizer and categorical cross-entropy as the loss function. The tagging model is compiled using Tensorflow in the backend.

The system was trained on 2,30,000 words and tested on 57000 words. The overall accuracy of the tagging is 90.02%. The pre-processed tagged text was used for training. The developed model was a neural network with four layers. Setting the model parameters is an important task in fine-tuning neural networks. We empirically discovered that the best architecture for our model is a four-layered architecture. The hidden layers contain 400 neurons with 'relu'activation units. The output layer contain a loss function called categorical cross entropy. Through experiments, it was found that increasing the number of layers beyond a limit cannot help in improving accuracy. A network with two hidden layers was enough to represent the data. Network with more than two hidden layers did not improve the accuracy but only delayed the convergence time. The size of the hidden layers was also varied to fine tune the network. Size of the first layer was fixed, as it depends on the input dimension. And the size of the last layer was also fixed as it depends on the number of output classes. Hidden layers with a small

number of neurons were not enough to represent the data efficiently, and a large number of neurons only increased the computational complexity. According to the experiments, the best performance of the network was achieved when the number of neurons in the hidden layer was 400. A size of 200 was not enough to represent the data, whereas a size of more than 400 could not improve the performance. Hence, the hidden layer size was empirically finalized as 400.

The input given is a vector of features. This vector was formed by the concatenation of vectors of tags, words and morphological features. Constructing the optimal feature set was a trial and error process. A set of 8 features were considered for experiments. These features were the word, word morphology, previous words, successive words and tag of previous words. Each feature and its descriptions are given in table 4.3.

Table 4.3: Features and descriptions

Feature	Description
w	Target word
w^{-1}	Previous word
w^{-2}	The word before the previous word
w^{+1}	Successive word
w^{+2}	The word next to the successive word
t^{-1}	Tag of previous word
t^{-2}	Tag of the word before the previous word
$Suffix$	Suffix of target word

Input vector for each target word was finalized as $f_i : [w, w-morph, t-1, t-2]$, where f_i is the concatenation of four feature vectors. Our goal was to find the most probable tag for the word w_i, where it was followed by two prior tags t-1 and t-2. W-morph is a vector of morphological features of the word w_i.

Words and morphological features were converted into vectors using word2vec. Tags were converted into one hot vector. In one hot representation, each tag will have a unique representation in terms of zeros and ones. The performance of the model for different features is shown in table 4.4.

Table 4.4: Performance of the tagger for different features

Feature set	Accuracy
W	72.5%
W,W-1	74.4%
W,W-1,W-2	79.2%
W,W+1	73.3%
W,W+1,W+2	73.8%
W,T-1	75.6%
W,T-1,T-2	80.1%
W,T-1,T-2,Suffix	90.02%
W,W+1,T-1,T-2,Suffix	90.01%

Word2vec model was built based on a manually created corpus of 27 lakhs words. The model was constructed with a context window size of 10 and a minimum count value of one. Different models were built with different vector sizes. Each of them was used iteratively for training to find the one with the best performance. In Word2Vec, we had used CBOW (continuous bag of words) configuration. CBOW uses the mean of context words. Embedding of morphological features was also done using Word2Vec. The morphological features of words were extracted using a suffix stripper. Each word from the corpus was passed to the suffix stripper and the suffix stripper will search for the largest suffix matching with the set of stored rules. If there is no match, the suffix of length five was returned and if there is a match that matching suffix itself was returned. So each word in the corpus was replaced by a suffix

from the rules dictionary or a suffix of length five of that particular word. A set of 340 suffix stripping rules were prepared for this purpose. Then that corpus was trained using Word2Vec. The feature set in the input layer had four sub-vectors(target word, morphological representation of the target word, one hot representation of the word-1 tag, one hot representation of the word-2 tag).

During the experiments, it was found that the performance of the model gets improved with the increase in word vector size. Different results were explored with different embedding sizes of 20,40,60,80,100,120 and 140. The performance of the model improved until the word vector size reached 100. Beyond 100, we could not see a gradual change in accuracy. Hence, word vector size was finalized as 100. The performance of the model for different features were also experimented. It was found that the performance of the model was at its best when the number of features was four[w,w-morph,t-1,t-2]. It was also seen that the performance of the model increased along with the increase in training data size. Figure 4.9 shows the performance of the tagger for different amounts of training data. The performance of the system can still be increased by increasing the training data size.

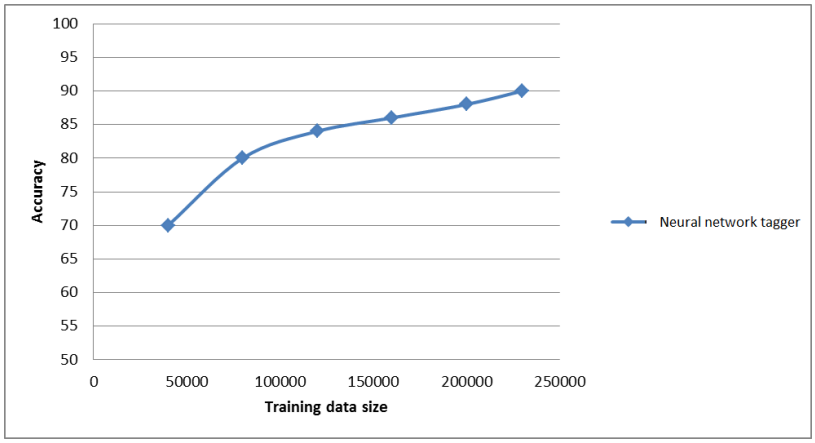


Figure 4.9: Performance of neural tagger for different training data size

4.5 POS tagging using deep learning

4.5.1 Introduction

Deep learning is one of the main contributors to the advancement of artificial intelligence in the current scenario. It is a subfield of machine learning which deals with the algorithms inspired by the structure and function of the human brain. The peculiarity of the deep learning models is the presence of multiple hidden layers with each layer accepting information from previous layers. Deep learning models can create complex statistical models from its own iterative output. Performance of the deep learning models improves with the increase in data, whereas the traditional machine learning models saturate after a particular point. Another key difference between traditional machine learning and deep learning models is on how feature extraction works. In traditional machine learning, all the features are hand-engineered. But in deep learning, the features are extracted by the hidden layers themselves. No more hand-

engineering is required in the case of deep neural networks. The word 'deep' in 'deep learning' is inspired by the number of hidden layers.

4.5.2 Architecture

Our objective is to construct a POS tagger which tags all the words in a sentence with corresponding POS tags. Even though neural networks have shown their outstanding performance in the last decade, they still have some limitation. They can't capture the contextual information, where the current input is affected by its previous inputs. Moreover, they assume all its inputs and outputs as independent of each other. If we want to predict the next tag in a sequence, it is better to know which tags came before it. Sequence to sequence tagging in deep learning is a promising solution for that. As shown in Figure 4.10, sequence to sequence learning problems optimizes the output sequence corresponding to the input sequence [100]. Deep neural networks are exceptionally powerful tools for the sequence to sequence learning. They are characterized by hierarchical feature learning, where each layer creates the abstract representation of its lower layer features. They can learn complex functions that can map the input to output directly.They can also perform parallel computation for an unassuming number of steps. Figure 4.11 shows the block diagram of our architecture.

In Malayalam, new words are formed by adding suffixes to words one after another. These suffixes may either carry grammatical functions or help in forming new nouns or verbs. And

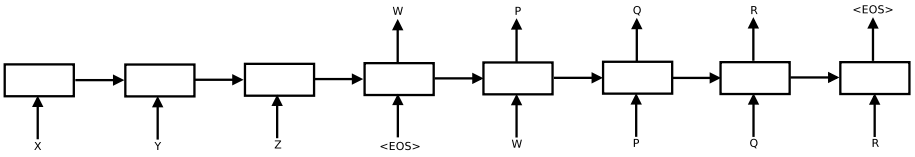


Figure 4.10: 'XYZ'is the input sequence and 'PQR'is the output sequence

there is no restriction on the degree and extent of agglutination in Malayalam language. Hence, it is not advisable to consider a full word as a processing unit. Unfortunately, there is no reliable and publicly available stemmer (or lemmatizer) for Malayalam language. Therefore, we decided to go with a word as a unit of consideration. Using the word level embedding features alone seems to be insufficient for POS tagging in inflectionally rich languages. Hence, to achieve additional improvement in performance, we added suffix level embedding features along with word-level embedding features.

In the following sections, the major focus is on presenting a novel word representation by combining character level, word level and suffix level features(embeddings) as shown in figure 4.12.

4.5.2.1 CNN-based character-level word representation

The first successful work on character-based compositional word embedding was proposed by Dos Santos and Zadrozny in 2014 [101]. They used convolutional neural networks to constitute word vectors from character embeddings encoded by column vectors in an embedding matrix. The character-based word embedding vector that we have used is similar to [102], where

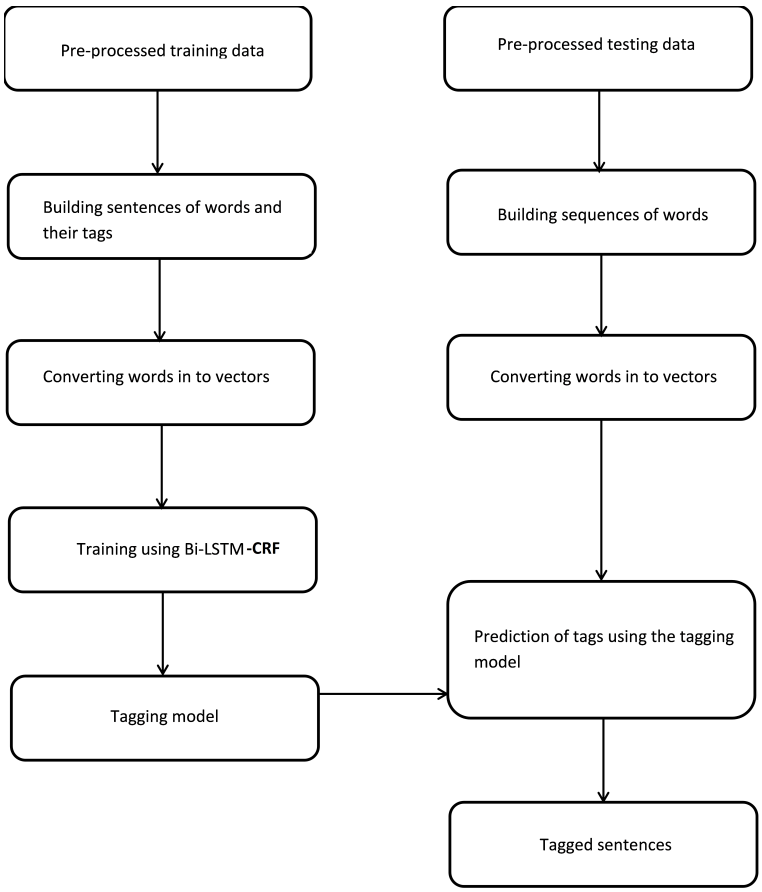


Figure 4.11: General system architecture

convolutional filters of different sizes are applied to capture the character n-gram features. The concatenated output of different convolutional filters through max-pooling layer act as the character based word embedding vector. Figure 4.13 shows the architecture of the character based word embedding model.

Given a word 'w' composed of 'm' characters $c_1, c_2, c_3, ...c_m$, where $c_i \in V_c$ is the character vocabulary set. Let $C_1, C_2, C_3, ...C_m$ be the character embedding vectors that encode the characters $c_1, c_2, c_3, ...c_m$ present in a word 'w'. The character embeddings

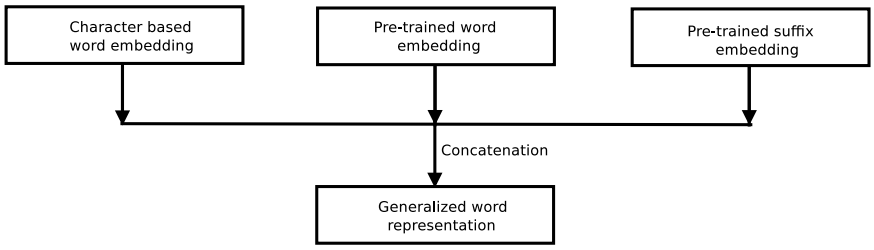


Figure 4.12: Generalized word representation

are obtained by matrix-vector product as given in equation 5.4.

$$C_1 = W_c V_c \tag{4.8}$$

Where W_c is the embedding matrix, $W_c \in R_{d_c} * |V_c|$ and V_c is the one-hot vector representation of a particular character. Hence, each word is transformed into a sequence of character embeddings $C_1, C_2, C_3, ...C_m$. We apply convolutional kernels to each of the sliding context window of size 'k'. The resulting vectors are passed through a max-pooling layer to generate the maximum value. We then concatenate these vectors from different convolutional kernels to produce the required word embedding vectors. These vectors are expected to capture information from different n-grams of the same word.

Formally, the character level embedding of each word 'w' is calculated as follows,

$$V_c = \underset{1<i<m}{max} [W_{conv} Z_m + b_{conv}] \tag{4.9}$$

Where W_{conv} and b_{conv} are parameters of the model and Z_m is the concatenation of character embeddings expressed as

$$Z_m = \left(C_{m-(k-1)/2}, ..., C_{m+(k-1)/2} \right) \tag{4.10}$$

The convolution operation is applied to find the simple patterns of embedding vectors of different n-grams over the character

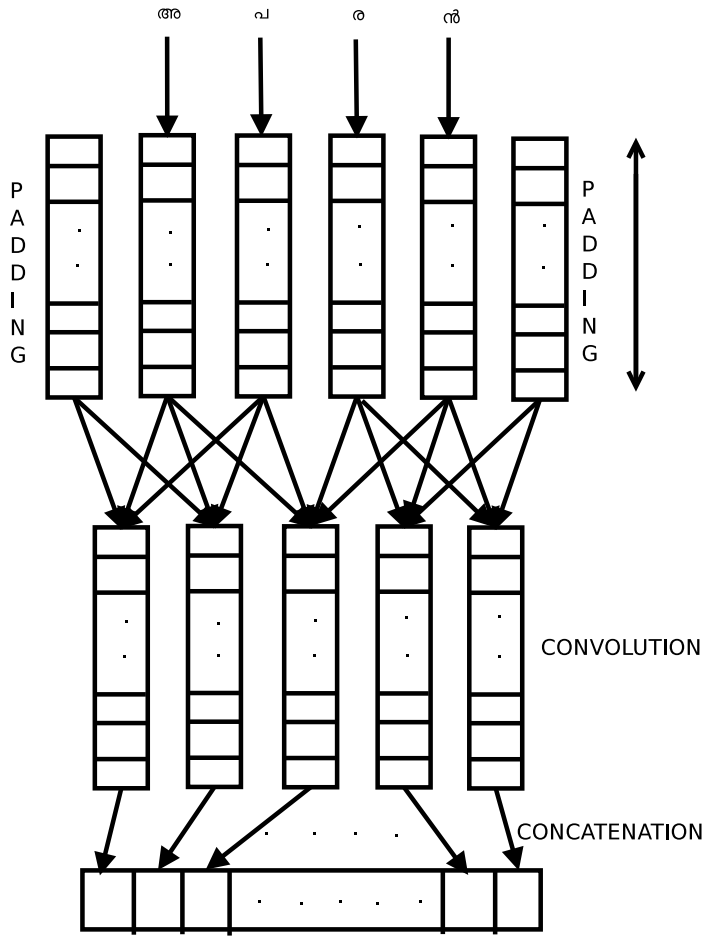


Figure 4.13: Architecture of the character based word composition model

sequence. Among different n-grams the following maxpooling layer try to extract position invariant n-gram features. Therefore, the character-based word composition model is expected to detect unvarying local spelling features from the character sequence.

4.5.2.2 W2V-based pre-trained word representation

Word embeddings are proven tools for capturing the context of a word along with its syntactic and semantic similarities in vector representation. They can also model the relation with other words in a corpus. They are created by applying a large collection of unlabelled text over shallow neural networks. In our study, we used Word2vec to create pre-trained word embeddings. Word2vec is the most popular tool for creating the word embeddings. It was developed by Tomas Mikolov and team at Google in 2013.

4.5.2.3 W2V-based pre-trained suffix representation

Embedding of morphological features was done using word2vec. The suffix level features of words were extracted using a suffix stripper and those features were used to create embedding vectors. Instead of using sentences of words, here we used sentences of suffixes returned by the suffix stripper. Later that corpus was used for building suffix embedding of the desired size using Word2Vec. Everything else remained the same as the Word2Vec based word representation model.

Formally, assume the given Malayalam sentence $S_{[1:n]}$ is a sequence of n words, where 'n'is the maximum length of the sentence. In our case, it was limited it to 30 for the ease of computation. Each word 'w_i' in the sentence was converted into a composition of vectors corresponding to that word. These vectors included character level word composition vector, pre-

trained word embedding vector and suffix embedding vector. Hence each word in the sentence was replaced by a vector of d-dimension, where $d = d_c + d_w + d_s$ such that d_c = dimensionality of character level word embedding, d_w = dimensionality of pre-trained word embedding and d_s = dimensionality of suffix embedding. Equations 4.11 and 4.12 shows the mathematical representation of a single sentence.

$$S = [w_1, w_2, w_3, w_4, ...w_n] \tag{4.11}$$

$$S = [[v_{c1}, v_{c2}, v_{c3}.....v_{cp}, v_{w1}, v_{w2}, v_{w3}.....v_{wq}, v_{s1}, v_{s2}, v_{s3}.....v_{sr}], [v_{c1}, v_{c2}, v_{c3}.....v_{cp}, v_{w1}, v_{w2}, v_{w3}.....v_{wq}, v_{s1}, v_{s2}, v_{s3}.....v_{sr}], ...] \tag{4.12}$$

where $v_{c1}, v_{c2}, v_{c3}.....v_{cp}$ corresponds to character based word vector, $v_{w1}, v_{w2}, v_{w3}.....v_{wq}$ corresponds to pretrained word vector and $v_{s1}, v_{s2}, v_{s3}.....v_{sr}$ corresponds to suffix embedding vector. Equation 4.12 shows the complete representation of an input sentence to Bi-LTM-CRF tagger.

