IMPROVING POS TAGGING FOR TAMIL USING DEEP LEARNING

A. Alstan

Index Number: 13000063

Supervisor: Dr. A. R. Weerasinghe

December 2017

Submitted in partial fulfillment of the requirements of the
B.Sc. in Computer Science Final Year Project (SCS4124)

Declaration

I certify that this dissertation does not incorporate, without acknowledgement, any material previously submitted for degree or diploma in any university and to the best of my knowledge and belief, it does not contain any material previously published or written by another person or myself except where due reference is made in the text. I also hereby give consent for my dissertation, if accepted, be made available for photocopying and for interlibrary loans, and for the title and abstract to be made available to outside organizations.

Candidate Name: Ann Anobiya Alstan

……..

Signature of Candidate Date:

This is to certify that this dissertation is based on the work of Ms. Ann Anobiya Alstan under my supervision. The thesis has been prepared according to the format stipulated and is of acceptable standard.

Supervisor Name: Dr. A. R. Weerasinghe

……

Signature of Supervisor Date:

Abstract

Part of Speech (POS) tagging is one of the basic and important application of Natural Language Processing (NLP). The accuracy of POS tagging have influence on the performance of many other NLP applications. This research presents a novel deep learning based POS tagger for Tamil language. Tamil is an agglutinative, morphologically rich and free word order language. The recent research works for Tamil language POS tagging were not be able to give state of the art POS tagging accuracy like other languages. Therefore, this research is done to improve the POS tagging for Tamil language using deep learning approaches.

In the first phase of the research, few classification based models such as Decision Tree classifier, Naïve Bayes classifier and Support Vector Machine (SVM) classifier have been used to build POS tagger for Tamil language. Few handcrafted features were used to train these models. There are difficulties in useful feature extraction because of the complex structure of Tamil language. To avoid the use of handcrafted features and to improve the performance of the POS tagging of Tamil language a novel model was built using Long Short Term Memory (LSTM) neural network in this research.

The models were evaluated with the AUKBC Tamil POS corpus which contains 50,876 sentences. Based on the experiments on the corpus, Support Vector Machine model was selected as the baseline model for this research. The accuracy of 95.697%, precision 96%, recall of 96% and f1-measure of 96% were obtained for the SVM classifier based POS tagger. An experiment on the AUKBC Tamil POS corpus with the LSTM model was carried out by changing the number of training epochs and the efficiency of the proposed POS tagger was evaluated on the corpus using the evaluation metrics precision, recall, f1-measure and accuracy. The accuracy of 96.74%, precision of 97%, recall of 97% and f1-measure of 97% were obtained for the LSTM model with five training epochs.

Keywords – Part of Speech Tagging, Tamil Language, Deep learning

Preface

Part of Speech tagging is one of the basic and popular research area of Natural Language Processing. Part of Speech for Tamil language has reasonable amount of research works recently. There are various approaches have been used for POS tagging. The literature of Tamil POS tagging shows no works based on deep learning approaches. This research work mainly focus on improving POS tagging for Tamil language using deep learning.

The dataset used for this research is obtained from the Computational Linguistic Research Group, AUKBC research Centre, MIT Campus of Anna University. Whole analysis on this AUKBC Tamil POS corpus and the tag set is solely done by me to understand the structure of the corpus.

Different approaches have been used to develop the POS tagger with this corpus. To evaluate the performance of the deep learning for POS tagging of Tamil language, baseline models with the approaches used previously was built during this research. The implementation works and the idea behind the models is my own work. The supervisor has given the guidance for each work.

Acknowledgement

First of all I would like to express my heartfelt gratitude to my supervisor Dr.A.R.Weerasinghe for the support and generous guidance that motivated me to make this research a success. I am grateful to him for finding out time to meet me every week and respond to my e-mails as quickly as possible. His guidance helped me in all the time of this research and writing of this thesis. It was a pleasure to work with him.

I would like to thank Mr. Viraj Welgama and Dr. T. Sritharan for the advice and guidance given through reviewing my work as examiners.

I would also like to thank our research project coordinator Dr. H E M H B Ekanayake, for his guidance given throughout the year. Also I want to show my gratitude for the university staff and all the lecturers for the support given to successfully complete this research.