Figure 4.14 demonstrates the complete architecture of our model. The model accepts sequences of length 'n'. 'We' and 'Se' are word embedding and suffix embedding respectively, where 'We' is the concatenation of pre-trained word embedding and character-based word embedding. P0, P1, P2, etc. are the emission scores coming from the Bi-LSTM layer. Each of them indicates the probability of a particular tag for a particular word. Hence, 'm' corresponds to the maximum number of tags in the tag set. Final predictions are made by the CRF layer based on

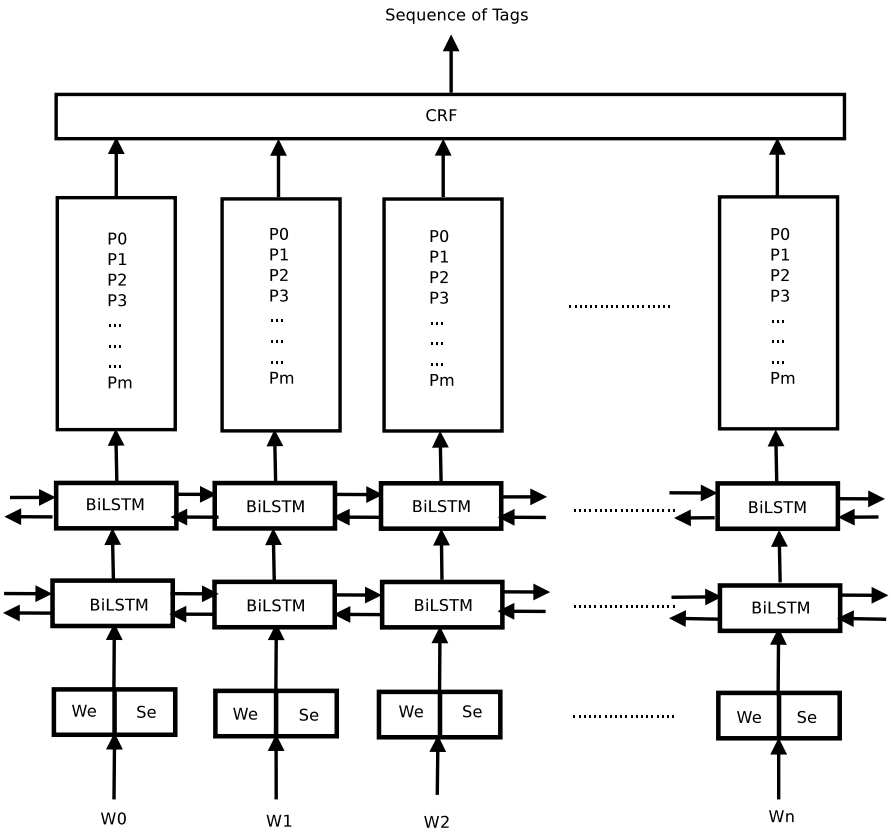


Figure 4.14: Architecture of the of Bi-LSTM model with CRF output layer

these probabilities.

Keras functional API was used to build the sequence to sequence learning model. In sequence to sequence learning, two recurrent neural networks work together convert one sequence to another. First one is known as the encoder network and the second one is called as the decoder network. The encoder network condenses the input sequence into a vector and the decoder network unfolds the encoded vector into a new sequence. This point is illustrated in Figure 4.16.

Bi-LSTM with CRF on top was used to map the input se-

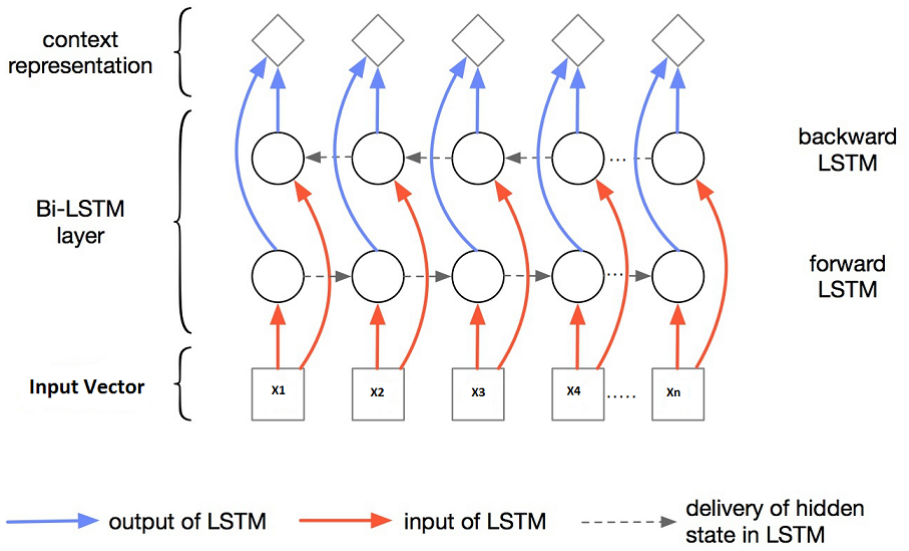


Figure 4.15: Bi-LSTM layer architecture

quence to output sequence. LSTM is a variant of RNN and free from long term dependency problem. Their default behaviour is to remember information for intervals of time. RNNs are capable of using contextual information in sequence prediction problems. But the range of context is limited, which is the major drawback of RNN. LSTMs overcomes this problem using a set of gates, namely input gate, output gate and forget gate. RNNs and LSTMs can preserve information only in one direction. Bi-LSTM is a bidirectional variant of LSTM that can preserve the information from both the directions. Figure 4.15 shows the architecture of a single Bi-LSTM layer. The forward function of LSTM is calculated using the equations 4.13 and 4.14.

$$a_h^t = \sum_{k=1}^{K} y_k^t W_{kh} + \sum_{h'=1,t>0}^{H} b_{h'}^{t-1} W_{h'h} \tag{4.13}$$

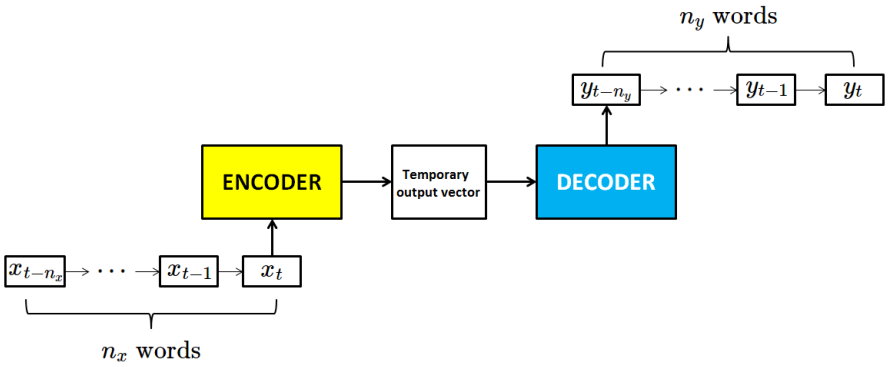


Figure 4.16: Encoding and Decoding of sequences by LSTM

$$b_h^t = \Theta_h(a_h^t) \tag{4.14}$$

Where y_t is a sequence input, a_t is the input to the LSTM unit h at time t, b_h is the activation function at time t, w_{kh} is the weight of the input k towards h. w_{hh}' is the weight between the hidden layers h and h' While the backward function of Bi-LSTM is calculated by the equations 4.15 and 4.16.

$$\frac{\delta O}{\delta W_{hl}} = \left(\sum_{t=1}^{T} \frac{\delta O}{\delta a_h^t} b_h^t \right) \tag{4.15}$$

$$\frac{\delta O}{\delta a_h^t} = \Theta_h(a_h^t) \left(\sum_{l=1}^{L} \frac{\delta O}{\delta a_h^t} W_{hl} + \sum_{h'=1,t>0}^{H} \frac{\delta O}{\delta a_{h'}^{t+1}} W_{hh'} \right) \tag{4.16}$$

The shaded nodes in figure 4.17 indicate LSTM sensitivity to the input at time one. The black nodes are highly sensitive and white nodes are completely insensitive. The states of different

gates are displayed to the below, left and right of the hidden layer. All gates are either completely open or closed. The memory cell can retain the information about the first input as long as the forget gate is open and the input gate is closed. The sensitivity of the output layer is controlled by output gates. LSTM tries to find the conditional probability $P(y_1, y_2, \ldots, y_T | x_1, x_2, \ldots, x_T)$ as given in equation 4.17, where $(x_1, x_2, \ldots, x_T)$ is the input sequence and $(y_1, y_2, \ldots, y_T)$ is the output sequence.

$$p(y_1, y_2, \ldots, y_T | x_1, x_2, \ldots x_T) = \prod_{t=1}^{T} P(y_t | v, y_1, y_2, \ldots, y_{t-1}) \tag{4.17}$$

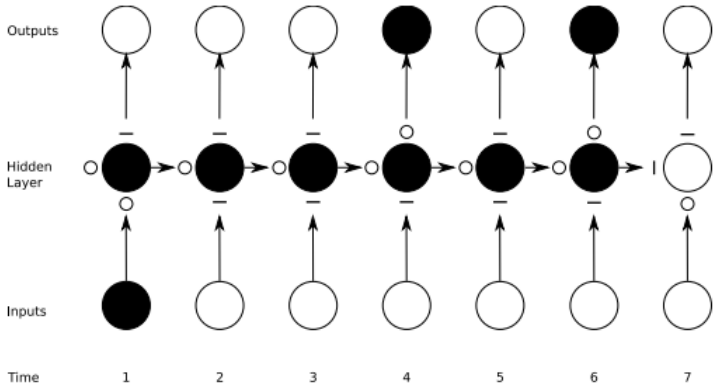


Figure 4.17: Preservation of gradient information by LSTM

A CRF layer was used to make the final predictions, given the probabilities for various tags for each word. CRF layer is capable of incorporating some additional constraints to the probabilities generated by the Bi-LSTM layers. These constraints are automatically learned by the CRF layer from the training data. Thereby the CRF layer ensures the validity of generated tag sequence. The CRF layer consider two types of scores

namely-emission score and transition score. Emission score is the probability values coming from the BiLSTM layers and the transition score is taken from a probability matrix stored in CRF layer, which indicate the transition probability among different tags. The summation of these two scores was used in the calculation of path scores of different sentences. The way of calculating the path score for a sentence of length five is shown in equations 4.18, 4.19 and 4.20.

$$\begin{aligned} EmissionScore = {} & x_{0,START} + x_{1,Noun} + x_{2,Demonstrative} \\ & + x_{3,LocativeNoun} + x_{4,FiniteVerb} + x_{5,Punctuation} + x_{6,END} \end{aligned} \tag{4.18}$$

$$\begin{aligned} TransitionScore = {} & t_{START->Noun} + t_{Noun->Demonstrative} + \\ & t_{Demonstrative->LocativeNoun} + t_{LocativeNoun->FiniteVerb} \\ & + t_{FiniteVerb->Punctuation} + t_{Punctuation->END} \end{aligned} \tag{4.19}$$

$$PathScore = EmissionScore + TransitionScore \tag{4.20}$$

Here $'x'_0$ and $'x'_6$ are the start and end markers which will be considered for all the sentences. The loss function calculates the real path score and the total score for all possible paths in the label sequences. Real path score is the score of the correct label sequence, and the total score is the sum of all possible path scores for a particular sequence. During the training process, the parameters of BiLSTM-CRF model will be updated again and again as in equation 4.21.

$$LossFunction = \frac{P_{RealPath}}{P_1 + P_2 + ... + P_N} \tag{4.21}$$

The proposed network contains three parts. First one is the input module, which receives the embedded representation of words and suffixes. The second one is constituted by a set of hidden layers. Hidden layers are BiLSTM layers with 'tanh' activation. The calculation of 'tanh' value for a particular input 'y' is shown in equation 5.9.

$$\begin{aligned} tanh(y) &= (2\sigma(2y) - 1), where \\ \sigma(2y) &= \frac{e^{2y}}{(1 + e^{2y})} \end{aligned} \tag{4.22}$$

The last layer of the architecture is the output layer. CRF was used for decoding in the output layer. CRF finds the best tag sequence corresponding to the word sequence. Dropout was used to prevent overfitting, which avoids complex co-adaptations on training data [103].

4.5.3 Experiments and Results

The preprocessed tagged text was converted into a sequence of words and sequence of their corresponding tags. For example, a tagged sentence like "Raju\NNP met\VB Raman\NNP .\RD_PUNC" was converted into "Raju met Raman ." and "NNP VB NNP RD_PUNC". Since words can't be directly fed to neural networks, they were converted to numeric values using Word2vec. The Word2vec model was constructed using our

own corpus of 2.7 million words. The context window size was set as 6, and the minimum count was set to one. To evaluate the performance of the proposed deep neural network on different vector sizes, we constructed different Word2vec models with different vector sizes. CBOW configuration was used to create the models, which used the mean of context words. Out of vocabulary words were handled using an Unknown token. Word2vec was also used to embed the morphological features. In this case, each word in the training corpus was either replaced by the morphological feature of the word or part of the word itself. The morphological features of words were extracted using a suffix stripper which searched for the largest suffix matching with a set of stored rules. It there is a match that matching suffix was returned else a suffix of length five was returned. Hence, each word in the corpus was replaced by a suffix from the rules dictionary or a suffix of length five of that particular word. A set of 340 suffix stripping rules were prepared for this purpose. Some of the stored rules are given in Figure 4.18. Finally, that corpus was used for building the suffix embedding features.

ുപയോഗിച്ച്=ുച് + ഉപയോഗിക്കുക
യായിരുന്നു= + ആയിരുന്നു
ള്ളതാണ് =ള്ളത് + ആണ്
യുണ്ടാക്കി= + ഉണ്ടാക്കി
മായാണ്= ം + ആയാണ്
യായി= + ആയി
യാകും= + ആകും
യുള്ള= + ഉള്ള

Figure 4.18: Examples of rules used to extract the suffix

Different sentences may have a different length. But Bi-LSTM network requires sequences of uniform length. Hence, zero padding was applied to make the sequence length uniform.

Sentences with less number of words were padded with zeros to make their length compatible with the maximum length sequence. In experiments, the sequence length was limited to 30 for the ease of computation. After that, tag sequences were replaced by integer sequences such that each tag was assigned with a unique integer. These numbers were transformed into one hot vector to make the network understand the data.

Keras functional API was used to implement the network. The network was a stack of layers which performs feature extraction and transformation. Information about the shape was provided in the first layer and rest of the layers did automatic inference about the shape. Fully connected network with two Bi-LSTM and one CRF layers constituted the model. 'Adam' was used as the optimizer and 'categorical cross entropy' as the loss function. The model was compiled using Tensor flow in the backend. Batch size was fixed as 100. The network was trained for 10 epochs. 80% of the total data was used for training and the rest of the data was used for testing. The proposed system achieves an overall accuracy of 94.33%.

4.5.3.1 Impact of network parameters

Parameter setting is the most challenging part when working with neural networks. We empirically discovered that the best architecture for our network is a four-layered architecture. The first layer was the input layer which recieves the input word representations. The input shape was also specified in the first layer. The second layer was a Bi-LSTM layer with 400 neurons and 'tanh' activation. The third layer was also a Bi-LSTM

layer with 400 neurons and 'tanh' activation. The return sequences argument must be set to true in all Bi-LSTM layers so that the successive layers has a three-dimensional sequence input. The last layer was the output layer which makes the predictions. CRF layer was used to make final predictions. A network with more than two hidden layers was not able to improve the precision and only delayed the convergence time.

The size of the hidden layers was also a vital element in setting the network parameters. Bi-LSTM layers with less number of neurons could not represent the data efficiently, and a very large number of neurons only increased the computational complexity. According to the experiments conducted, the network performs well, when the number of units in the hidden layer was moderate. A size of 150 was not enough to represent the data, whereas a layer with more than 400 units did not improve the performance. Hence the hidden layer size was experimentally finalized to 400. Adam, an extension to stochastic gradient descent was used to minimize the cost function over training data. Batch size was fixed at 16 and learning rate was initialized to 0.0001. The remaining set of hyperparameters for our experiments were fine-tuned as follows: the dimensionality of character-based word embedding and pre-trained word embedding was set to 100 and the dimensionality of suffix embedding was fixed to 50.

4.5.3.2 Impact of word embedding size

The effect of different word embedding size on the performance of the deep neural network was also investigated. Experiments

were conducted to evaluate the performance of the network for different word embedding sizes such as 40, 60, 80, 100, 120 and 140. We could see a gradual improvement in accuracy until the word vector size reaches 100. Beyond 100, there was no considerable change in performance. This point is illustrated in Figure 4.19.

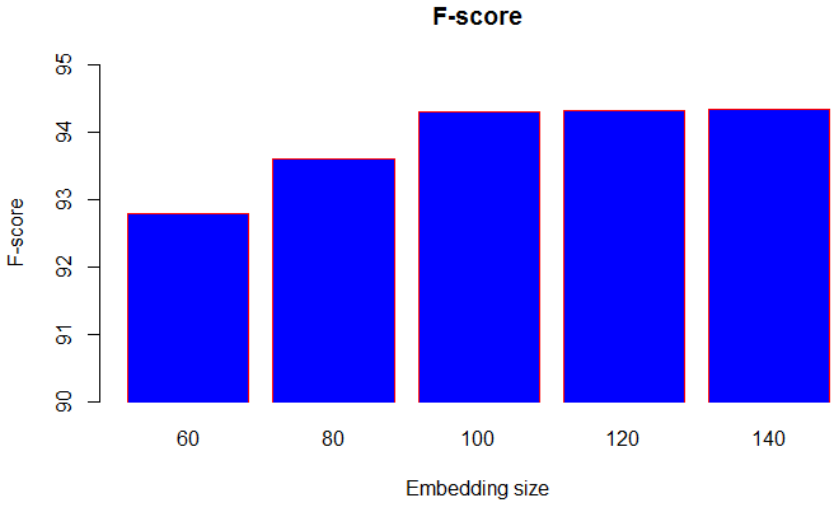


Figure 4.19: Performance of the system for different pre-trained word embedding sizes

4.5.3.3 Impact of suffix embedding features

Effective utilization of suffix level features can improve the performance of the tagger in morphologically rich languages. Since Malayalam is a morphologically rich language use of such features was very helpful in improving the accuracy. The performance of the proposed system in the presence and absence of suffix embedding were evaluated. Figure 4.20 illus-

trates this point. Here 'A' is the accuracy of the tagger in the absence of morphological features and 'B' is the accuracy of the tagger in the presence of suffix embedding features. An improvement of 0.8% was obtained by incorporating suffix embedding along with word embedding features. An example of the tagged text in the presence and absence of suffix embedding is given in Figure 4.21. Here the correct tag for the second word is 'V_VM_VNF', which was generated only after incorporating the morphological embedding.

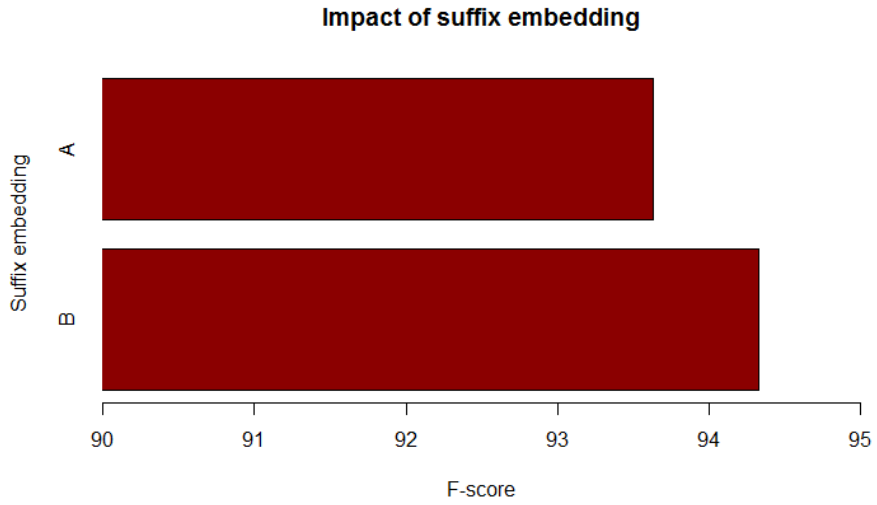


Figure 4.20: Performance of the model with and without the presence of suffix embedding

ക്ലിയോപാടയുടെ \N_NN ,മരണത്തിനു \N_NN ,കാരണം \CC_CCD ,ആയതും \N_NN ,അസ്പ് \N_NN ,ആണ് \V_VM ,എന്നു \CC_CCS ,കരുതപ്പെടുന്നു \V_VM_VF ,. \RDPUNC

ക്ലിയോപാടയുടെ \N_NN ,മരണത്തിനു \V_VM_VNF ,കാരണം \CC_CCD ,ആയതും \N_NN ,അസ്പ് \N_NN ,ആണ് \V_VM ,എന്നു \CC_CCS ,കരുതപ്പെടുന്നു \V_VM_VF ,. \RDPUNC

Figure 4.21: Example of the tagged text with and without the presence of suffix embedding

4.5.3.4 Comparison with the existing systems

Even though various works were reported for POS tagging in Malayalam, none of them is publicly available as a tool except the one from IIITMK, Trivandrum [104]. To compare the performance of the proposed system with different existing methods, most of them were simulated using scikit-learn and NLTK libraries. They include SVM, HMM, Unigram, Bigram and Hunpose. HMM-tagger is implemented using the NLTK implementation available in [105]. The HMM tagger obtained an accuracy of 72.02% on experiment dataset. Among the available systems, the best results were produced by the CRF based POS tagger, which was our previous work [106]. CRF tags with an accuracy of 91.2%. Hunpose, an open source POS tagger well suited for morphologically rich languages performed with an accuracy of 84% [107]. But the proposed system outperformed the existing systems by a minimum margin of 3.13%. A comparison of their performances on the experiment dataset is given in Figure 4.22.

4.6 Summary of the chapter

This chapter discusses the preparation of a POS tagged corpus and three different POS tagging mechanisms for Malayalam. The preparation and validation of dataset are discussed in detail. The proposed algorithms were tested on the constructed dataset. Experiments were conducted to evaluate the performance of different tagging algorithms on our dataset. The power of word embeddings is also utilized in the study with the

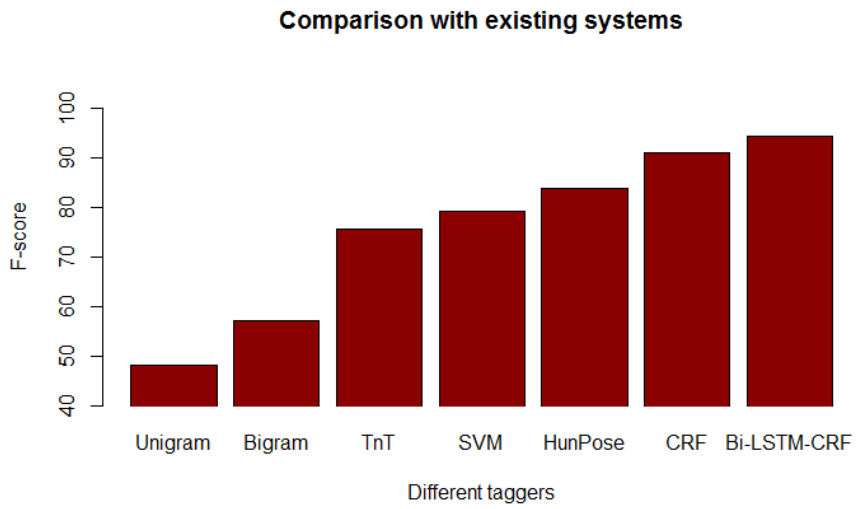


Figure 4.22: Accuracy of different tagging algorithms over CUSAT corpus

help of Word2vec. The proposed work is one of the preliminary studies in Malayalam NLP, which utilizes the power of word embeddings. Deep learning techniques were explored in the study with the help of Keras neural network API. Morphological features of the language were utilized in the study with the help of suffix stripping algorithms. The use of suffix embeddings, along with word embedding features, resulted in a significant improvement in performance. During the training phase of all the algorithms, it was observed that the performance of the models increases with the increase in training data size. Hence, it is better to increase the size of the training data for more improved performance.

5

Named Entity Recognition

This chapter describes different schemes used for NER in Malayalam. NER is the process of identifying the basic units (words) in a text document and classifying them to predefined categories such as person, location, organization, etc. It is also known as entity identification since it locates and classifies the entities mentioned in an unstructured text. It is a subtask of information extraction that finds important application in content recommendation and customer support based systems. Due to the lack of publicly available datasets, a dataset for NER in Malayalam is prepared. Different methodologies like CRF, neural networks, deep neural networks, etc. are experimented on the dataset. A comparative study regarding the performance of different entity tagging algorithms on Malayalam language is conducted. Different features that affect the performance of entity recognition in Malayalam are also experimented.

Section 5.1 of this chapter gives a brief introduction about named entity recognition. Section 5.2 discusses dataset preparation. It also discusses the details about the shared data set that we received as part of the shared data competition. Section 5.3 of this chapter describes named entity recognition using neural networks. Section 5.4 gives a detailed description

of NER using Conditional Random Fields. Section 5.5 presents the application of deep learning techniques for NER in Malayalam. A comparative study regarding the performance of different sequence labelling algorithms on NER in Malayalam is also presented in that section. The chapter is concluded in section 5.6 with a brief overview of the summary.

5.1 Introduction

Named entity recognition is an important technique to extract meaningful information from unstructured text. Named entities are often more informative and have unique contexts. They add semantic knowledge to the text and helps to properly understand the subject of a given document. Entity recognition systems can automatically scan the complete article and identify which are the important people, places and organizations discussed in them. Knowing the relevant tags for articles, helps in the automatic categorization of articles and smooth content discovery. NER can also be used in customer support departments of electronic shops where the customer's feedbacks go through the entity recognition API to find out the relevant tags from the feedback. These tags can be used to categorize the complaint and allocate it to the concerned department within the organization.

Lots of challenges are there in building an automatic entity recognition system for Malayalam. First one is the lack of capitalization feature, which is one of the major feature used in languages like English. Inflectional nature of the language

is the second biggest challenge which allows the appearance of the same word in different forms. Lack of resources like standard datasets, POS taggers, morphological analyzers, etc. are also a barrier for creating a named entity recognition system for Malayalam.

5.2 Dataset preparation

Since no standard dataset was available for entity recognition task in Malayalam, a dataset that considered mainly three types of entities-namely person, location and organization was prepared. The POS tagged corpus developed as part of our POS tagging works was also used to create the NER corpus. A set of 204080 words from the POS tagged corpus were tagged using BIO (Begin, Inside, Outside) tagging scheme. A sample text tagged using the BIO tagging scheme is shown in Figure 5.1. The authenticity of the prepared dataset was evaluated using the Fleiss kappa coefficient, as mentioned in the previous chapter. An overall Kappa score of 0.81 was achieved for the entity tagged corpus. The statistics of the different tags in the tagged corpus is given in table 5.1.

Later, a dataset for entity recognition was recieved through the participation in IECSIL-2018, a track in FIRE-2018 [108]. IECSIL-2018 was a shared data challenge organized by ARNEKT solutions in association with FIRE-2018 [109]. The dataset contained training data for five Indian languages, namely Malayalam, Kannada, Tamil, Telugu, and Hindi. The dataset contained nine tags including name, location, organization, event,

Table 5.1: Frequency of Different Named Entity Tags present in the Training Data

Tag	Expansion of Tag	Frequency
B-PER	Begin Person	1480
I-PER	Inside Person	233
B-LOC	Begin Location	871
I-LOC	Inside Location	21
B-ORG	Begin Organization	264
I-ORG	Inside Organization	134
MISC	Miscellaneous	74220
OUT	Outside	79424

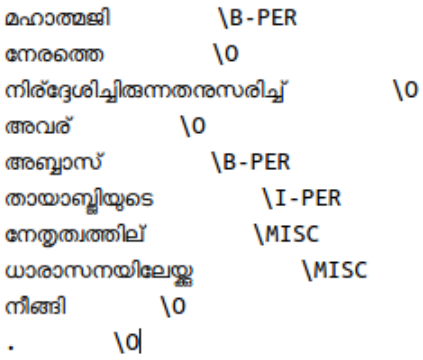


Figure 5.1: Sample text tagged using the BIO tagging scheme

things, occupation, number, date/time and other. Summary of the dataset is given in table 5.2.

Table 5.2: Summary of the dataset

Dataset	# Sentences	# Words	# Unique Words
Hindi	76,537 25,513	1,472,033 493,602	87,842 43,797
Kannada	20,536 6846	297,820 100,479	73,712 34,200
Malayalam	65,188 21,730	838,333 280,130	143,990 67,361
Tamil	134,030 44,677	1,492,230 497,548	185,926 89,529
Telugu	63,223 21,075	777,681 259,458	108,059 51,555

5.3 NER using neural networks

5.3.1 Introduction

Neural networks are a set of algorithms that are inspired by the structure and function of the human brain. They are characterized by their immense ability to recognize patterns. Neural networks follow a different approach towards solving a problem than that of the traditional algorithms. The traditional algorithms follow a set of instructions to get the solution for a problem. If the set of instructions are not clear and concrete, they can't solve the problem. This restricts the ability of computers to solve problems using conventional algorithms. Here comes the advantage of neural networks, where they could do things that we don't have an exact solution in an algorithmic

way. Neural networks solve problems in a similar way that the human brain does. Different layers of the neural network work together to solve a particular task. As in the case of humans, examples act as supervisors in the learning scenario.

5.3.2 Architecture

Figure 5.2 depicts the architecture of the proposed system. The important parts of the architecture are the training phase and the testing phase. In the training phase, the pre-processed training data was fed to the feature extraction module. The feature extraction module extracts a set of features for each word. The set of features considered in the study were the POS information of the target word and preceding words, embedded representation of the target word and preceding words and embedded representation of the suffix. The features returned by the feature extraction module were concatenated and fed to the neural network training module along with the labels for each word. After training, the model file was saved and used for testing. Testing phase also contained steps like feature extraction and feature concatenation. The concatenated features were finally given to the trained model for tag prediction. The predicted tags were aligned with the input words to produce the required output.

The proposed network included an input layer, two hidden layers and an output layer. The feature set for each word was provided to the input layer, where the input features were multiplied by a set of weights to reach the hidden layer. In the hidden layer, an activation function was applied on the weighted

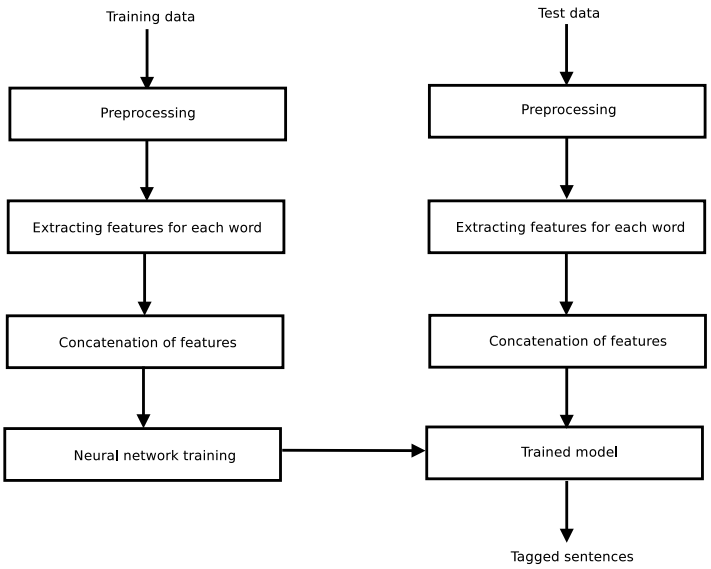


Figure 5.2: Architecture of the neural network based NER system

sum of inputs to bring non-linearity. This process was repeated up to the output layer. Cost functions were used to estimate the difference between the predicted value and the actual value. Network weights were adjusted using the backpropagation algorithm. The process of weight adjustment continued until the difference between the predicted value and real value was acceptable.

5.3.3 Experiments and Results

Experiments were conducted using an inhouse corpus. 80% of the total data was used for training and the rest of the data was used for testing. MLP classifier was used to conduct training and testing [110]. The proposed network had two hidden layers with 400 neurons each. 'Relu' was used as the activation function and 'Adam' as the optimizer. Batch size was fixed at

100 and learning rate as '0.001'. The network was implemented in python using Tensorflow in the backend. Word embeddings and suffix embeddings were prepared, as discussed in section 4.3 of chapter 4. Different features were iteratively applied to find their effects in accuracy. From experiments, we were able to find the most relevant features that affected the accuracy of Malayalam NER systems. They were word embedding of the target word, POS information of the target word, and suffix embedding of the target word. The influence of different features on the performance of NER is given in table 5.3. The number of epochs was limited to the point of maximum validation accuracy and it was 25 epochs.

Table 5.3: Accuracy of the system with respect to different feature set

Feature set	Accuracy
Word	85.4%
Word, POS	86.9%
Word, POS, Suffix	91.3%
Word, POS, POS-1, Suffix	91.45%
Word, POS, Word-1, Suffix	94.4%
Word, POS, Word-1, Word-2, Suffix	95.3%
Word, POS, POS-1, POS-2, Word-1, Suffix	94.25%
Word, POS, POS-1, POS-2, Word-1, Word-2, Suffix	95.33%

Performance of the network for different word vector sizes were experimented. Different results were obtained by incorporating different word vector sizes. Best results were obtained when the word vector size was 100. Hence, the word embedding size was fixed at 100. Figure 5.3 shows the performance of the neural network on different word embedding sizes.

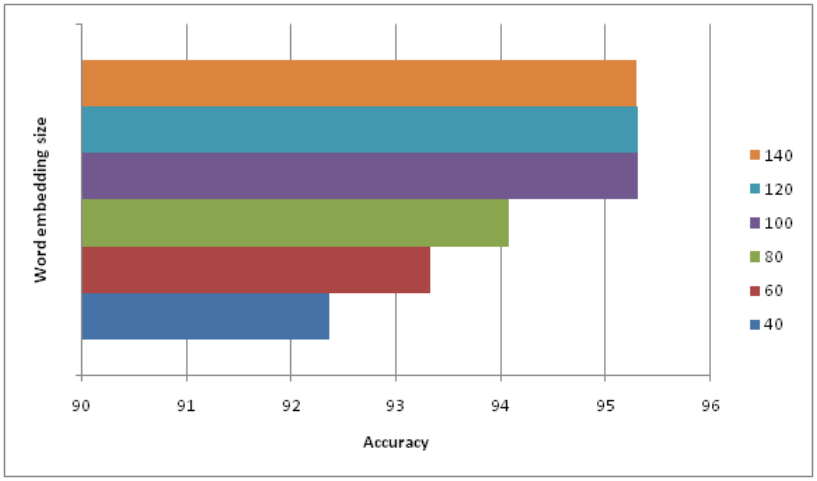


Figure 5.3: Accuracy of the proposed NER system on different word embedding sizes

5.4 NER using CRF

5.4.1 Introduction

CRFs are well known for tools for entity recognition tasks. Even the best performing Stanford university NER tagger is based on the linear chain Conditional Random Fields. In Indian languages, the application of CRF for NER is not novel. Variety of works are reported in languages like Tamil, Telugu, Kannada, etc. that used CRF. But most of them are in personalized datasets of limited size. CRFs are a class of statistical modelling tools that are often used in structured prediction problems. They are well-known for their ability to take context into account in sequence prediction problems. In this work, we have considered linear chain CRFs, a variant of CRF to model our problem. The dataset for this work was provided by the

competition organizers of IECSIL-2018 [1].

5.4.2 Architecture

The problem was framed in the same way as in the POS tagging problem. The only difference is in the feature selection part. In the POS tagging problem, a set of different features such as the previous word, succeeding word, suffixes of different length, etc were considered. But here some more features including POS of the target word and POS of the neighbouring words were taken into consideration. The schematic view of the overall architecture is shown in Figure 5.4.

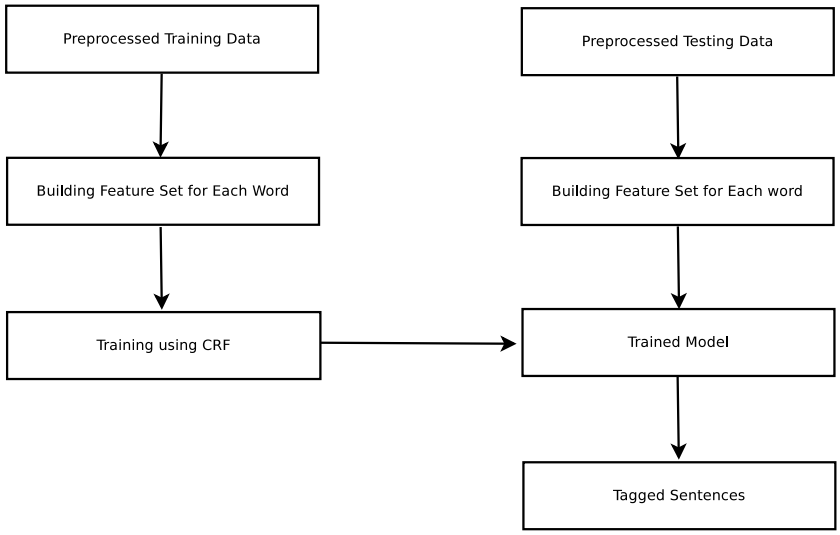


Figure 5.4: Architecture of the CRF-based NER system

Let $X = x_1, x_2, x_3, ...x_n$ be the input sequence and $Y = y_1, y_2, y_3, ...y_n$ be the corresponding label sequence. CRFs try to maximize the conditional probability distribution$P(Y/X)$ given the input sequence. The best entity tag sequence corresponding to a word

[1] https://github.com/BarathiGanesh-HB/ARNEKT-IECSIL

sequence was calculated as shown in equation 5.1.

$$\hat{\bar{y}} = \arg\max_{\bar{y}} P(\bar{y} \mid \bar{x}; \bar{w}) \tag{5.1}$$

Here $\bar{x}$ is the observable word sequence and $\bar{y}$ is the corresponding entity tag sequence. The probability of an entity tag sequence $\bar{y}$, for a given word sequence $\bar{x}$, was calculated as shown in equation 5.2. Where $\bar{w}$ denotes the weight vector and F is the global feature vector.

$$P(\bar{y} \mid \bar{x}; \bar{w}) = \frac{\exp(\bar{w} \cdot F(\bar{x}, \bar{y}))}{\sum_{\bar{y}' \in Y} \exp(\bar{w} \cdot F(\bar{x}, \bar{y}'))} \tag{5.2}$$

The conditional probability of Y_i on X is defined through a set of feature functions. Each feature function was assigned by a particular weight, as shown in equation 5.3. CRFs can accommodate any number of feature functions. The feature functions can inspect the entire input sequence X at any point during the inference. Each feature function can analyze the entire observation sequence $\bar{x}$, the current y_i and previous y_{i-1} positions in the tag sequence and current position i in the observation sequence. A feature function is computed by summing f_k over all n different state transitions $\bar{y}$.

$$F(\bar{x}, \bar{y}) = \sum_i \sum_j \lambda_j f(y_{i-1}, y_i, \bar{x}, i) \tag{5.3}$$

Finally, the decoding of the best tag sequence was done by using the Viterbi algorithm.

First of all, the training data was preprocessed by a preprocessing module, where the input data was transformed into a sequence of words and a sequence of corresponding tags. Figure 5.5 shows the output of the preprocessing module for a single tagged sentence. The preprocessed training data was given to the feature extraction module, where the feature set for each word was extracted. Set of features extracted for a sample word is given in figure 5.6. Afterwards, the sequence of features was provided to the CRF training module. After training, the model file was saved for testing. During the testing phase, the input data was processed in the same way as in the training phase without the presence of tags. The sequence of words were converted into sequences of features and provided to the trained model for tag sequence prediction. Finally, the predicted tag sequence was harmonized with the input word sequence.