I would like to convey my thanks to my family especially my sister for the guidance and support throughout this research.

Finally yet importantly, I would like to thank my friends for supporting me throughout the research work and giving me courage.

Table of Content

List of Figures

List of Tables

List of Acronyms

API	–	Application program Interface
BIS	–	Bureau of Indian Standard
LSTM	–	Long Short Term Memory
NLP	–	Natural Language Processing
NLTK	–	Natural Language Tool Kit
RNN	–	Recurrent Neural Network
POS	–	Part of Speech
SGD	–	Stochastic Gradient Descent
SVC	–	Support Vector Classification
SVM	–	Support Vector Machine
UDHR	–	Universal Human Rights Declaration

Chapter 1

Introduction

Part of Speech (POS) tagging is the process of assigning one of the part of speech tags (Grammatical category) for each word in a given sentence or text based on the context of the word, which is one of the disambiguation techniques at lexical level of Natural Language Processing (NLP). POS tagging is one of the important aspect in Natural Language Processing tasks such as speech recognition, natural language parsing, morphological parsing, information retrieval and machine translation. Even POS tagging seems simple task when compare to other NLP tasks, it is very important to achieve good performance on POS taggers. Because most of the NLP applications use POS taggers in preprocessing step and accuracy of such applications mainly depends on the performance of the POS taggers.

Assigning POS tags to each word in a given text manually is a laborious and time-consuming task. Also the manual process require linguistics with huge linguistic knowledge of the language. This lead to the development of many approaches to automate the POS tagging process. Most of the automatic POS taggers takes a sentence as input, assigns a POS tag to each word in the sentence, and gives the annotated text as output.

Different approaches have been tried for POS tagging in European languages like English and stated good accuracy. There are many state of the art POS taggers with different approaches for English. However, morphologically rich and complex languages

like Tamil lack such standard state of the art POS taggers. In addition to the complex structure of the language, lack of large lexical resources also have being the barrier for building standard POS taggers.

This research is an attempt of contributing to the NLP related researches of Tamil language by achieving good accuracy on POS tagging with deep learning approaches. So that a POS tagger is developed using Long Short Term Memory (LSTM) network in this research.

1.1. Background and Motivation to the Research

Tamil language is a member of Dravidian language family, primarily spoken by Tamils in India, Sri Lanka and Singapore and has a significant number of speakers in Malaysia, Mauritius and emigrant communities around the world. It is the official language of Indian state Tamil Nadu, also one of the official languages in Sri Lanka and Singapore. With more than 77 million speakers, Tamil is one of the widely spoken language in the world. In 2004, Tamil was declared as classical language, which means that, Tamil met the criteria that, its origins are ancient, it has an independent tradition and it possess a considerable body of ancient culture [1].

Tamil is an agglutinative and morphologically rich language. Tamil words consist of lexical root and affixes attached to it. Generally, most of the affixes are suffixes. There can be any number of suffixes attached to a root word. There are no limitation to the suffixes that can be attached to a root word in Tamil. There may many English words need to translate a single Tamil word. For example, the word 'pōkamuṭiyātavarkaḷukkāka' (போகமுடியாதவர்களுக்காக) consist of seven morpheme components attached to the root word 'pōka' [2].

Tamil: pōkamuṭiyātavarkaḷukkāka – போகமுடியாதவர்களுக்காக
English: for the sake of those who cannot go

pōka	muṭi	y	āta	var	kaḷ	ukku	āka
go	**accomplish**	**word joining letter**	**negation (Impersonal)**	**nominalizer He/she who does**	**Plural marker**	**to**	**for**

Tamil suffixes can be divided into derivational suffixes, which change the meaning or the POS category of the word and inflectional suffixes, which mark categories such as person, mood, tense, number, etc.

Tamil is a free word order language. Typically, Tamil follows the Subject – Object – Verb order. However, this can be flexible as the main verb of the sentence must be at the end of the sentence. All the other categories can be anywhere in the sentence. For example, consider the sentence 'I gave him a pen'. This can be translated to Tamil in different ways [3].

1. நான் அவனுக்கு ஒரு பேனா கொடுத்தேன்
   naan avanukku oru peenaa kotuththeen (I him a pen gave)

2. அவனுக்கு நான் ஒரு பேனா கொடுத்தேன்
   avanukku naan oru peenaa kotuththeen (him I a pen gave)

3. ஒரு பேனா நான் அவனுக்கு கொடுத்தேன்
   oru peenaa naan avanukku kotuththeen (a pen I him gave)

4. நான் ஒரு பேனா அவனுக்கு கொடுத்தேன்
   naan oru peenaa avanukku kotuththeen (I a pen him gave)

Here all of these four translations give the correct meaning in Tamil. But the direct mapping of the English words to the Tamil words in the sentence does not make any meaningful sentence in English. This nature of Tamil language make the POS tagging quite hard when compare to other languages, which follow strict word order like English.

All of the natural languages are ambiguous. When an utterance has more than one semantic representation then it is referred as ambiguous utterance. Tamil language can have lexical ambiguity and structural ambiguity.

Lexical ambiguity refers to the type of ambiguity, which occurs when a word can be assigned to more than one grammatical or syntactic category based on the context. For example, the word 'kaal' could be noun or cardinal in the following sentence [4].

- அவன் கால் பகுதியை சாப்பிட்டான்

avan kaal pakutiyaic caappiTTaan

English Translation:

1. He ate quarter of something (Cardinal)
2. He ate leg part of something (Noun)

POS tagging can resolve this lexical ambiguity.

Structural ambiguity refers to the type of ambiguity, which occurs when constituents in larger structures have more than one interpretation based on their internal structure and syntactic position. For example [4]:

- *வெள்ளை மருந்து குப்பி*
  veLLai maruntu kuppi

English Translation:

1. Medicine bottle which is in white color
2. A bottle with white color medicine.

This kind of grammatical structure of Tamil makes Part of Speech tagging for Tamil language quite hard. Part of Speech tagging works as one of the preprocessing step for many NLP applications. Therefore, there must be some POS taggers with good accuracy. Then only other application, which depend on POS tagger's performance, can work well. So implementing a good POS tagger is a crucial task.

1.2. Research Problem and Research Question

There have been few research works done for Tamil language POS tagging over the past few years using different traditional approaches of POS tagging. But there are no stable state of the art POS tagger for Tamil in the literature. Lack of standard POS tagset and lack of standard large annotated corpus for Tamil language are also limitation for developing state of the art POS taggers like other languages. Also the complex morphology, agglutinative nature and the free word order structure of Tamil grammar makes POS tagging for Tamil harder. Current state of the art POS taggers of other

languages use machine learning approaches especially deep learning methods. So based on that this research is focused on the following questions.

- How deep learning approaches can help to improve the accuracy in Part of Speech tagging for Tamil language?

- How deep learning techniques are compared to existing tagging methods for Tamil?

1.3. Significance of the Research

There are various works have been done for Tamil POS tagging using rule based method, statistical method and both combination of rule based and statistical methods. In early stage of POS tagging rule based POS taggers have been developed. But this approach is not work well with unknown words, exhaustive set of hand coded rules should be used to overcome this problem. Since Tamil is morphologically rich and agglutinative language, developing rule based tagger require a large amount of rules. We need to spend a lot of effort and time, also need huge knowledge of complex grammatical structures of Tamil language to define the rules, which is practically difficult.

The stochastic models can be developed for Tamil POS tagging and it works better than rule based POS taggers. However, this model also will not work well with the unknown text; most of them are tagged as noun by some taggers. This can be solved by using the morphological information of the unknown word when calculating the probability. Even though some tag sequences can be given from the tagger for the given sentences that are not correct according to the grammar rules of Tamil language.

As mentioned earlier there are no POS tagging works based on the neural networks approach for Tamil language. But, using deep learning approaches in POS tagging result in high accuracy than the rule based methods and stochastic methods for other languages. So applying such deep learning approach to the morphologically rich Tamil language may also lead to high accuracy in POS tagging. Since accuracy of many NLP applications depends on the accuracy of the POS taggers, it is good to have a POS

tagger with good accuracy. So exploring the deep learning approaches for Tamil POS tagging and propose a novel approach is good. Therefore, this research work is important.