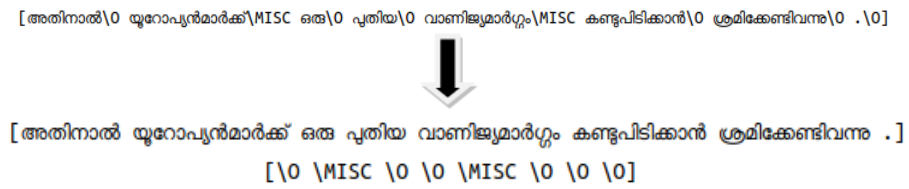


Figure 5.5: Preprocessing

['word.lower=പുതിയ', 'word[-5:]=പുതിയ', 'word[-4:]=ുതിയ', 'word[-3:]=തിയ', 'word[-2:]=ിയ',
'word[0:2]=പു', 'word[0:3]=പുത', 'word.isdigit=False', 'word[pos]=\\JJ', 'word[pos_prev]=\
\QT_QTO', '-2:word.lower=യൂറോപ്യന്\u200dമാര്\u200dക്ക്', '-1:word.lower=ഒരു',
'+1:word.lower=വാണിജ്യമാര്\u200dഗ്ഗം', '+2:word.lower=കണ്ടുപിടിക്കാന്\u200d']

Figure 5.6: Feature set for a single word

5.4.3 Experiments and Results

For training, Pycrfsuite, a python implementation of Conditional Random Field [97] was used. Fine tuning the model parameters is an important step in the training phase. We fixed the

coefficient of 'L1' penalty as '1.0' and L2 penalty as '1e-3'. Training was conducted on 80% of the total data and testing was performed on the remaining data. Training was stopped after 50 epochs and model file was saved. The proposed system was tested with two datasets, namely pre-evaluation and final evaluation datasets. The saved model predicted the tag sequence for each sentence. An accuracy of 97.44% was obtained with CRF.

5.5 NER using Deep learning

5.5.1 Introduction

No works were reported in Malayalam NER using deep learning techniques. But deep learning has proved its key role in various sequence labelling tasks such as POS tagging, NER, time series prediction, etc. Hence, it was decided to go for an approach that employs deep learning techniques for the entity recognition task. Deep learning techniques are characterized by its hierarchical feature learning nature. They can learn complex representations from data that can lead to state-of-the-art results in various classification tasks. In languages like English and German, the state of art results in NER are obtained from deep learning based systems. In addition to the conventional deep learning architecture, language-specific features were also incorporated to get improved accuracy. Training and testing were conducted using the shared corpus from ARNEKT solutions [108].

5.5.2 Architecture

Traditional neural networks have been outstanding over the last decade. Still, they are not up to the mark for sequence labelling tasks, where the current label is affected by its previous labels. They consider all its inputs and outputs as mutually independent. But, sequence to sequence learning implemented using deep neural networks is a hopeful solution for this problem. Deep neural networks have an amazing capability of considering the entire context in a sequence when dealing with sequence labelling tasks. Bi-LSTM-CRF-a well-known solution for sequence labelling problems was employed for named entity tagging.

In Indian languages, a word is often constituted by a set of morphemes corresponding to that language. Therefore, it is unsuitable to consider a complete word as a processing unit. Hence, it was decided to consider affixes from both ends of the word as additional features to represent the word. The methodology used was similar to [111], where the character level, word level and affix level features were combined to represent a complete word. The proposed architecture differed from other models by the way in which character level and word level features were created. In this work, CNN (convolutional neural network) with various kernel sizes was used to generate the character level word embedding. The affix level features were generated by selecting the frequent affixes from a general corpus corresponding to each language. The proposed word vector generation model can be straightforwardly applied for any morphologically rich language.

5.5.2.1 Generalized word representation

The pre-trained word embedding models suffer from the Out of Vocabulary (OOV) problem, where the word embedding vectors may not be present for words that are absent in the training data. The percentage of various named entities present in the training data over the FastText [112] word embedding file is given in table 5.4. The overall missing rate of words is 4.17%. This invites the need for a generalized word representation, which can effectively avoid the OOV problem confronted by the pre-trained word representation models. The generalized word representation model proposed in the work is shown in figure 5.7. The proposed word representation model combined features from various aspects of a word namely-character level, affix level, and word level. The character level features were formed using CNN's with different kernel sizes. The word level features were generated using pre-trained word embedding models. The affix level features were created using the most frequent affixes in a language. Finally, all these vectors were concatenated to form the final word representation.

Table 5.4: Percentage of the training set entities present in FastText word embedding files.

Language	Event	Things	Org	Occupation	Name	Location	Other	Average Presence
Hindi	99.69	99.33	99.23	99.48	94.96	98.91	96.38	98.28
Kannada	98.85	97.11	96.85	96.92	89.17	96.94	89.4	95.03
Malayalam	94.86	96.65	97.17	95.72	90.71	96.52	86.14	93.96
Tamil	98.34	98.3	97.95	96.93	91.72	95.13	93.05	95.91
Telugu	98.9	99.16	98.72	98.72	83.65	99.15	93.48	95.96
Class Avg.	98.12	98.11	98.00	97.55	90.04	97.33	91.69	95.83

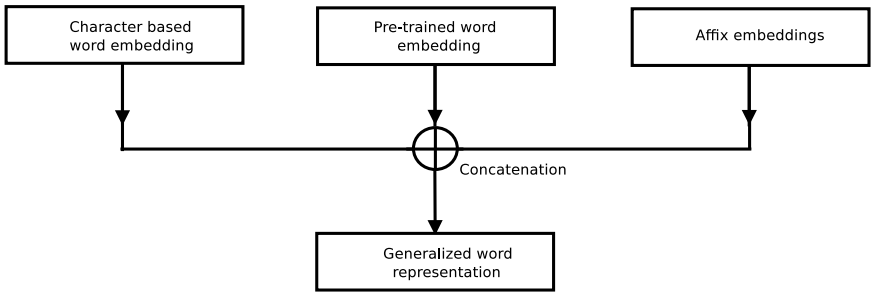


Figure 5.7: Generalized word representation

5.5.2.2 Convnet based character level word representation

Character level features are proved to be strong for entity recognition tasks [113–115]. Convolutional kernels with various filter sizes were used to generate the character level word representation. These representations are expected to capture the subword information present in words. Indian languages are characterized by their lengthy subword units. For this reason, convolutional filters of different kernel sizes were used. Subsequently, the outputs of different convolutional kernels through max-pooling layer were concatenated to form the character based word representation. Figure 5.8 shows the architecture of the character based word representation model.

Given a word 'w' composed of 'm' characters $c_1, c_2, c_3, ...c_m$, where $c_i \in V_c$ is the character vocabulary set. Let $C_1, C_2, C_3, ...C_m$ be the character embedding vectors that encode the characters $c_1, c_2, c_3, ...c_m$ present in a word 'w'. The character embeddings were obtained by matrix-vector product as given in equation 5.4.

$$C_1 = W_c V_c \tag{5.4}$$

Where W_c is the embedding matrix, $W_c \in R_{d_c * |V_c|}$ and V_c is the

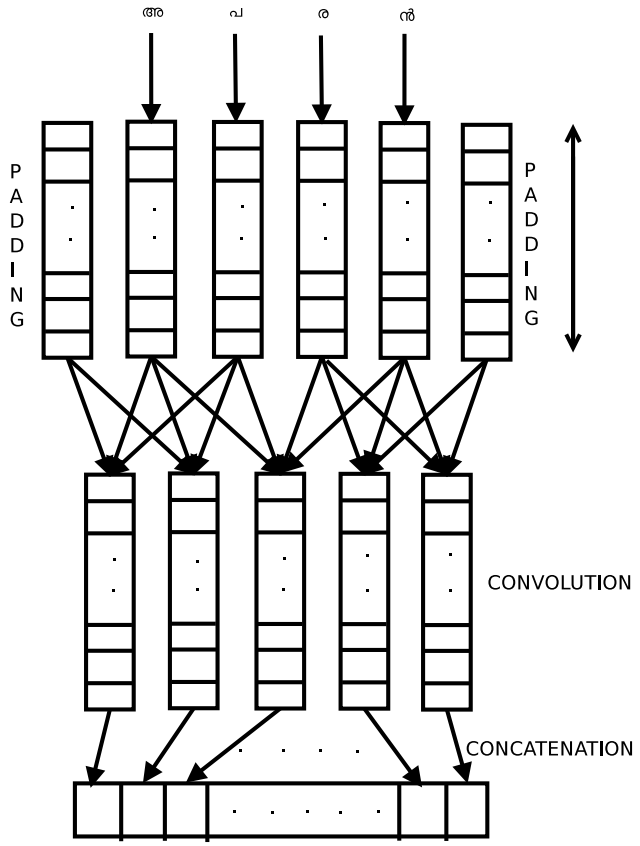


Figure 5.8: Architecture of the character-based word composition model

one-hot vector representation of a particular character. 'd_c' is a hyperparameter corresponding to the size of the character embedding. Hence, each word was transformed into a sequence of character embeddings $C_1, C_2, C_3, ...C_m$. Convolutional kernels were applied to each of the sliding context window of size 'k'. The resulting vectors are passed through a max-pooling layer to generate the maximum value. Then these vectors from different convolutional kernels were concatenated to produce the required word embedding vectors. These vectors are expected to capture information from different n-grams of the same word.

Formally, the character-level embedding of each word 'w' is calculated as follows,

$$V_c = \underset{1<i<m}{max} [W_{conv} Z_m + b_{conv}] \tag{5.5}$$

Where W_{conv} and b_{conv} are parameters of the model and Z_m is the concatenation of character embeddings expressed as

$$Z_m = \left(C_{m-(k-1)/2}, ..., C_{m+(k-1)/2} \right) \tag{5.6}$$

The convolution operation was applied to find the simple patterns of embedding vectors of different n-grams over the character sequence. Among different n-grams the following max-pooling layer try to extract position invariant n-gram features. Therefore, the character-based word composition model was expected to detect unvarying local spelling features from the character sequence.

5.5.2.3 Pre-trained word representation

Word embeddings are proven tools for capturing the context of a word along with its syntactic and semantic similarities [98]. They are the representation of words in n-dimensional space. They are also well known for modelling the relationship with other words in a corpus. They are typically created by applying a large collection of unannotated text over shallow neural networks through an unsupervised process such as CBOW model. In this study, FastText, a library for efficient learning of word representations was used for each language in the training set [112]. FastText, an extension to Word2vec by Facebook, can

efficiently represent words from their n-gram units. The embedding vector for a particular word is generated as the sum of all its n-gram vectors.

5.5.2.4 Affix level word representation

To approximate the true affixes of a language, the most frequent n-gram prefixes and suffixes of words in each language were identified from an unannotated corpus (specific to each language). Since frequent n-gram affixes are likely to behave like the true morphemes of the language, it was decided to learn a task-specific representation for them. To identify the n-gram size and the threshold frequency of affixes, various combinations of n (n-gram size) and T (threshold frequency) such as n=2,3,4,5 and T=100, 300, 500, 750, 1000 were experimented. The best results from the experiments were obtained, when the n-gram size was 3 and the threshold frequency was 500. Before training, the true affixes present in the training data were identified using a dictionary lookup method with the identified affixes (from unannotated corpus). During training time, the affix embeddings were initialized randomly and later tuned to learn a task-specific semantic representation. Finally, the individual representations were concatenated to construct a full vector representation for each word.

Formally, consider a Malayalam sentence $S_{[1:n]}$ that is a sequence of 'n' words, where 'n'is the maximum length of the sentence. In experiments, it was limited to 30 for the comfort of computation. Each word 'w'_i in the sentence was converted into a vector constituted by a composition of vectors corre-

sponding to that word. Hence, each word in the sentence was replaced by a vector of d-dimension, where $d = d_c + d_w + d_a$ such that d_c = dimensionality of character-level word embedding, d_w = dimensionality of pre-trained word embedding and $d_a = d_s + d_p$ is the dimensionality of affix embeddings, where d_s=dimensionality of suffix embedding and d_p= dimensionality of prefix embedding. Equations 5.7 and 5.8 shows the mathematical representation of a single sentence.

$$S = [w_1, w_2, w_3, w_4, ...w_n] \tag{5.7}$$

$$S = [[v_{c1}, v_{c2}, v_{c3}.....v_{cp}, v_{w1}, v_{w2}, v_{w3}.....v_{wq}, v_{a1}, v_{a2}, v_{a3}.....v_{ar}]_1, [v_{c1}, v_{c2}, v_{c3}.....v_{cp}, v_{w1}, v_{w2}, v_{w3}.....v_{wq}, v_{a1}, v_{a2}, v_{a3}.....v_{ar}]_2, ...] \tag{5.8}$$

where $v_{c1}, v_{c2}, v_{c3}.....v_{cp}$ corresponds to character-based word vector, $v_{w1}, v_{w2}, v_{w3}.....v_{wq}$ corresponds to pretrained word vector and $v_{a1}, v_{a2}, v_{a3}.....v_{ar}$ corresponds to affix embedding vector.

Figure 5.9 shows the schematic diagram of the proposed system. The complete architecture is similar to the one used for POS tagging discussed in the last chapter, with the main difference in the way in which the words are represented and provided to the network. The model accepts sequences of length 'n'. 'Wp', 'Wc', 'Wa' are pre-trained word embedding, character-based word embedding and affix embedding respectively. P0, P1, P2, etc. are the emission scores coming from the Bi-LSTM layer. Each of them indicates the probability of a particular tag

for a particular word. Hence, ‘m’ corresponds to the maximum number of tags in the tag set. Final predictions are made by the CRF layer based on these probabilities.

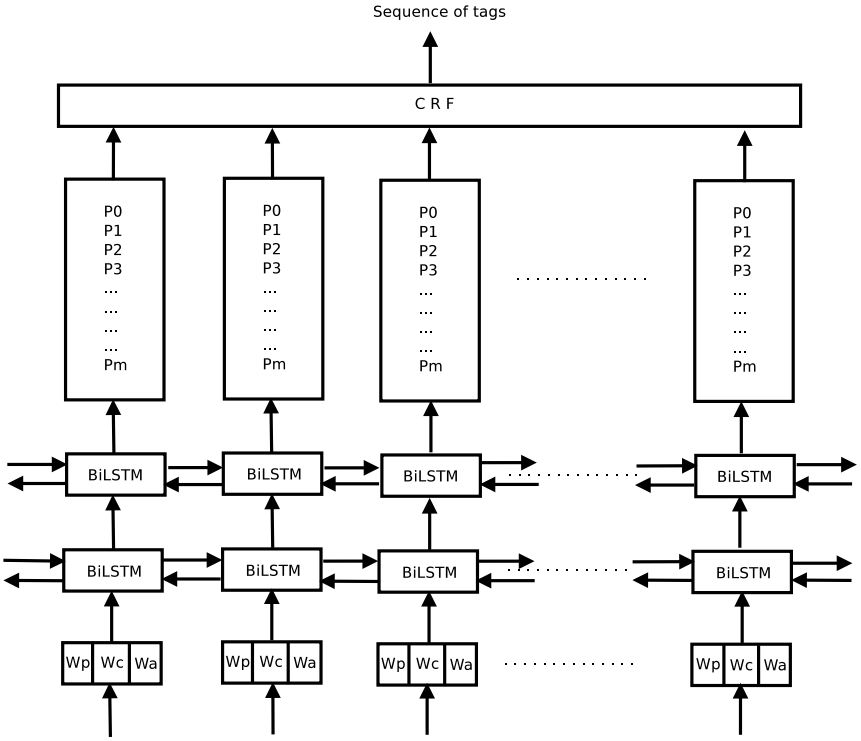


Figure 5.9: Architecture of the deep learning based NER system

Functional API of Keras was used to create the network [99]. The network contained three modules. First one was the input module, which receives the concatenated feature vector. The second module consisted of a set of hidden layers, where the output of each layer was provided to the successive hidden layer and finally to the output layer. ’Tanh’ was used as the activation in hidden layers. The mathematical simulation of ’Tanh’ for a sample input x is given in equation 5.9. Dropout was used to prevent overfitting. Python language was used for coding with Tensorflow in the backend.

$$tanh(x) = (2\sigma(2x) - 1), where$$
$$\sigma(2x) = \frac{e^{2x}}{(1 + e^{2x})} \tag{5.9}$$

5.5.3 Experiments and Results

The preprocessed tagged text was used for training. The tagged text was converted into sequences of words and tags. Here the first sequence corresponded to the input sequence and the second sequence was the output sequence. Words were converted into vectors of size 300 using FastText word embedding file [112]. Words that were not present in the vocabulary (OOV) were handled using a dummy vector. The final feature vector was generated by adding character-based word embedding and affix embedding to the pre-trained word embedding. Hence, each word was replaced by the concatenation of 150-dimensional character-based word embeddings, 60-dimensional affix embeddings and 300-dimensional pre-trained word vectors. The dimensionality of the feature vector was 510, where 210 was contributed by the combination of character level and affix level word embeddings. Finally, the input vectors of dimension 30*510 and the output vectors dimension 30*9 were sent to the Bi-LSTM network for training. Nine named entity classes were available in the training data.

The deep learning model of Keras is a linear stack of layers. Information about the input shape was provided in the first layer. The remaining layers can do automatic inference about the shape. Adam, an extension to the stochastic gradient descent

algorithm, was used for optimization task. 'categorical cross entropy' was used as the loss function. The network was compiled using Tensorlflow in the backend. Network weights were updated in batches of size 100. The training data consisted of 51000 sentences and testing data consisted of 12000 sentences. Training was conducted for 20 epochs and the model file was saved.

5.5.3.1 Impact of network parameters

Fine-tuning the network parameters is very important in deep learning paradigms. It was empirically found that the best architecture for the model is a 4-layered architecture. The first layer was the input layer, which received the concatenated feature vectors. Second and third layers were BiLSTM layers with 600 neurons and 'Tanh' activation. The last layer was the CRF layer, which makes the prediction. Increasing the number of neurons in the hidden layer was not promising. Until 600, there was a considerable improvement in performance. Hence, we finalized our hidden layer size to 600(300 in forward direction and 300 in backward direction). Different activations such as 'ReLU','Sigmiod'and 'Tanh'were also tried in hidden layers. But 'Tanh'seemed to be the optimum choice.

5.5.3.2 Impact of character-based word embedding

In order to assess the effectiveness of the proposed combined word representations, the same system on different individual word representations were compared. Combining the character-

level word composition vector to pre-trained word vector seems to capture character-level features of the word in addition to word-level features. Table 5.5 presents the impact of different word representations on Bi-LSTM-CRF tagger. This includes the performance of the tagging model on individual word representations and combined word representations. The character-based word representations could improve the accuracy of word-based models by 1.12%. This indicates the significance of character based word representations.

Table 5.5: Impact of different word representations on BiLSTM-CRF tagger Accuracy(%).

Representation	Hindi	Kannada	Malayalam	Tamil	Telugu	Average
FastText	96.87	96.41	96.68	96.22	96.66	96.57
FastText+char_ConvNet	97.98	97.30	97.76	97.61	97.84	97.69
FastText+char_ConvNet+Affix	**98.44**	**97.62**	**98.25**	**98.35**	**98.41**	**98.21**

5.5.3.3 Impact of affix embeddings

Since Indian languages are morphologically rich, the use of affix level features seems to be very effective in improving the overall performance of NER systems. The analysis implied that appending affix level features to the word level features can capture the inflectional characteristics of agglutinative words. From table 5.5, it is clear that affix based features can improve the overall performance. When added to the word level features, the affix level features could improve the results by .52%. An example sentence tagged using the proposed Bi-LSTM-CRF model is shown in figure 5.10 and figure 5.11. Here the correct tag for the second word is 'location', which was obtained only after incorporating the affix level features to word-level

features (figure 8).