1.4. Goal and Objectives

- **Goal**

  The main goal of this research project is to improve the Part of Speech tagging for Tamil language using deep learning approaches. So other NLP applications which use POS tagging could be able to get benefit from it and perform well.

- **Objectives**
  - Obtaining an annotated large corpus for POS tagging.
  - Obtaining informative features for Tamil language POS tagging.
  - Designing a baseline model for Tamil POS tagging based on the literature.
  - Designing a novel deep learning based model for Tamil language POS tagging.

1.5. Scope and Limitations

This research is done for POS tagging of Tamil language. The spoken language is rapidly changing one and has slight changes in different countries as well as different regions of the country. So by considering that, this research purely done for written Tamil. The written Tamil also could be different in some situations like novel writing, blog writing because the writer may use different styles to enhance the creativity and maintain a unique style. Also as mentioned earlier the written language have differentiations based on the domain and the period that is used. This research project is limited to the AUKBC Tamil Part of Speech corpus (AUKBC-TamilPOSCorpus2016v1) [5]. Since our POS tagger is built based on the AUKBC Tamil POS corpus, which is collected from a historical novel the accuracy of the tagger may vary based on the domain of the corpus.

The AUKBC Tamil POS corpus is manually annotated using the Bureau of Indian Standard (BIS) tag set, which is the standard tag set for Indian languages. So the scope of this research is limited to the tags that is defined in the BIS tag set.

1.6. Research Methodology

During the past years, there have been many research works on POS tagging for all the languages. Most of them use the rule based, stochastic and machine learning approaches for POS tagging. In Tamil language, also these above mentioned methods have been used. In recent research works, they have explored the deep learning approaches for POS tagging on various languages that gave better results than these methods. Therefore experimenting deep learning approaches for Tamil POS tagging also crucial.

In this regards a manually annotated and verified corpus with large size is used for this research work. This research follows quantitative approach as the research methodology. In the first phase of the research the literature review on different approaches used for POS tagging in Tamil language and some other languages is done. Here the methodology, dataset, tag set and the evaluation criteria of the research work is considered.

In the second phase of the research some of the existing POS tagging tools with different approaches is found out or few POS taggers with existing standard approaches is constructed. In addition, the AUKBC Tamil POS corpus is evaluated with those models to get the baseline model for this research work. The results obtained from those various methods are compared and one final baseline model is selected from that.

Then as the final phase, deep learning approaches used for other languages is experimented for the Tamil language using the AUKBC Tamil POS corpus. According to the results obtained from these experiments, a novel model for Tamil language POS tagging is defined. Then the new model is experimented and refined with the above mentioned corpus until a better result than the previous methods is obtained.

1.7. Outline of the Dissertation

This chapter gives the introduction to the research. This contain brief introduction of the background of the research, research problem and research question, significance of this research, goal and objectives of the research and the scope and limitations. Chapter 2 provides the literature review. Chapter 3 explains the design of the research. It contains the architecture of the baseline model and the deep learning model. All the implementation details of the models are mentioned in the chapter 4. Chapter 5 provide the evaluation and results of the models built in the research and finally chapter 6 have the conclusion of this research work.

Chapter 2

Literature Review

2.1. Different POS tagging Approaches

Automatic POS tagging can be done using different approaches. These approaches can be rule-based approach, corpus based approach and hybrid approach. Rule based approach require a set of hand crafted rules according to the language grammar. The corpus based approach use the details from the dataset to build the POS taggers. It can be dived into supervised and unsupervised taggers based on the nature of the corpus being used. The corpus-based taggers can be either statistical taggers or machine learning taggers. Hybrid taggers combine any two from the above to utilize the advantages of both taggers.

2.1.1. Supervised Tagging and Unsupervised Tagging

POS tagging work can be mainly categorized into two groups such as supervised tagging and unsupervised tagging based on the degree of automation of the task. Automation of the POS tagging can be done using the annotated corpora or unannotated corpora. In supervised POS taggers, an annotated corpus is used for the training process. Therefore, the accuracy of the supervised POS tagger is depend on the accuracy of the corpus annotation. Therefore, the mistakes in the corpus affect the POS tagging in supervised learning.