രാമൻ\name ആലുവയിൽ\other കല്യാണത്തിന്\other പോയി\other .\other

Figure 5.10: Sample tagged text without incorporating the affix-level features

രാമൻ\name ആലുവയിൽ\location കല്യാണത്തിന്\other പോയി\other .\other

Figure 5.11: Sample tagged text by incorporating the affix-level features

5.5.3.4 Performance comparison for Malayalam

The results of the proposed system were compared with the different existing results in the domain. Among the reported results, the best performance was from Amrutha University Coimbatore [66]. Their work was reported as part of FIRE-2014. But the proposed system was able to produce improved results in the area of NER for Malayalam language. The comparison of performance across different works in this area is given in Figure 5.12.

5.6 Summary of the chapter

This chapter presents a detailed description of the preparation of a named entity tagged corpus and three different mechanisms for NER in Malayalam. The exclusive feature of the proposed systems is their performance in comparison with the existing methods in the domain. Different features that contribute to the entity recognition process were iteratively experimented. They include word embedding of the target word, suffix embedding

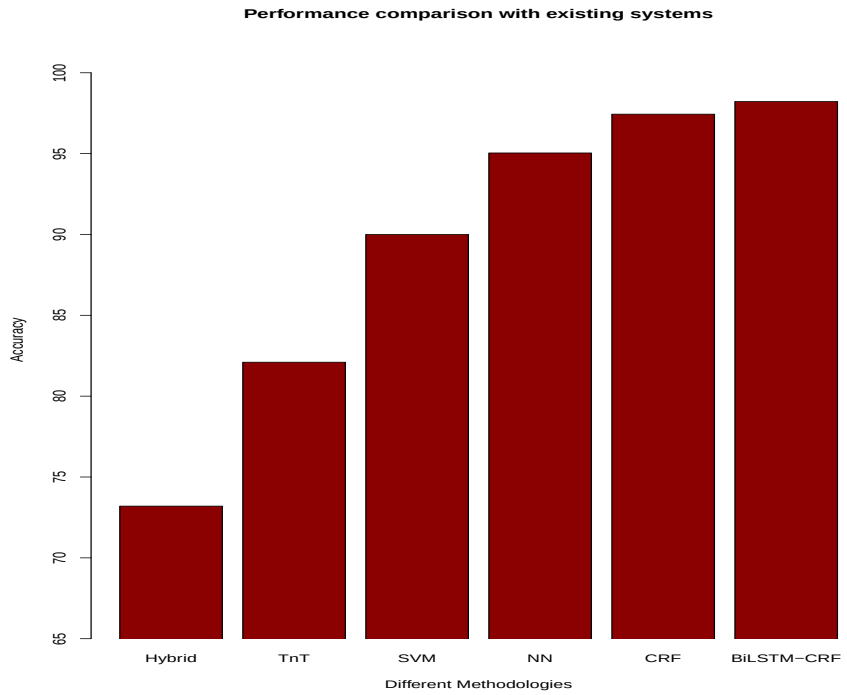


Figure 5.12: Performance comparison with the existing systems

of the target word, POS information of the target word and surrounding words, and word embeddings of the context words. Among the proposed systems, the deep learning based entity recognition system was the most promising one in terms of accuracy. Morphological features of Malayalam language was effectively utilized in all the above systems with the help of suffix stripping rules. Incorporating morphological information of words appears to be a promising thought for languages like Malayalam. Performance of all the systems can be improved by increasing the training data size, which is the most tedious job in the research. The proposed methods can also be used for various sequence labelling tasks in NLP such as phrase chunking, semantic role labelling, etc.

6

Deep level tagging

In this chapter, a deep level tagging methodology for Malayalam text is attempted using machine learning and rule-based techniques. Deep level tagging is the process of assigning deeper level information to nouns and verbs in a text document along with normal POS tags. It is one of the preliminary steps in the automatic analysis and representation of natural language text. Since Malayalam is a morphologically rich and agglutinative language, effective utilization of morphological features is essential for the computational analysis of the unstructured text. In this study, an in-depth analysis of nouns and verbs is conducted which can be effectively utilized for higher level tasks such as sentiment analysis, anaphora resolution, text summarization, etc. The analysis of nouns includes identifying the number, gender and case details. The analysis of verbs includes determining tense, aspect and modality details associated with them. Both machine learning and rule-based algorithms are employed for the proposed system.

Section 6.1 of this chapter presents a brief introduction about deep level tagging of Malayalam text. The necessity of such a system for the analysis of natural language text is also discussed in that section. Section 6.2 gives a detailed description

of the architecture of the proposed system. A detailed overview of the subsections of the schematic diagram is given in that section. Sample texts showing the input and output of each module is also given there. Section 6.3 discusses the experimental part of the proposed system. Finally, section 6.4 concludes the chapter with an overall summary of the work.

6.1 Introduction

The amount of natural language text over the internet is increasing day by day. Computational analysis of this text is essential for information extraction. Information extraction is a branch of artificial intelligence that deals with extracting meaningful facts from unstructured text. Deep level tagger is a middle way technology towards information extraction which assigns deeper level information to each noun and verb in a text document along with normal POS tags. Anaphora resolution, automatic text summarization, semantic graph construction, etc. are some of the application areas of deep level tagging systems. Identifying the number and gender information from nouns help us to resolve anaphors present in the text. Since anaphors present in a discourse refers to a noun which in turn agree with the number and gender of the anaphor, resolving PNG information associated with the nouns is of utmost importance in discourse analysis. Similarly, machine translation systems require adequate knowledge about the subject and object in a sentence. Case information associated with the nouns can provide enough linguistic cues about the subject and object in a sentence. Moreover, sentiment analysis systems can make use of the 'TAM' information

associated with verbs.

Automatic analysis of verbs and nouns in sentences is an essential task for the computational understanding of the natural language text. Different studies are conducted to analyze the morphology of Malayalam words [116–121]. Morphological analyzers take one word at a time and analyze its structure, syntax, and morphological properties [122]. Identifying the morphological properties of agglutinative words is a challenging task. However, it does not contribute much to the semantic understanding of the document. Here comes the advantage of deep level taggers. Deep level taggers are tools that help to process the text in a semantically meaningful manner. It considers all the nouns and verbs in a document and generates an in-depth analysis, which can be effectively utilized for higher end tasks such as anaphora resolution, text summarization, sentiment analysis, etc. The in-depth analysis of nouns includes capturing the number, gender and case information associated with them. Whereas, the in-depth analysis of verbs includes capturing the tense, aspect and modality information associated with them.

The dataset required for various tasks in deep level tagging was provided by the researchers from Thunjath Ezhuthachan Malayalam University. A detailed study on 'TAM' information associated with verbs in Malayalam was conducted there. Malayalam verbs can be classified into 25 categories according to the 'TAM' information associated with it [123]. Table 6.1 shows different classes of verbs identified during the study. It was found that the verbs in the same class always share the

Table 6.1: Different classes of verbs according to 'TAM' analysis

Verb Label	Class name
Verb_PAST	Past Tense
Verb_PRES	Present Tense
Verb_FUT	Future Tense
Verb_mood_IMPR	Imperative Mood
Verb_mood_CMPL	Compulsive Mood
Verb_mood_CMPL_NEG	Negative Compulsive Mood
Verb_mood_PROS	Promissive Mood
Verb_mood_PRMS	Permissive Mood
Verb_mood_OPT	Optative Mood
Verb_mood_PRCT	Precative Mood
Verb_mood_PRCT_NEG	Negative Precative Mood
Verb_mood_DSRV	Desiderative Mood
Verb_mood_ABLT	Abilitative Mood
Verb_mood_IRLS	Irrealis Mood
Verb_mood_PURP	Purposive Mood
Verb_mood_COND	Conditional Mood
Verb_mood_STSF	Satisfactive Mood
Verb_mood_MONI	Monitory Mood
Verb_aspect_PROG_SAT	Progressive Stative Aspect
Verb_aspect_PROG_INS	Progressive Instantaneous Aspect
Verb_aspect_PROG_ITR	Progressive Iterative Aspect
Verb_aspect_PRF_SMPL	Simple Perfect Aspect
Verb_aspect_PRF_CTP	Contemporaneous Perfect Aspect
Verb_aspect_PRF_RMT	Remote Perfect Aspect
Verb_aspect_HBTL	Habitual Aspect

same set of suffixes. Hence, it was decided to capture the suffix level features for identifying the verb class. A suffix stripping algorithm with the help of a set of suffix stripping rules was used for this purpose. The case information associated with the nouns was identified using a rule-based methodology. A set of possible cases and associated affixes are given in table 6.2.

6.2 Architecture

Figure 6.1 demonstrates the general block diagram of the proposed deep level tagging system. The proposed architecture

Table 6.2: Case markers and corresponding suffixes

CASE	SUFFIXES
Accusative	e
Sociative	odu
Dative	kku , nu
Genitive	aal
Instrumental	ude, nte
Locative	il
Nominative	All remaining suffixes

contains four phases. The first phase is the preprocessing phase, where the input text is segmentized into sentences and words. The second phase is the POS tagging phase, where the words from the input text are tagged with POS information. B.I.S tagset is used to tag the words from the input document. This tag set is developed by the POS tag standardization committee nominated by the department of information technology, Government of India. It exploits the linguistic hierarchy between different classes of words. The third phase is the named entity recognition phase. This phase helps to identify the person entities from the set of nouns in the tagged text. The final phase of the architecture is the deep level tagging phase, where the deep level information associated with nouns and verbs are explored. Working of each phase of the complete architecture is presented in the following sections.

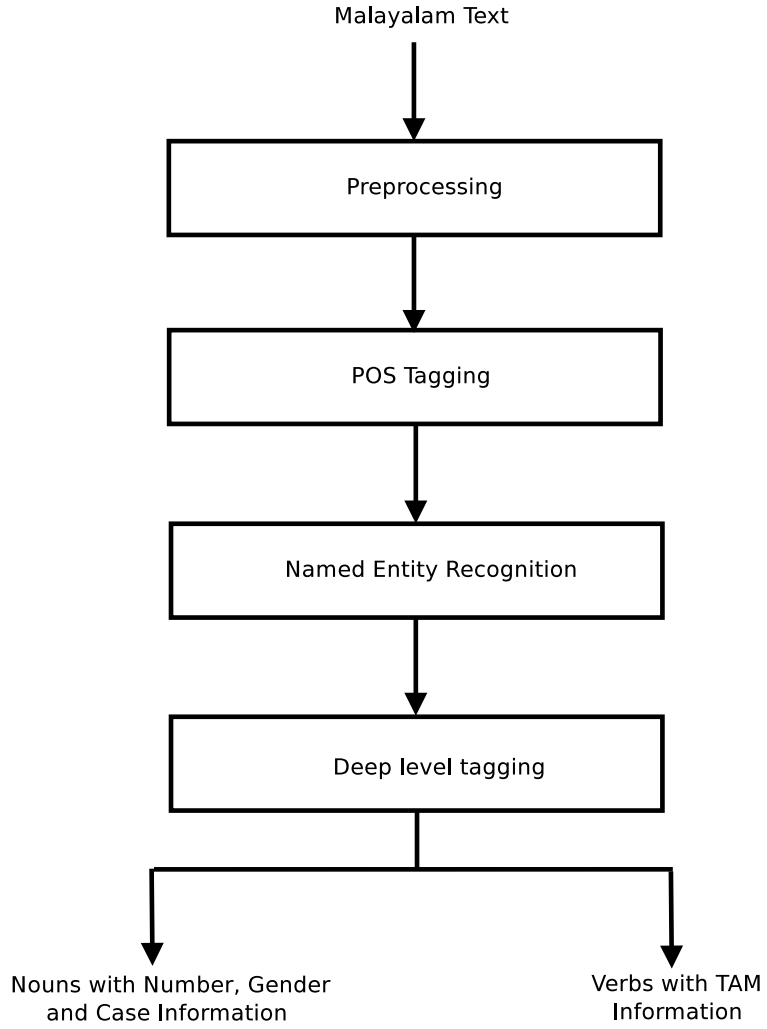


Figure 6.1: General architecture of deep level tagging system

6.2.1 Preprocessing phase

Preprocessing is the first phase of the architecture which converts the raw Malayalam text into sentences and words. It is the phase which converts the raw input text into a form that can be easily handled by machine learning algorithms. Regular

expressions and NLTK tools were used for realizing this phase.

6.2.2 POS tagging phase

POS tagging is an important step in most of the NLP applications. It assigns the words in a document to a particular grammatical category based on its meaning and context. The preprocessed Malayalam text is provided to the POS tagging module. In this study, a deep learning based implementation of POS tagger discussed in section 5 of chapter 4 [106] was used. BiLSTMs are well-known tools for labelling sequential data. B.I.S tagset was used for tagging individual words. Hence, the maximum number of possible tags was from a limited tagset of 36. A piece of text, tagged using the above-mentioned tagger is given in figure 6.2.

```
[['സ്റ്റാറ്റിൻസ്\\N_NN', 'എന്നത്\\CC_CCD', 'ഹൃദയസംബന്ധിയായ\\N_NN', 'രോഗങ്ങളുള്ളവർക്ക്\\N_NN'
'സാധാരണയായി\\RB', 'നൽകിവരുന്ന\\V_VM_VNF', 'മരുന്നാണ്\\V_VAUX', '.\\RDPUNC'], ['എന്നാ
\CC_CCD', 'കഴിഞ്ഞ\\JJ', 'കുറച്ചകാലമായി\\N_NN', 'ഇതിന്റെ\\DM_DMD', 'ഉപയോഗത്തെയും\\N_NN',
'പാർശ്വഫലങ്ങളെയും\\N_NN', 'സംബന്ധിച്ച്\\PSP', 'വിവിധങ്ങളായ\\JJ', 'വാദപ്രതിവാദങ്ങളാണ്\
\V_VAUX', 'വൈദ്യശാസ്ത്രരംഗത്ത്\u200c\\N_NN', 'നടന്നവരുന്നത്\\V_VM_VNF', '.\\RDPUNC']]
```

Figure 6.2: Sample text tagged using deep learning based POS tagger

6.2.3 Named Entity Recognition phase

Named entity recognition (NER)is a preliminary step towards information extraction that aims to locate and classify the named entities, including persons, location, organizations, etc. present in a text document. All the nouns present in a text may not contain number and gender details. Therefore we should identify the nouns for which in-depth analysis is to be performed.

Named entity recognition phase helps to identify the person entities from the POS-tagged text. The deep learning based entity tagger discussed in chapter 5 was employed for this purpose. Sample text showing the input and output of the second phase of the architecture is given in figure 6.3.

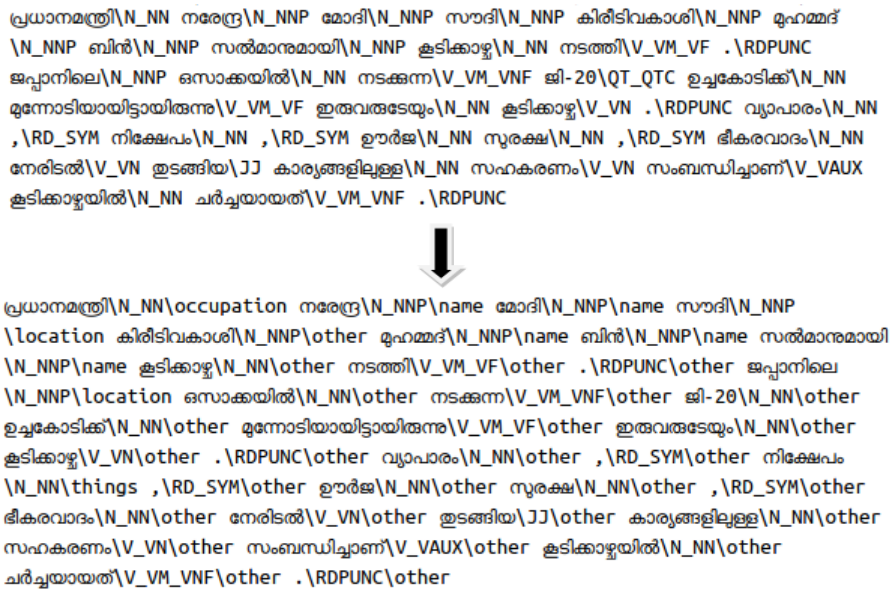


Figure 6.3: Sample text showing the input and output of the NER module

6.2.4 Deep level tagging phase

The final phase of the block diagram is the deep level tagging phase, which accomplishes the in-depth analysis of nouns and verbs. The in-depth analysis of person entities includes exploring the number, gender and case details associated with them. Similarly, nouns which are not tagged with person entity tag were sent to number and case identification module. On the other side, the in-depth analysis of verbs include exploring the tense, aspect, and modality details associated with them. Noun

words and verbs from the previous modules are provided as input to the deep level tagging module. The morphological richness of Malayalam language was utilized to explore the detailed information associated with verbs and nouns. A suffix stripping algorithm, along with a set of suffix stripping rules, was employed for this purpose. MLP classifier was used for the detailed study of verbs and nouns.

From the entity tagged text in the third phase, words with person entity tag were provided to the number and gender identification module where the suffixes of various length acted as the feature set for PNG identification. A suffix stripper was used to extract suffixes of different size, which acted as the feature set for the classifier. MLP was used to build the classifier [110]. MLP is a function approximator which can theoretically estimate any kind of function. Case information associated with the nouns were also explored in this phase using a rule-based technique. Similarly, verbs were provided to the 'TAM' identification module. Here also the suffixes of different length acted as the feature set. MLP classifier with 25 class labels was prepared for this purpose. The class labels itself indicated the 'TAM' details associated with the verb. Different configurations of the MLP classifier were experimented for both the tasks. A sample text showing the input and output of the complete system is given in figure 6.4.

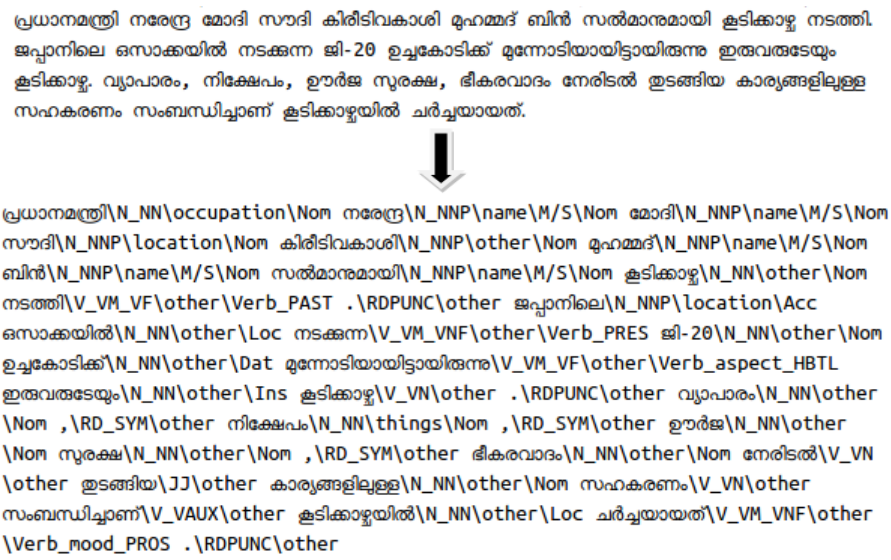


Figure 6.4: Sample text showing the input and output of the complete architecture

6.3 Experiments and Results

This section presents the experiments conducted on each phase of the proposed architecture. The first step is the preprocessing step, where the raw Malayalam text was given to sentence segmentation and word tokenization operations. The sentence segmentation operation was carried out using the NLTK word tokenizer [124]. The POS tagging module accepts the preprocessed Malayalam text and generates the POS-tagged text. The POS tagger gave a test accuracy of 94.33%, which appeared as a comparable performance for languages like Malayalam. The third phase of the proposed system deals with entity extraction. A deep learning based approach discussed in chapter 5 was employed for this purpose. The dataset used to build the entity tagger was the corpus provided by IECSIL-2018 organizers [108].

The final phase of the architecture is the deep level tagging

phase. The first part of the deep level tagging phase is the case labelling of nouns. A rule-based algorithm was used for case labelling. A set of suffixes corresponding to each case was stored in a lookup table. Each noun identified in the POS tagging phase was sent to a suffix extraction unit, where suffixes of length 2,3,4 and 5 were extracted and sent to the lookup table. If there existed any match between the extracted suffix and stored rule, the corresponding case was triggered. Otherwise, a nominative case was returned by the rule-based module.

The second part of the deep level tagging phase deals with the in-depth analysis of verbs and nouns. Here different classifiers were attempted to distinguish the best performing classifier on the dataset. Unlike in the earlier scenario, MLP outperformed the remaining classifiers on both the tasks ('TAM' analysis and number-gender analysis). Figures 6.5 and 6.6 illustrate this point. A list of 12600 names belonging to the different categories of names was prepared as the training data for number and gender identification classifier. Suffixes of length 1 to 8 were used as features for each name. Since machine learning algorithms require features as numeric values, these feature set (suffixes) was converted to numeric values using Dictvectorizer, a python functionality [125]. Dictvectorizer over Word2vec was chosen in this phase, since they were well suited in encoding categorical features with multiple possible values. Moreover, Word2vec is appropriate in situations where the syntactic and semantic roles of words are necessary. Dictvectorizer was employed in situations where the feature set is a list of dictionaries rather than a list of categorical items. The feature set was a list of dictionaries where each dictionary refers to a set of

suffixes corresponding to a single word. Thus, the total feature size was 7468.

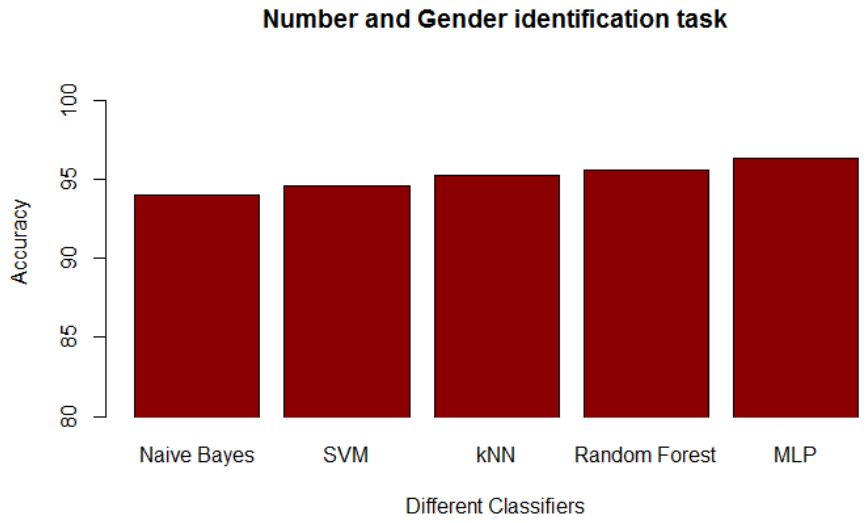


Figure 6.5: Performance of different classifiers on number and gender identification task

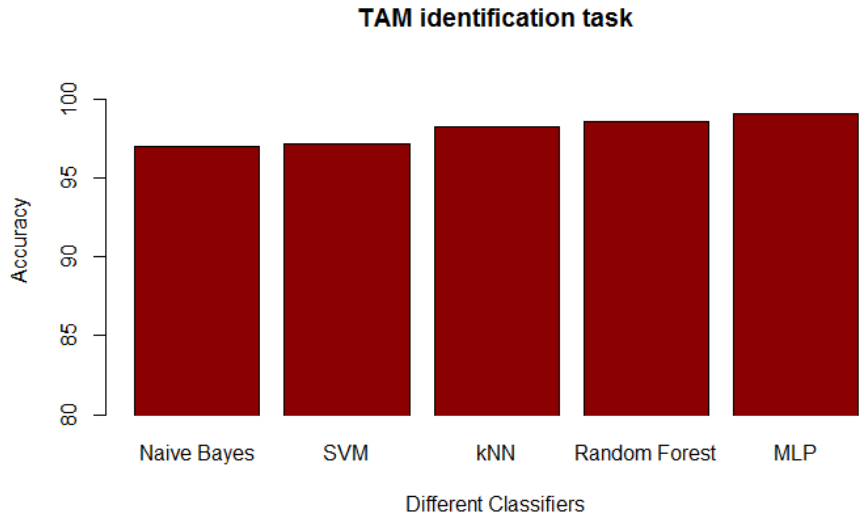


Figure 6.6: Performance of different classifiers on 'TAM' analysis task

Different configurations of the MLP classifier were attempt-

ed in the study. A smaller network was not able to represent the data efficiently and increasing the number of layers did not improve the accuracy significantly. Hence, it was experimentally finalized that the hidden layer configuration as (2*100), where 2 is the number of hidden layers, and 100 is the size of each hidden layer. 'Relu' was used as the activation function and 'Adam' as the optimizer. The performance of the number and gender classifier with the different number of features is shown in figure 6.7. From the figure, it is clear that the accuracy of the system increased with the increase in suffix features and the maximum accuracy was achieved when the number of features was 10. The maximum accuracy obtained by the classifier was 96.21%.

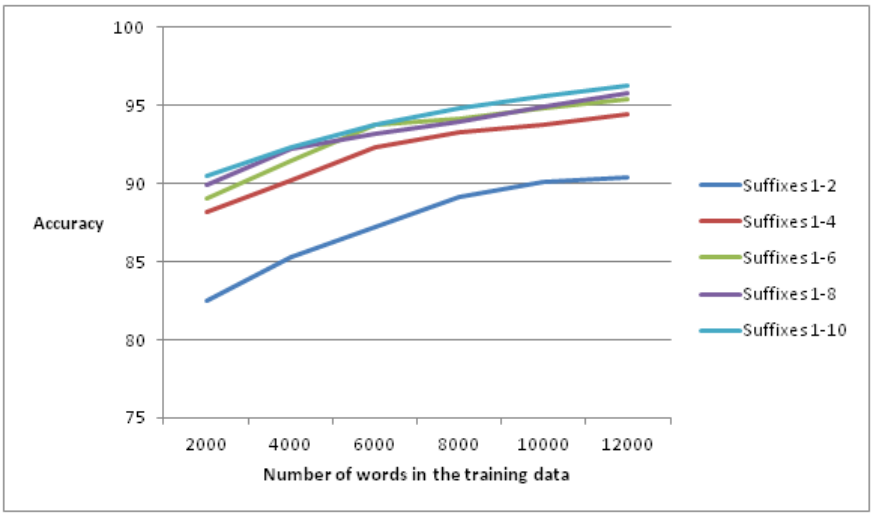


Figure 6.7: Effect of different features on the performance of the number and gender identification classifier

The configuration of the 'TAM' analysis neural network is not completely different from the number and gender neural network. The only difference is in the number of classes and training data. The training data contained 1205 verbs belong-

ing to 25 classes. Similar to the number and gender network, suffixes of different length were used as features. Here also, the feature vector was converted to a numeric representation using Dictvectorizer, a python functionality. A neural network with two hidden layers constituted the model. The parameters of the network were exactly the same as the number and gender identification neural network. The maximum performance obtained by the classifier was 99.17%. Performance of the 'TAM' analysis model on different sets of features is shown in figure 6.8. From the figure, it can be inferred that the accuracy of the model increases with the increase in suffix features.

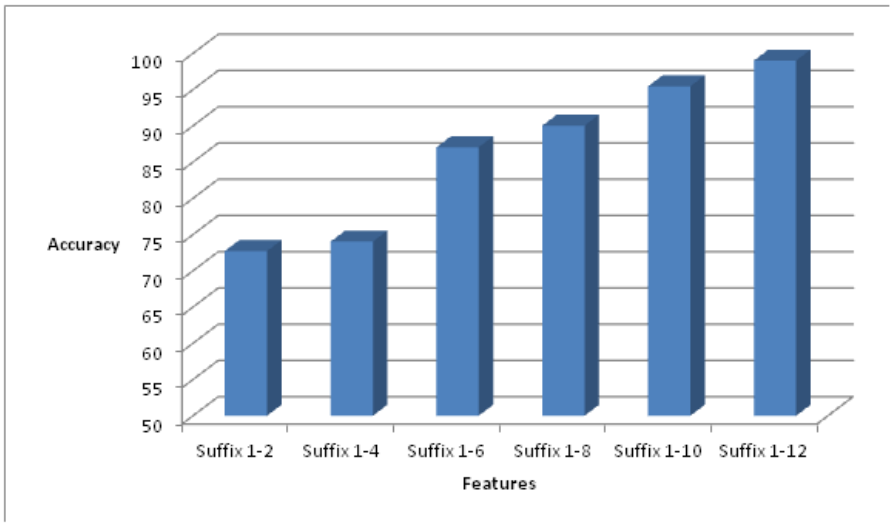


Figure 6.8: Effect of different features on the performance of the 'TAM' identification classifier

The training data required for all the experiments were prepared with the help of Malayalam University, Tirur. The final accuracy of the complete system is 90.2%. The detailed information regarding the overall performance of the proposed system is shown in table 6.3.

Table 6.3: Overall performance of the proposed system

Module-1	Training	Testing
Number of words	230371	57517
Accuracy	91.2%	
Module-2	Training	Testing
Number of words	838333	280130
Accuracy	97.44%	
Module-3-A (number and gender)	Training	Testing
Number of words	10080	2520
Accuracy	96.21%	
Module-3-B (TAM)	Training	Testing
Number of words	964	241
Accuracy	99.17%	
Overall accuracy	90.2%	

6.3.1 Analysis

To better understand the performance of the models on the constructed datasets, a detailed analysis was also performed. ROC curve-the best metric for evaluating the performance of any classifier was employed to evaluate the performance of each model. The area under the ROC curve represents the degree or measure of separability between different classes predicted by the classifier. Figures 6.9 and 6.10 show the ROC curves for the number-gender identification model and 'TAM' identification model respectively. All the models were relatively good in demonstrating the tradeoff between different classes across various settings of the classifiers. Nevertheless, it can be observed that the area under 'TAM' curve is mostly higher than the area under the curves of the other models. This is contributed by the expressive nature of suffix endings in Malayalam verbs.

To determine the contribution of suffixes of different length

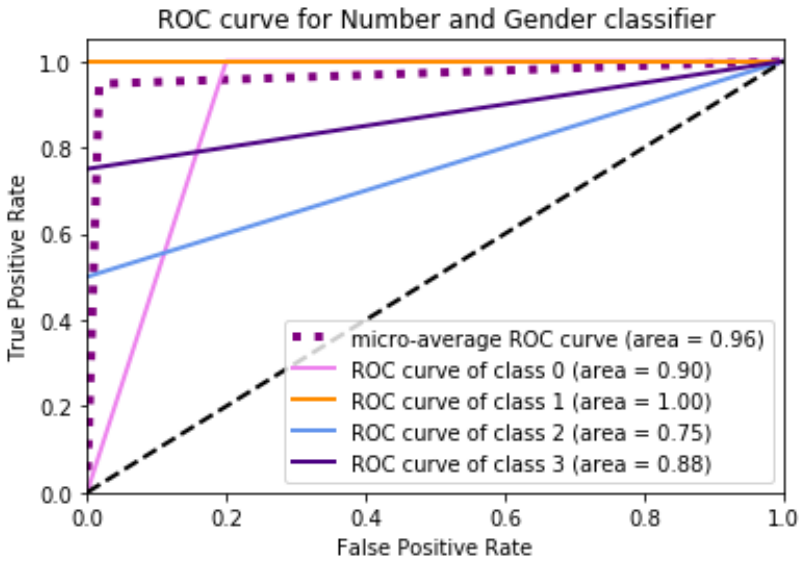


Figure 6.9: ROC curve of the number and gender classifier

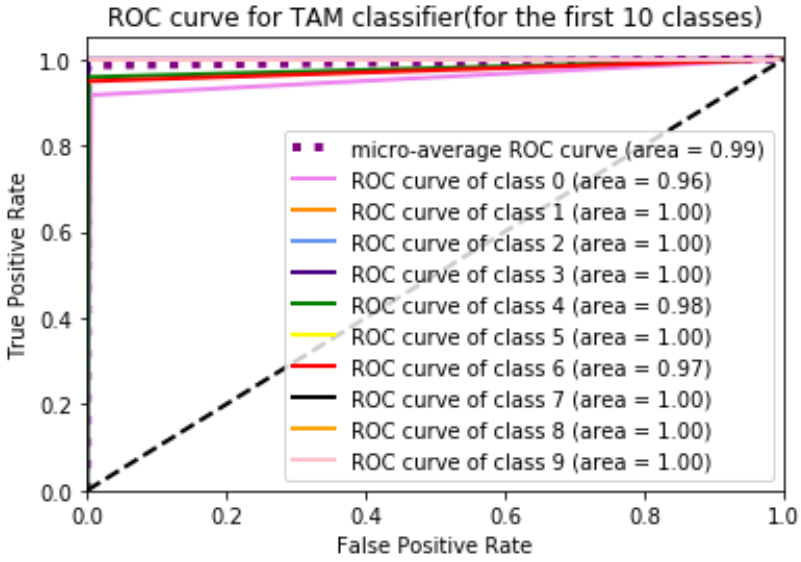


Figure 6.10: ROC curve of the 'TAM' classifier

(as compared to simply using a fixed length suffix) towards classification accuracy, we ran our model with suffixes of different length (ranging from 1 to 12). The performance of the mod-

els (in-depth analysis) without using the suffixes of different lengths were even lower than the proposed combined feature systems. This shows the impact of suffix level features on computational processing of Malayalam text.

Further, the effect of different classifiers on the training data was also verified. It was found that MLP-one of the best choice for multiclass classification problems outperformed the other classifiers. MLP was found to be the best choice for such tasks. Theoretically, MLP can estimate any function or equivalently able to find any mappings [126].

6.4 Summary of the chapter

This chapter discussed in detail about the development of a deep level tagger for Malayalam. The exclusive feature of the proposed system is the detailed analysis of nouns and verbs in a text document. This detailed information about nouns and verbs can be effectively utilized for the semantic understanding of the natural language text. Both machine learning and rule-based techniques were applied for the computational analysis of Malayalam text. In this study, entity extraction technique to recognize the person entities from POS-tagged text was used. Machine learning based techniques rather than traditional rule-based approaches were used in the work because of its scalability and convenience. The morphological richness of Malayalam language was effectively utilized for the developed system with the help of suffix stripping algorithms. The conversion of suffix level features to numeric values was done with the help of

python functionalities. The impact of deep level tagging system on different NLP applications like sentiment analysis, anaphora resolution, abstractive summarization, etc can be explored in future.

7

Pronominal Anaphora Resolution

This chapter presents a novel approach towards resolution of pronominal anaphors present in Malayalam text. Anaphora resolution is the process of identifying the antecedent of an anaphoric expression present in natural language text. Most of the NLP applications such as information extraction, question answering, text summarization, etc. require successful resolution of anaphors. Pronominal anaphors are a subclass of anaphors realized using short words called pronouns. In this study, we have used a deep level tagger discussed in chapter 6 as a pre-processor for the resolution of pronominal anaphora. The proposed methodology is a hybrid architecture employing rule-based and machine learning techniques. Two datasets were prepared to evaluate the proposed system. The performance of the proposed system was evaluated using those datasets.

Section 7.1 of this chapter gives a brief introduction to the problem of anaphora resolution. Section 7.2 presents a detailed description of the general architecture of the proposed system. Each part of the architecture is discussed in detail in that section. Section 7.3 outlines the experimental part of the methodology. Results obtained by the system on sample datasets are also given there. Finally, the chapter is concluded in section

7.4 with an overall summary of the work.

7.1 Introduction

Anaphora is the use of an expression which refers to another expression in the same document. The origin of the word anaphora goes back to the 16th century [127]. It consists of two units called 'ana' and 'phora', where 'ana' means back and 'phora' means carrying. Anaphors are commonly used to avoid repetition of an item in a discourse. These items are usually noun phrases representing real-world objects. But occasionally they can be verb phrases or sentences. Depending on the way in which different clauses of a sentence are linked, anaphora are of mainly three kinds, namely-pronominal, definite noun phrases and quantifiers [128]. Pronominal anaphora are realized using pronouns. Among the various types of anaphora, pronominal anaphora is the most widely used one, which is realized through anaphoric pronouns. Pronouns are short words that can be substituted for a noun or noun phrase. They always refer to an entity that is already mentioned in the discourse. Pronominal anaphors are always stronger than full definite noun phrases [129]. They are again classified into five, namely personal pronouns, possessive pronouns, reflexive pronouns, Demonstrative pronouns and relative pronouns. All the above-mentioned pronouns need not be anaphoric. Definite noun phrase are the second type of anaphors realized using noun phrases of the form $< the >< nounphrase >$. Here the antecedent is always referred using a definite noun phrase representing either same concept or semantically similar concept.

And the quantifier anaphors are realized using words like one, first, last, etc. Among various types of anaphora, pronominal anaphora is the commonly used one [129]. Hence, we decided to go for a system which can effectively resolve all the pronominal anaphors present in a Malayalam document.

It is a dream of humankind to communicate with the computers, since the beginning of the digital era. Communication with computers is only possible through a proper understanding of natural language text. The proper understanding of natural language text is possible only after anaphora resolution. There is a large amount of text data available online in Malayalam. Malayalam Wikipedia itself contains more than 30,000 articles. This warrants us to develop tools that can be used to explore digital information present in Malayalam and other native languages. Anaphora resolution is one such tool that helps in semantic analysis of natural language text. It is also necessary for areas like text summarization, question answering, information retrieval, machine translation, etc.

7.2 Architecture

The objective is to build a pronominal anaphora resolution system for Malayalam. The overall architecture of the proposed system is illustrated in Figure 7.1. It contains five successive modules, where each module performs a set of specific tasks. The first module is the preprocessing module, where the sentence segmentation and word tokenization operations are carried out. After preprocessing, the preprocessed text is chan-

nelled to the POS tagging module. In the POS tagging module, each word from the preprocessed text is assigned with the corresponding grammatical category. Similar to section 4.2, BIS tagset is used in this study. The POS tagged text is then provided to the Named Entity Recognition module, where the person, location and organization entities present in the tagged text are identified. Person entities are the most common antecedent of pronominal anaphora. They are the most compatible words to correlate with the pronouns in a text document. Identifying the person entities from the set of nouns in the text document is the most crucial part of the proposed system.

The fourth module of the architecture is the deep level tagging module, where the in-depth analysis (number and gender identification) of person entities is performed. Deep level tagging is performed as discussed in section 6.2.4 of the previous chapter. Finally, the deep level tagged text is provided to the anaphora resolution module, which is the core part of our architecture. The overview of anaphora resolution module is given in Figure 7.2. The first step in the anaphora resolution module is the identification of pronouns present in the text. Words with PRP (Pronoun) POS tag are collected and stored as the pronominal anaphora.