In unsupervised taggers, unannotated corpus is used for the training. Therefore, no need of pre annotated corpus. The model itself identify the different cluster of tags based on the features and train based on that. Normally annotating large corpus is laborious and require huge linguistic knowledge. Therefore, in unsupervised taggers the time and effort required for the corpus annotation is not necessary. Even though unsupervised learning is more practical the efficiency and accuracy of the tagger is less than the supervised taggers.

2.1.2. Rule Based Method

Rule based tagging is the very old approach in POS tagging, it uses a set of hand written rules based on morphological and contextual information. Most rule-based taggers contain two stage architecture. The first stage is the dictionary lookup procedure, which returns a set of possible tags for each word to be tagged. The second stage use a set of hand crafted rules to identify the correct tag when a word has more than one possible tag. This rules help to overcome the ambiguity problem.

Since rule based taggers use a set of hand crafted rules, huge linguistic knowledge is needed to define the rules. Therefore, this manual rule definition process consume huge amount of time and human effort. It makes the construction of rule-based taggers quite hard.

2.1.3. Stochastic Method

Stochastic tagging follows data driven approaches in which frequency based information is automatically derived from the tagged corpus and used to tag new words. Any model which incorporates frequency or probability may be labelled stochastic model. The stochastic taggers disambiguate words based on the probability that the word occurs with a particular tag. Some of the stochastic approaches are Hidden Markov Model (HMM), Maximum Entropy Model and Conditional Random Fields (CRF).

These statistical taggers give more accuracy than rule based taggers. Stochastic taggers use probability information from the training dataset. Therefore, it works well only on the domain of the training dataset. For example if the tagger is trained on the corpus which have political news and used to tag the corpus with the educational news

then the tagger will not give good result on it. Also a huge annotated dataset is required for the training.

2.2. Related Works

2.2.1. Rule Based Method

- Rule based approach is used to develop a morphological tagger for Tamil language by V. Ranganathan [6] in this research. He have developed a tagger called 'tagtamil' using Prolog. This POS tagger was built with a knowledge base consist of morphological rules of Tamil. He have implemented the system based on the principals of the theory of the lexical phonology and morphology. A small dictionary also built as a part of this tagger, which contains information about the root forms of Tamil words and the grammatical information describing the nature of words. In this research work, he did not used any standard approach to evaluate the tagger.

2.2.2. Statistical Method

- S. L. Pandian and T. V Geetha [7] have designed a statistical language model based on Bayes' theorem that considers the lexical category of the stem and morphological components of the word for POS tagging in this research. They have defined their own tag set, which have 35 POS tags. They have used a tagged corpus of 470,910 words, which is tagged with 35 POS tags in a semi-automatic way using an existing morphological analyzer. They have designed two different models considering single morphological component and multiple morphological component. They have created an algorithm to estimate the contribution factors of the model. In the evaluation process, they have used Precision, Recall, F-measure and Perplexity to evaluate the proposed model. System is evaluated with two set of corpus, one test corpus having 43,678 words, in which 36,128 words are morphologically analyzed within their tag set, 6,123 words are named entity, while the remaining words are un-identified. And other corpus have 62,765 words. Overall accuracy of 95.92% and 94.45% have been achieved respectively.

- Some of the statistical models have been used to experiment on the monolingual, bilingual and multilingual corpora for POS tagging by M. Ramanathan et.al [8]. They have selected Tamil, Hindi, English and French languages for the POS tagging purpose. Universal Human Rights Declaration corpus (UDHR) has been used in this research; this corpus is translated in 300 languages and consists of 75 lines of short text. UDHR corpus have been tagged using existing tools, Hindi was tagged using a tagger developed by the Society for Natural Language Technology Research and English and French was tagged using Tree Tagger tool and Tamil corpus was tagged manually with the tag set of 12 tags. As the preprocessing they have done the sentence alignment using Microsoft Researches Bilingual Sentence Aligner Tool and word alignment using GIZA++ tool.