For each pronoun present in the set of pronominal anaphora, a set of possible candidates were shortlisted. Person entities present in the preceding four sentences of the anaphor which agree with the number and gender of the anaphor act as the likely candidate set. The possible candidates were assigned with a set of salience values corresponding to a set of salience factors.

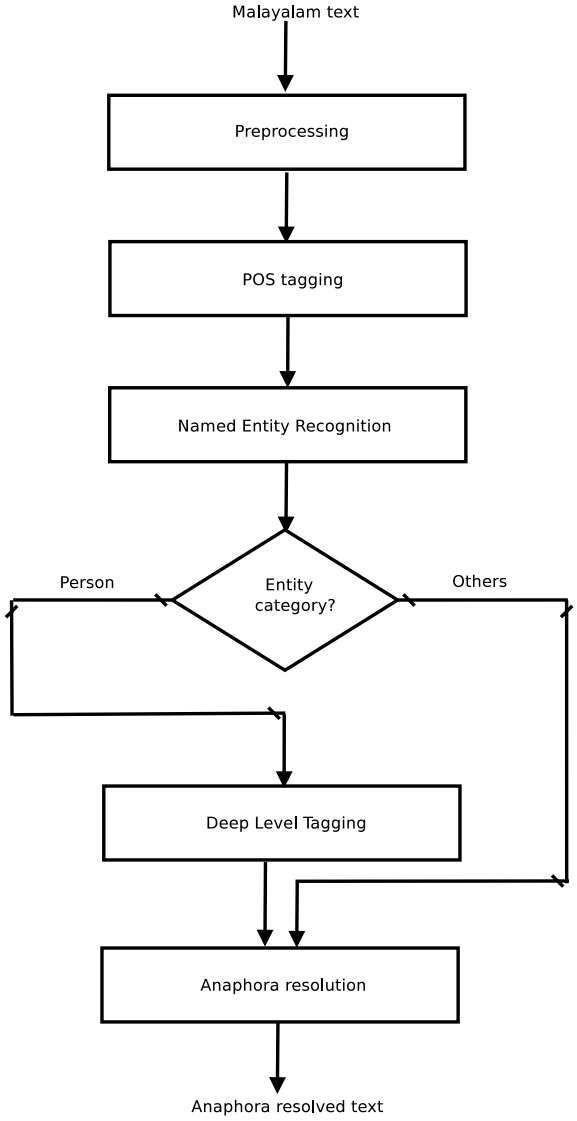


Figure 7.1: General architecture of the proposed system

Salience values were computed through the proper analysis of the Malayalam language. The sum of salience values indicate the possibility of a candidate being the actual antecedent. The set of salience factors and their corresponding weights are given in table 7.1. The salience factors considered in the study include sentence-recency, case information, syntactic role, clause

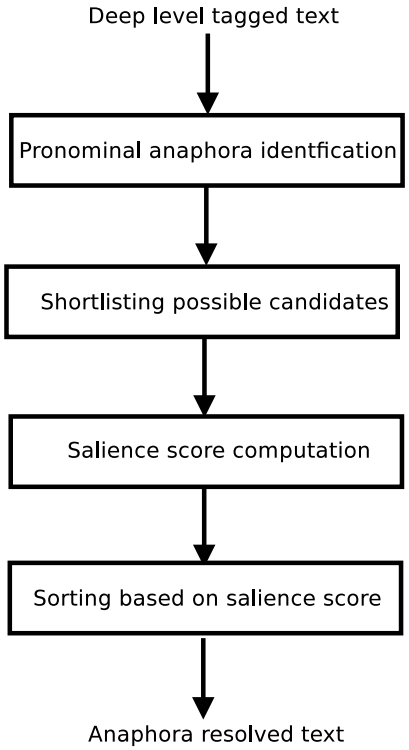


Figure 7.2: Architecture of the anaphora resolution module

information and semantic role.

The clause information and subject-object information of shortlisted candidates were identified using a rule-based technique. For that, all the sentences containing the shortlisted candidates were divided into clauses. The words with a verb or adjective POS tag act as the clause boundary in a sentence [130]. In the case of adjectives, the nouns following them were also included in the same clause. For each clause identified in the last step, the subject, object and predicate information were extracted using the following set of rules [131].

- Predicate —The verb or adjective present in the clause will

Table 7.1: Salince factors and their weights

Salience factors	Weights
Current sentence	100
Preceding sentences	Reduce sentence score by 10
Subject	80
Object	50
Case	10 to 40
Current clause	100
Immediate clause	50

form the predicate.

- Object —If the predicate is a verb, the noun preceding it will be the object and if the predicate is an adjective, the noun following it will be the object.
- Subject —If the predicate is a verb, the noun preceding the object will be the subject. And if there is no noun preceding the object in the same clause, follow the given rules.

  1. If the predicate of the preceding clause is an adjective, its object will form the subject.
  2. Else the subject of the preceding clause will be the subject.

  Conversely, if the predicate is an adjective, the noun preceding it will be the subject.

Finally, the possible candidates were sorted based on the total salience score and the candidate having the highest salience score was selected as the antecedent of the anaphor.

7.3 Experiments and Results

The preprocessing of raw Malayalam text was conducted using NLTK implementation of sentence tokenizer and word tokenizer [124]. Sentence tokenizer receives the raw Malayalam text and converts it to a sequence of sentences. The word tokenizer receives this sequence of sentences and converts each sentence into a sequence of words. The POS tagging module receives this sequence of words and generates the POS-tagged text. The deep learning based POS tagger discussed in section 4.5 was used for this purpose [106]. The POS tagged text was then given to the NER module. The person entities from the NE tagged text was provided to the deep level tagging module, where the number, gender and case information associated with the nouns were recognized.

To evaluate the performance of the proposed system, two different datasets were prepared and experimented. The experiments were also conducted to explore the possibilities of the proposed deep level tagger. The characteristics of the prepared datasets are given below.

7.3.1 Dataset 1

This dataset is a collection of short-stories that belongs to the children story domain. The short stories were downloaded from [132], a popular site for Malayalam short stories. Each of them consists of 20 to 30 sentences of reasonable length. Since the narrative style of children stories follow a straight forward ap-

proach, the structural complexity of the sentences was limited. Hence, the performance of the proposed algorithm on this dataset represents the baseline performance. Table 7.2 illustrates the performance of the algorithm on this dataset.

Table 7.2: Results of our experiments on dataset 1

Total sentences	Total pronouns	Correctly resolved	Accuracy
353	19	16	84%

7.3.2 Dataset 2

This dataset is a collection of text from news article domain. The articles were taken from [133], a popular site for news articles. Here, the structural complexity of the sentences was not limited. Each article consists of more than 20 sentences. The length of the sentences was not limited. Performance of the proposed system on dataset two is illustrated in table 7.3.

Table 7.3: Results of our experiments on dataset 2

Total sentences	Total pronouns	Correctly resolved	Accuracy
466	21	17	80%

The overall accuracy acquired by the proposed system is 82.5%. The correctness of the results was verified with the help of language experts. According to the observations, various factors that affected the performance of the proposed system were

- Free ordering nature of Malayalam language
- Case information associated with the nouns

- Use of the same pronoun to refer both animate and non-animate nouns
- Ability to distinguish person entities from non-person entities, etc.

7.4 Summary of the chapter

This chapter describes a novel idea towards the resolution of pronominal anaphora in Malayalam. Various factors that affect the resolution of pronominal anaphors are also studied in detail. The performance of the system is mainly decided by the accuracy of the NE tagger and deep level tagger. The selected salience factors and their values are globally accepted parameters. The exclusive feature of the proposed system is the use of machine learning as well as rule-based techniques. The proposed pronoun resolution algorithm is the best use case of the deep level tagger discussed in chapter 6. The current work is focussed only on the resolution of pronominal anaphors. In future, the resolution of non-pronominal anaphors, which is not explored in the literature so far can be carried out.

8

Conclusions and Future directions

8.1 Conclusions

This thesis is mainly focussed on the resolution of pronominal anaphors in Malayalam text using an integrated and hybrid approach. Malayalam, being an inflected and agglutinative language, presents many challenges, which results in the detailed analysis of different language level features that could improve the overall performance of the system. Different features that were considered in the study included word level features, character level features and affix level features. The word level features are extracted using pre-trained word embeddings, while the affix level and character level features are identified using feature inferring neural networks and handcrafted rules. Without losing generality, the proposed word representation model can be effectively applied for any morphologically rich language. Experiments were conducted to identify the impact of each feature and combined feature representations on POS tagging performance. The performance results gave an important insight about the impact of combined word representation over the individual ones. This is due to the inflectional characteristics of Indian languages as compared with other lan-

guages. Without using any form of external resources (lexicon, POS, etc.), the state of art performance in Malayalam NER and POS tagging were achieved. It was found that the combination of features improved the accuracy of tagging systems. Even though the experiments were mainly focussed on Malayalam, the proposed approaches could be applied for any of the Indian languages.

Deep learning techniques were used for different language processing applications, which were not yet explored in the case of Malayalam language. Deep learning techniques demand a large amount of training data and the dataset was prepared in-house. The authenticity of the data set was verified using standard evaluation metric such as Fleiss's Kappa coefficient.

A study on various popular machine learning techniques for POS tagging and NER has been carried out. Extensive statistical analysis was conducted to recognize the best performing algorithm for sequence labelling task. A novel word representation model was developed using convolutional neural networks and suffix separation rules. The accuracy of the proposed systems is evident from the results demonstrated in section 4.5 and 5.5. The performance evaluation was carried out in comparison with the traditional sequence labelling techniques such as HMM, CRF, etc. Experimental results have proven that the proposed deep learning systems perform much better than existing classical machine learning techniques. In short, deep learning based POS tagging with improved word representation can be projected as a novel technique for POS tagging in low resource languages. Similarly, incorporating affix level features along

with pre-trained word embeddings is also a promising thought.

A study on extracting deeper level information associated with nouns and verbs in a text document is also performed. Since most of the semantics in a text document is contained in nouns and verbs, it is necessary to have a deep level tagger for the proper analysis of Malayalam text. The advantage of this tagger in contrast to the general POS tagger is the identification of PNG information associated with nouns and exploration of 'TAM' details associated with verbs. This deep level of information can be used in various language processing applications like sentiment analysis, anaphora resolution, text summarization, etc. Since the manual tagging of deeper level information is costly and labour intensive, a computational approach utilizing the deep level tagger can be effectively employed. In addition to deep level tagger, 648 unique suffixes (separable using rules) for the Malayalam language was extracted and it gave a clear indication about the rich morphology of Malayalam language. The performance of the deep level tagging algorithm is evident from the results given in section 6.3. Experimental results have proven this fact.

A study on algorithms to resolve anaphors in natural language text has been carried out. But, most of the research works on anaphora resolution is concentrated on rule-based techniques utilizing the syntax of the language. Identifying the antecedents that are not in the vicinity of the anaphor are rarely discussed. Extracting meaningful antecedents is possible only if one has semantic information about noun phrases preceding the anaphor. The deep level tagging algorithms discussed in chapter 6 helps

to achieve this goal. Hence, a novel pronoun resolution algorithm utilizing the semantics of the noun phrase antecedents was developed using deep level tagging and anaphora resolution algorithms. The performance of the proposed algorithm is evident from the experimental results given in section 7.3. The performance evaluation was done in comparison with the existing systems in the domain. Experimental results have proven that the proposed algorithm performs much better than state of the art.

One of the most significant challenges in Natural language processing is the development of computational tools for identifying the correct antecedents for anaphoric expressions. Since anaphors are very important for semantic graph construction, they must be appropriately addressed and solved correctly. The proposed pronominal anaphora resolution algorithm resolves all the pronominal anaphors present in Malayalam text. The advantage of the proposed system is that, in contrast to existing rule-based systems, the proposed system makes use off deep level tagger which can give a clear indication about the semantics of nouns and verbs in the text document. The morphological features of the language are effectively utilized for the computational analysis of Malayalam text. Despite using limited linguistic features, the proposed algorithm provides better results which can be utilized for higher level NLP tasks such as question answering, machine translation, text summarization, etc. The power of word embeddings is also exploited in the study, which helped in recognizing the POS and entity categories of words. Since the manual resolution of anaphoric expressions from bulk natural language text is costly and labour

intensive, proposed computational strategies can be effectively employed for automatic text processing mechanisms.

8.2 Future Directions

This thesis includes an elaborate study on different features and architectures that can be adapted to improve the sequence labelling tasks in low resource languages. After unveiling efficient word representation and seq2seq algorithms, improvements have been attained on these areas resulting in new benchmark results. The work also includes a detailed study on the resolution of pronominal anaphors present in Malayalam text. The highlight of all the above systems is the application of machine learning techniques from all the possible directions. Since the manual exploration of above-specified areas are labour intensive and time-consuming, computational strategies suggested through the developed systems can reduce this overhead. As the word embedding based strategies proposed in this thesis offer a global perspective to feature enhanced NLP, it can be easily adapted to other Indian languages having the same status.

This work can be further explored to various sequence labelling tasks such as semantic role labelling, phrase chunking, etc. in low resource languages. The thesis also shed some light towards the influence of affix level features in Malayalam POS tagging and NER. Moreover, this study helps in the identification of factors affecting the disambiguation of tags (POS or NER) for individual words. This can be extended to the next level where the word embedding based cluster features act as

additional information towards sequential labelling. Thus, the semantic class of the word can be predicted more accurately. As an extension of this work, the effect of autoencoders in reducing the combined word representation size can also be investigated. It was able to extract semantic information about nouns and verbs in Malayalam through a deep level tagger. The same method could be extended to any agglutinative language with rich inflections. The availability of local language enabled keywords, and smartphones steer the increased use of social media platforms and conversational systems by the Indian language users. This leads to a drastic increase in the Indian language content over the web. Hence incorporating the proposed entity extraction systems on online Malayalam document search can speed up information extractor systems on Malayalam language.

List of Publications

Journal Papers

1. Ajees A P, and Sumam Mary Idicula. 'An Improved Word Representation for Deep Learning Based NER in Indian Languages'. Information 10.6 (2019): 186. MDPI, (ESCI)
2. Ajees A P, and Sumam Mary Idicula. 'A Deep Level tagger for Malayalam, a morphologically rich language', Journal of Intelligent Systems, De Gruyter, (ESCI) (accepted)
3. Ajees A P, and Sumam Mary Idicula. 'Deep Learning Based Deep Level Tagger for Malayalam ', Recent Patents on Computer Science, Bentham Science, 2019, 12, 82-100 , DOI: 10.2174/221327591257. ISSN: 2213-2759. (SCOPUS)
4. Ajees A P, and Sumam Mary Idicula. 'A Relation Extraction System for Indian Languages', Advances in science, technology and engineering systems journal , ASTES, 2018, 4 ,65-69, ISSN: 2415-6698.(SCOPUS)
5. Ajees A P, and Sumam Mary Idicula. 'A POS tagger for Malayalam using Conditional Random Fields', International Journal of Applied Engineering Research, RIP, 2017, 13, ISSN: 0973-4562. (SCOPUS)
6. Ajees A P, and Sumam Mary Idicula. 'A Named Entity Recognition System for Malayalam using Neural

Networks'. Procedia computer science , Elsevier, 143 (2018): 962-969., ISSN: 1877-0509.(SCOPUS)

7. Ajees A P, and Sumam Mary Idicula. 'An Integrated Framework for Pronominal Anaphora Resolution in Malayalam'. Advances in science, technology and engineering systems journal , ASTES, ISSN: 2415-6698. (SCOPUS) (accepted)

8. Ajees A P, and Sumam Mary Idicula. 'A Novel Word Representation for Deep Learning Based POS Tagging in Malayalam'. Transactions on Asian low resource language processing (TALLIP), ACM, (SCIE) (under review).

9. Ajees A P, and Sumam Mary Idicula. 'A Deep Learning Based POS Tagger for Malayalam, a Morphologically Rich Language'. Malaysian Journal of computer science, PKP,(SCIE) (under review).

Conference Papers

1. Ajees A P, and Sumam Mary Idicula. 'A Named Entity Recognition System for Malayalam using Conditional Random Fields'. ICDSE-2018, IEEE, 2018. 33-39.

2. Ajees A P, and Sumam Mary Idicula. 'A Native Language Identification System using Convolutional Neural Networks'. FIRE-2018, Springer, 2018. 1-8.

3. Ajees A P, and Sumam Mary Idicula. 'A Named Entity Recognition System for Indian Languages'. FIRE-2018, Springer, 2018. 45-53.

Bibliography

[1] Henry Sweet. *The history of language*. Dent, 1900.

[2] Wikipedia contributors. Rotokas language — Wikipedia, the free encyclopedia, 2019. [Online; accessed 24-July-2019].

[3] Memidex. `http://www.memidex.com/anaphora+epanaphora/`. Accessed: 2016-08-28.

[4] Wikipedia. Languages with official status in india. 2012 (accessed December 7, 2018).

[5] ENCYCLOPDIA BRITANNICA. Malayalam language. 2015 (accessed February 18, 2018).

[6] Wikiquote. Malayalam — wikiquote,, 2019. [Online; accessed 19-June-2019].

[7] Wikipedia contributors. Malayalam — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/w/index.php?title=Malayalam&oldid=902164977`, 2019. [Online; accessed 19-June-2019].

[8] Wikipedia contributors. Analytic language — Wikipedia, the free encyclopedia, 2019. [Online; accessed 19-June-2019].

[9] Wikipedia contributors. Brahmi script — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/w/index.php?title=Brahmi_script&oldid=901772441`, 2019. [Online; accessed 19-June-2019].

[10] Ethnologue. How many languages are there in the world?, 2019. [Online; accessed 19-April-2019].

[11] Eric Brill. A simple rule-based part of speech tagger. In *Proceedings of the third conference on Applied natural language processing*, pages 152–155. Association for Computational Linguistics, 1992.

[12] Atro Voutilainen. Morphological disambiguation. *Karlsson et al*, pages 165–284, 1995.

[13] Mohammad Hjouj Btoush, Abdulsalam Alarabeyyat, and Isa Olab. Rule based approach for arabic part of speech tagging and name entity recognition. *Int. J. Adv. Comput. Sci. Appl.(IJACSA)*, 7(6):331–335, 2016.

[14] Nisheeth Joshi, Hemant Darbari, and Iti Mathur. Hmm based pos tagger for hindi. In *Proceeding of 2013 International Conference on Artificial Intelligence, Soft Computing (AISC-2013)*, 2013.

[15] Asif Ekbal, S Mondal, and Sivaji Bandyopadhyay. Pos tagging using hmm and rule-based chunking. *The Proceedings of SPSAL*, 8(1):25–28, 2007.

[16] GM Ravi Sastry, Sourish Chaudhuri, and P Nagender Reddy. An hmm based part-of-speech tagger and statistical chunker for 3 indian languages. *Shallow Parsing for South Asian Languages*, 13, 2007.

[17] Muhammad Fahim Hasan, Naushad UzZaman, and Mumit Khan. Comparison of unigram, bigram, hmm and brill's pos tagging approaches for some south asian languages. 2007.

[18] Eugene Charniak. *Statistical language learning*. MIT press, 1996.

[19] Fatma Al Shamsi and Ahmed Guessoum. A hidden markov model-based pos tagger for arabic. In *Proceeding of the 8th International Conference on the Statistical Analysis of Textual Data, France*, pages 31–42, 2006.

[20] Thorsten Brants. Tnt: a statistical part-of-speech tagger. In *Proceedings of the sixth conference on Applied natural language processing*, pages 224–231. Association for Computational Linguistics, 2000.