  Supervised learning have been done in Tamil corpus with HMM, CRF and SVM models. Fourteen different features have been considered in this monolingual learning. Tamil corpus has been aligned with each language separately for the bilingual learning. Bilingual supervised learning have been done using CRF++ tool and semi supervised learning have been done using projection and aggressive tag probability re-estimation technique. Eleven different features have been considered in this bilingual learning. For multilingual learning, tags from all other languages have been projected into Tamil. CRF++ tool was used for this multilingual learning. They have used four different features for this. They have considered a baseline accuracy 33.47%, which is obtained for the most frequent tag in the Tamil Language. In monolingual learning, the SVM model gave the maximum accuracy of 61.29%. For other experiments, results were not mentioned in accurate values.

2.2.3. HMM models

- P. Arulmozhi and L. Sobha [9] developed a POS tagger using Hidden Markov Model and Viterbi transition in this research. They have defined their own tag set, which have 17 basic tags and 31 sub tags. They got totally 350 unique tags by considering the inflections. The POS tagger have two modules, one is for building the language model and other is for find out the Viterbi tag sequence for

a given sentence. In this proposed model, they have assumed each POS tag as a state and words as observable symbols. The system was implemented in Perl language. The training corpus consists of 25,000 tagged words. The test data set had 3000 words and testing have done in three phase with 1000 words each. They have used precision and recall in the evaluation process. The advantage of Viterbi algorithm in this system is ignoring small errors in the corpus. There were many sentences not tagged because of the unknown words.

2.2.4. SVM models

- V. Dhanalakshmi et.al [10] developed a POS tagger using SVM methodology based on Linear Programming in this research. They have considered centered window of five token for feature extraction during this research. They have designed a non-linear SVM using linear programming. SVMs are trained for every POS tag and binary SVMs are extended to multi class SVMs using the one versus rest method. They have created their own tag set for this research, which contains 32 tags. A tagged corpus with 225,000 words was used for feature extraction, which is collected from Dinamani newspaper, yahoo Tamil news, online Tamil short stories, etc. Training of the model was done with 15,000 sentences and testing was done with 10,000 sentences from the corpus. Overall accuracy of 95.63% have been achieved in this research.

- V. Dhanalakshmi et.al [11] have automated the POS tagging and chunking process for Tamil using machine learning techniques in this research. They have defined their own tag set called “Amrita” by following guidelines mentioned in AnnCorra, Annotating Corpora Guidelines for POS and Chunk Annotation for Indian Languages. The POS tag set contains 32 tags and Chunking tag set contains 9 tags. Same corpus of their previous research [10] was used for training and testing process. POS corpus was trained and tested with the machine learning based tool called SVMTool by tuning the parameters and feature patterns for Tamil language. SVMTool is a software package, which contains three components SVMTlearn – model learner, SVMTagger – tagger and SVMTeval – the evaluator. Another tool YamCha is used for Chunking purpose. 95.64% and 95.82% was achieved for POS tagger and chunker respectively.

2.2.5. CRF Models

- S. L. Pandian and T. V Geetha [12] have developed a POS tagger and Chunker using CRF model in this research. They have used a tag set with five main tag categories and 131 sub categories. The CRF models are designed using the CRF++ tool. The baseline CRF model have created considering basic word features for POS tagging and Chunking. Then it was modified by including the morphological information of the word for both processes. Baseline and modified CRF models for POS tagging have trained with a corpus containing 39,000 sentences, which have average 13 words per sentence and this corpus is semi-automatically tagged and manually verified. The CRF models for Chunking have trained with the corpus tagged using the modified POS tagging model.

  Both POS tagging models have been tested with three different data sets with the size of 18,345, 19,834 and 18,907 words respectively. Chunking models have been tested with data sets with the size of 6342, 6834, 6521 chunks respectively. They have used precision, recall and F-measure for the evaluation. For the baseline POS tagging model, precision 0.8622, 0.8706 and 0.8690 Recall 0.8304, 0.8304 and 0.8421 F-measures 0.8460, 0.8674 and 0.8553 and for the modified POS tagging CRF model, precision 0.8711, 0.8794 and 0.8886, Recall 0.8993, 0.9167 and 0.8971, F-measures 0.8850, 0.8977 and 0.8928 respectively obtained for the above mentioned three data sets. For the baseline Chunking CRF model, precision 0.7833, 0.7908 and 0.7943, Recall 0.7996, 0.7609 and 0.7821 F-measures 0.7913, 0.7755 and 0.7882 and for the modified Chunking model, precision 0.7917, 0.8398 and 0.8343, Recall 0.8849, 0.8402 and 0.8698, F-measures 0.8357, 0.8400 and 0.8517 respectively obtained for the mentioned chunking test data set.