[21] Lourdes Araujo. Part-of-speech tagging with evolutionary algorithms. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 230–239. Springer, 2002.

[22] Sang-Zoo Lee, Jun-ichi Tsujii, and Hae-Chang Rim. Part-of-speech tagging based on hidden markov model assuming joint independence. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, pages 263–269. Association for Computational Linguistics, 2000.

[23] Walter Daelemans, Jakub Zavrel, Peter Berck, and Steven Gillis. Mbt: A memory-based part of speech tagger-generator. *arXiv preprint cmp-lg/9607012*, 1996.

[24] Ruhi Sarikaya, Mohamed Afify, Yonggang Deng, Hakan Erdogan, and Yuqing Gao. Joint morphological-lexical language modeling for processing morphologically rich languages with application to dialectal arabic. *IEEE transactions on audio, speech, and language processing*, 16(7):1330–1339, 2008.

[25] Adwait Ratnaparkhi. A maximum entropy model for part-of-speech tagging. In *Conference on Empirical Methods in Natural Language Processing*, 1996.

[26] L Màrquez and J Giménez. A general pos tagger generator based on support vector machines. *Journal of Machine Learning Research*, 2004.

[27] Eugenie Giesbrecht and Stefan Evert. Is part-of-speech tagging a solved task? an evaluation of pos taggers for the german web as corpus. In *Proceedings of the fifth Web as Corpus workshop*, pages 27–35, 2009.

[28] 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.

[29] Amni Anirudh Agarwal Himashu. Part of speech tagging and chunking with conditional random fields. In *proceedings of NLPAI Contest*, 2006.

[30] Manish Shrivastava and Pushpak Bhattacharyya. Hindi pos tagger using naive stemming: Harnessing morphological information without extensive linguistic knowledge. In *International Conference on NLP (ICON08), Pune, India*, 2008.

[31] Asif Ekbal, Rejwanul Haque, and Sivaji Bandyopadhyay. Bengali part of speech tagging using conditional random field. In *Proceedings of Seventh Inter-*

national Symposium on Natural Language Processing (SNLP2007), pages 131–136, 2007.

[32] Asif Ekbal and Sivaji Bandyopadhyay. Part of speech tagging in bengali using support vector machine. In *2008 International Conference on Information Technology*, pages 106–111. IEEE, 2008.

[33] Hammad Ali. An unsupervised parts-of-speech tagger for the bangla language. *Department of Computer Science, University of British Columbia*, 20:1–8, 2010.

[34] Kamal Sarkar and Vivekananda Gayen. A trigram hmm-based pos tagger for indian languages. In *Proceedings of the International Conference on Frontiers of Intelligent Computing: Theory and Applications (FICTA)*, pages 205–212. Springer, 2013.

[35] Avinesh PVS and G Karthik. Part-of-speech tagging and chunking using conditional random fields and transformation based learning. *Shallow Parsing for South Asian Languages*, 21, 2007.

[36] Srinivasu Badugu. Morphology based pos tagging on telugu. *Proceedings of International Journal of Computer Science Issues*, 2014.

[37] PJ Antony and KP Soman. Parts of speech tagging for indian languages: a literature survey. *International Journal of Computer Applications*, 34(8):0975–8887, 2011.

[38] Vasu Renganathan. Development of part-of-speech tagger for tamil. In *Tamil Internet 2001 conference*, 2001.

[39] M Ganesan. Morph and POS Tagger for Tamil : A Rule based Approach.

[40] Shodhganga. Literature survey. `https://shodhganga.inflibnet.ac.in/bitstream/10603/9233/11/11_chapter%202.pdf`, 2016. [Online; accessed 16-June-2017].

[41] M Selvam, AM Natarajan, and R Thangarajan. Structural parsing of natural language text in tamil language using dependency model. *International Journal of Computer Processing of Languages*, 22(02n03):237–256, 2009.

[42] Rajendran S Soman K P Dhanalakshmi V, Anand kumar M. Pos tagger and chunker for tamil language. In *semanticscholar.org*, 2010.

[43] Mary Idicula Sumam, S Soumya, and K Manju. Development of a pos tagger for malayalam-an experience. 2009.

[44] RR Rajeev and Elizabeth Sherly. A suffix stripping based morph analyser for malayalam language. In *Proceedings of 20th Kerala Science Congress*, pages 482–484, 2007.

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

[46] P. C. Reghu Raj Robert Jesuraj K. Mblp approach applied to pos tagging in malayalam language. In

National Conference on Indian Language Computing (NCILC13), Kochi, India, 2013.

[47] C Sunitha et al. A hybrid parts of speech tagger for malayalam language. In *2015 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pages 1502–1507. IEEE, 2015.

[48] Madhu Sareesh Muhammad Noorul Mubarak D and S A Shanavas. A new a pproach to parts of speech tagging in malayalam. *International Journal of Computer Science & Information Technology*, 7(5):121–130, 2015.

[49] Sachin Kumar, M Anand Kumar, and KP Soman. Experimental analysis of malayalam pos tagger using epic framework in scala. 2016.

[50] MS Bindu and Sumam Mary Idicula. Named entity identifier for malayalam using linguistic principles employing statistical methods. *International Journal of Computer Science Issues(IJCSI)*, 8(5), 2011.

[51] Daniel M Bikel, Richard Schwartz, and Ralph M Weischedel. An algorithm that learns what's in a name. *Machine learning*, 34(1-3):211–231, 1999.

[52] GuoDong Zhou and Jian Su. Named entity recognition using an hmm-based chunk tagger. In *proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 473–480. Association for Computational Linguistics, 2002.

[53] Robert Malouf. Markov models for language-independent named entity recognition. In *COLING-*

02: The 6th Conference on Natural Language Learning 2002 (CoNLL-2002), 2002.

[54] Andrew Borthwick and Ralph Grishman. *A maximum entropy approach to named entity recognition*. PhD thesis, Citeseer, 1999.

[55] James Curran and Stephen Clark. Maximumentropy tagging. `https://www.cl.cam.ac.uk/teaching/1011/L107/clark-lecture6.pdf`. Accessed: 2018-04-04.

[56] Paul McNamee and James Mayfield. Entity extraction without language-specific resources. In *proceedings of the 6th conference on Natural language learning-Volume 20*, pages 1–4. Association for Computational Linguistics, 2002.

[57] Andrew McCallum and Wei Li. Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 188–191. Association for Computational Linguistics, 2003.

[58] Xavier Carreras, Lluís Màrquez, and Lluís Padró. Named entity extraction using adaboost. In *COLING-02: The 6th Conference on Natural Language Learning 2002 (CoNLL-2002)*, 2002.

[59] Wenhui Liao and Sriharsha Veeramachaneni. A simple semi-supervised algorithm for named entity recognition. In *Proceedings of the NAACL HLT 2009 Workshop on Semi-Supervised Learning for Natural*

Language Processing, pages 58–65. Association for Computational Linguistics, 2009.

[60] Shubhanshu Mishra and Jana Diesner. Semi-supervised named entity recognition in noisy-text. In *Proceedings of the 2nd Workshop on Noisy User-generated Text (WNUT)*, pages 203–212, 2016.

[61] Pavel P Kuksa and Yanjun Qi. Semi-supervised bio-named entity recognition with word-codebook learning. In *Proceedings of the 2010 SIAM International Conference on Data Mining*, pages 25–36. SIAM, 2010.

[62] Oren Etzioni, Michael Cafarella, Doug Downey, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S Weld, and Alexander Yates. Methods for domain-independent information extraction from the web: An experimental comparison. In *AAAI*, pages 391–398, 2004.

[63] Robert Munro and Christopher D Manning. Accurate unsupervised joint named-entity extraction from unaligned parallel text. In *Proceedings of the 4th Named Entity Workshop*, pages 21–29. Association for Computational Linguistics, 2012.

[64] Rahul Sharnagat. Named entity recognition: A literature survey. *Center For Indian Language Technology*, 2014.

[65] Jisha P Jayan, RR Rajeev, and Elizabeth Sherly. A hybrid statistical approach for named entity recognition for malayalam language. In *Proceedings of the*

11th Workshop on Asian Language Resources, pages 58–63, 2013.

[66] M Anand Kumar Abinaya N, Neethu John and Soman KP. Amrita_cen@ fire 2014: Named entity recognition for indian languages.

[67] Mr Jiljo and Mr Pranav PV. A study on named entity recognition for malayalam language using tnt tagger & maximum entropy markov model. *International Journal of Applied Engineering Research*, 11(8):5425–5429, 2016.

[68] G Remmiya Devi, PV Veena, M Anand Kumar, and KP Soman. Entity extraction for malayalam social media text using structured skip-gram based embedding features from unlabeled data. *Procedia Computer Science*, 93:547–553, 2016.

[69] Tyne Liang and Dian-Song Wu. Automatic pronominal anaphora resolution in english texts. *International Journal of Computational Linguistics & Chinese Language Processing, Volume 9, Number 1, February 2004: Special Issue on Selected Papers from ROCLING XV*, 9(1):21–40, 2004.

[70] Jerry R Hobbs. Resolving pronoun references. *Lingua*, 44(4):311–338, 1978.

[71] David Maclean Carter. *A shallow processing approach to anaphor resolution*. PhD thesis, University of Cambridge, 1986.

[72] Jaime G Carbonell and Ralf D Brown. Anaphora resolution: a multi-strategy approach. In *Proceedings*

of the 12th conference on Computational linguistics-Volume 1, pages 96–101. Association for Computational Linguistics, 1988.

[73] Elaine Rich and Susann LuperFoy. An architecture for anaphora resolution. In *Proceedings of the second conference on Applied natural language processing*, pages 18–24. Association for Computational Linguistics, 1988.

[74] Barbara J Grosz. The representation and use of focus in dialogue understanding. Technical report, SRI INTERNATIONAL MENLO PARK CA MENLO PARK United States, 1977.

[75] Aravind K Joshi and Steve Kuhn. Centered logic: The role of entity centered sentence representation in natural language inferencing. In *IJCAI*, pages 435–439, 1979.

[76] Michael Strube and Udo Hahn. Functional centering: Grounding referential coherence in information structure. *Computational linguistics*, 25(3):309–344, 1999.

[77] Shalom Lappin and Herbert J Leass. An algorithm for pronominal anaphora resolution. *Computational linguistics*, 20(4):535–561, 1994.

[78] Christopher Kennedy and Branimir Boguraev. Anaphora for everyone: pronominal anaphora resoluation without a parser. In *Proceedings of the 16th conference on Computational linguistics-Volume 1*, pages 113–118. Association for Computational Linguistics, 1996.

[79] L Sobha and BN Patnaik. Vasisth: An anaphora resolution system for indian languages. In *Proceedings of International Conference on Artificial and Computational Intelligence for Decision, Control and Automation in Engineering and Industrial Applications*, 2000.

[80] Ruslan Mitkov. Two engines are better than one: Generating more power and confidence in the search for the antecedent. *AMSTERDAM STUDIES IN THE THEORY AND HISTORY OF LINGUISTIC SCIENCE SERIES 4*, pages 225–234, 1997.

[81] Ruslan Mitkov. Factors in anaphora resolution: they are not the only things that matter: a case study based on two different approaches. In *Proceedings of a Workshop on Operational Factors in Practical, Robust Anaphora Resolution for Unrestricted Texts*, pages 14–21. Association for Computational Linguistics, 1997.

[82] Ido Dagan and Alon Itai. Automatic processing of large corpora for the resolution of anaphora references. In *Proceedings of the 13th conference on Computational linguistics-Volume 3*, pages 330–332. Association for Computational Linguistics, 1990.

[83] Sobha Lalitha Devi, Vijay Sundar Ram, and Pattabhi RK Rao. A generic anaphora resolution engine for indian languages. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 1824–1833, 2014.

[84] Bhargav Uppalapu and Dipti Misra Sharma. Pronoun resolution for hindi. In *Proceedings of 7th Discourse Anaphora and Anaphor Resolution Colloquium (DAARC 09)*, pages 123–134, 2009.

[85] Kamlesh Dutta, Nupur Prakash, and Saroj Kaushik. Resolving pronominal anaphora in hindi using hobbs algorithm. *Web Journal of Formal Computation and Cognitive Linguistics*, 1(10):5607–11, 2008.

[86] Kavi Narayana Murthy, L Sobha, and B Muthukumari. Pronominal resolution in tamil using machine learning. In *Proceedings of the First International Workshop on Anaphora Resolution (WAR-I)*, pages 39–50, 2007.

[87] A Akilandeswari and Sobha Lalitha Devi. Resolution for pronouns in tamil using crf. In *Proceedings of the Workshop on Machine Translation and Parsing in Indian Languages*, pages 103–112, 2012.

[88] R Vijay Sundar Ram and Sobha Lalitha Devi. Pronominal resolution in tamil using tree crfs. In *2013 International Conference on Asian Language Processing*, pages 197–200. IEEE, 2013.

[89] Apurbalal Senapati and Utpal Garain. Guitar-based pronominal anaphora resolution in bengali. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 126–130, 2013.

[90] Utpal Sikdar, Asif Ekbal, Sriparna Saha, Olga Uryupina, and Massimo Poesio. Adapting a state-of-the-art anaphora resolution system for resource-poor

language. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pages 815–821, 2013.

[91] S Athira, TS Lekshmi, RR Rajeev, Elizabeth Sherly, and PC Reghuraj. Pronominal anaphora resolution using salience score for malayalam. In *2014 First International Conference on Computational Systems and Communications (ICCSC)*, pages 47–51. IEEE, 2014.

[92] Ruqaiya Hassan and M Halliday. Cohesion in english. *P20*, 1976.

[93] Graerne Hirst. Discourse-oriented anaphora resolution in natural language understanding: A review. *Computational Linguistics*, 7(2), 1981.

[94] Aditya Misra. Metrics to evaluate your machine learning algorithm. https://towardsdatascience.com/metrics-to-evaluate-your-machine-learning-algorithm-f10ba6e38234, 2019. [Online; accessed 25-June-2019].

[95] Will Koehrsen. Beyond accuracy: Precision and recall. https://towardsdatascience.com/beyond-accuracy-precision-and-recall-3da06bea9f6c, 2019. [Online; accessed 25-June-2019].

[96] Wikipedia contributors. Fleiss' kappa — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=

Fleiss%27_kappa&oldid=895198898, 2019. [Online; accessed 25-June-2019].

[97] A python binding for crfsuite. https://github.com/scrapinghub/python-crfsuite. Accessed: 2017-09-30.

[98] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[99] François Chollet et al. Keras. 2015.

[100] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

[101] Cicero D Santos and Bianca Zadrozny. Learning character-level representations for part-of-speech tagging. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1818–1826, 2014.

[102] Xiang Yu, Agnieszka Faleńska, and Ngoc Thang Vu. A general-purpose tagger with convolutional neural networks. *arXiv preprint arXiv:1706.01723*, 2017.

[103] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[104] Malayalam POS Tagger Online. http://www.iiitmk.ac.in/MalayalamPOSTagger/index1.jsp. Accessed: 2016-09-30.

[105] Ewan Klein Steven Bird and Edward Loper. *Natural Language Processing with Python*. O'Reilly Media, 2009.

[106] AP Ajees and Sumam Mary Idicula. A pos tagger for malayalam using conditional random fields. 2018.

[107] Ayushi-Jain. Hunpos-tagger. https://github.com/ayushi-jain97/HunPOS-Tagger, 2017.

[108] HB Barathi Ganesh, KP Soman, U Reshma, Kale Mandar, Mankame Prachi, Kulkarni Gouri, and Kale Anitha. Information extraction for conversational systems in indian languages - arnekt iecsil. In *Forum for Information Retrieval Evaluation*, 2018.

[109] Forum for information retrieval evaluation. http://fire.irsi.res.in/fire/2019/home. Accessed: 2019-02-02.

[110] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[111] Vikas Yadav, Rebecca Sharp, and Steven Bethard. Deep affix features improve neural named entity recognizers. In *Proceedings of the Seventh Joint Con-*

ference on Lexical and Computational Semantics, pages 167–172, 2018.

[112] Edouard Grave, Piotr Bojanowski, Prakhar Gupta, Armand Joulin, and Tomas Mikolov. Learning word vectors for 157 languages. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*, 2018.

[113] Seung-Hoon Na, Hyun Kim, Jinwoo Min, and Kangil Kim. Improving lstm crfs using character-based compositions for korean named entity recognition. *Computer Speech & Language*, 54:106–121, 2019.

[114] 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.

[115] Chuanhai Dong, Jiajun Zhang, Chengqing Zong, Masanori Hattori, and Hui Di. Character-based lstm-crf with radical-level features for chinese named entity recognition. In *Natural Language Understanding and Intelligent Applications*, pages 239–250. Springer, 2016.

[116] Jisha P Jayan, RR Rajeev, S Rajendran, et al. Morphological analyser and morphological generator for malayalam-tamil machine translation. *International Journal of Computer Applications*, 13(8):0975–8887, 2011.

[117] 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.

[118] SK Saranya. Morphological analyzer for malayalam verbs. *Unpublished M. Tech Thesis, Amrita School of Engineering, Coimbatore*, 2008.

[119] PM Vinod, V Jayan, and VK Bhadran. Implementation of malayalam morphological analyzer based on hybrid approach. In *Proceedings of the 24th Conference on Computational Linguistics and Speech Processing (ROCLING 2012)*, pages 307–317, 2012.

[120] R Sunil, Nimtha Manohar, V Jayan, and KG Sulochana. Morphological analysis and synthesis of verbs in malayalam. *ICTAM-2012*, 2012.

[121] RR Rajeev and Elizabeth Sherly. Morph analyser for malayalam language: A suffix stripping approach. Proceedings of 20th Kerala Science Congress, 2007.

[122] Jurafsky and Martin. Speech and language processsing. 2002.

[123] Abrar K J. Malayalam verb morphological analysis: A computational linguistics approach. *Thunchath Ezhuthachan Malayalam University*, 2019.

[124] Edward Loper and Steven Bird. Nltk: The natural language toolkit. In *Proceedings of the ACL-02 Workshop on Effective tools and methodologies for teaching natural language processing and computa-*

tional linguistics-Volume 1, pages 63–70. Association for Computational Linguistics, 2002.

[125] Pedregosa Fabian, Varoquaux Gaël, Gramfort Alexandre, Michel Vincent, Thirion Bertrand, Grisel Olivier, Blondel Mathieu, Prettenhofer Peter, Dubourg Vincent, Vanderplas Jake, et al. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[126] Peyman Passban, Qun Liu, and Andy Way. Boosting neural pos tagger for farsi using morphological information. *ACM Transactions on Asian and Low-Resource Language Information Processing (TALLIP)*, 16(1):4, 2016.

[127] Kath. Word of the day anaphora. 2018 (accessed December 7, 2018).

[128] M Sadanandam and D Chandra Mohan. Telugu pronominal anaphora resolution.

[129] Dan Jurafsky and James H Martin. *Speech and language processing*, volume 3. Pearson London, 2014.

[130] MS Bindu and Sumam Mary Idicula. A hybrid model for phrase chunking employing artificial immunity system and rule based methods. *International Journal of Artificial Intelligence & Applications*, 2(4):95, 2011.

[131] Rajina Kabeer and Sumam Mary Idicula. Text summarization for malayalam documentsan experience. In *2014 International Conference on Data Science & Engineering (ICDSE)*, pages 145–150. IEEE, 2014.

[132] Bed time stories - wonderful stories in malayalam for kids. https://kuttykadhakal.blogspot.com/. Accessed: 2018-09-30.

[133] Madhyamam. https://www.madhyamam.com/. Accessed: 2018-08-28.