2.2.6. Hybrid Methods

- This research has been done to build rule based morphological analyzer and POS tagger by M. Selvam et.al [13]. Also they have applied Projection and Induction techniques to make improvements on the above. They have defined their own tag set, more than 600 POS tags were obtained based on morphological and

categorical information of Tamil words. They have used Bible aligned corpora (English and Tamil) and CIIL corpus (Tamil) for rule based morphological analyzer and POS tagger, and achieved 85.56% and 83% respectively. Improvement in the accuracy was achieved by using text alignment, POS projection and induction techniques from English to Tamil. First using the alignment dictionary and GIZA++ tool English words are aligned to Tamil words. After this, they have projected the POS tag and categorical information of the English word to the mapped Tamil word. Then they have identified the stem of the English word with the MBLEM lemmatization tool and induced the Tamil root word by a lookup with English stem in the parallel dictionary. Hash maps have been used in lookup dictionaries in order to reduce searching time. Only the Bible corpus was used in this experiment since the sentence aligned corpora is only available in Bible corpora. 92.48% accuracy has been achieved for this projection and induction method with the improvement of 7%.

- P. Arulmozhi et.al [14] have developed a hybrid POS tagger by combining the statistical HMM tagger and rule based tagger in this research. The statistical POS tagger is implemented using HMM and Viterbi transition like other HMM based POS taggers. This HMM tagger is implemented in Perl language. They have used a tag set with 18 basic tags and 30 sub tags. The tagger was trained using a small corpus, which is manually tagged with the above tag set. This HMM tagger is tested with dataset having 5000 words. The system is evaluated with three data sets containing 1565, 1810 and 1616 words and precision and recall is calculated. Precision 81.9%, 82.28% and 83.2%, Recall 63.6%, 64.1% and 65.7% obtained respectively for the above datasets.

The rule based POS tagger finds the POS of the root word using the inflection of the word without using any root word dictionary. Here the sentence is tagged at lexical level first with the help of suffix list and lexical rules and then the ambiguity is resolved using the context sensitive rules in the next level. The tag set of this tagger is limited with 12 major tags only. This system performance is evaluated with CIIL corrected corpus. Two sets of 1000 words are taken to the evaluation and achieved the precision 92% and 91.8% respectively.

In the hybrid POS tagger first the untagged sentence is given to the HMM tagger and tagged, then the output from the HMM tagger is passed to the rule based tagger, so the untagged sentences and wrongly tagged words are tagged. The system is trained with a corpus having 25,000 words and evaluated with 5000 words data set. 97.12%, 97.07% and 97.27% Precision and Recall is obtained for the 1565, 1810 and 1616 words data set from this hybrid POS tagger.

2.2.7. Deep learning methods

- C. D. Santos et.al [15] have designed a Deep Neural Network (CharWNN) that learn character level representation of words and combine it with normal word level representation to perform Part-of-Speech (POS) Tagging. Also the researchers have produced two POS taggers for English and Portuguese languages using this CharWNN. They have avoided the use of handcrafted features, by adding a convolutional network layer that helps effective feature extraction from any size of words. The proposed neural network extend the work done by Collobert et.al [16] in 2011 by adding a convolutional layer, which learn character level embedding of words. They have used the Collobert's Window approach to score each word in the sentence using the embedding that combines word level and character level embedding, and to identify the unique tags for each word Viterbi algorithm has been used. They have done the unsupervised pre-training of word embedding, which help to improve the accuracy in POS tagging. For both English and Portuguese languages, the pre-training was done using the word2vec tool with same set of parameters. They have experimented the model on Penn Tree-Bank Wall Street Journal corpus and achieved 97.32% accuracy for English language. Also for Portuguese language on Mac-Morpho corpus and achieved 97.47% accuracy.

- B. Plank et.al [17] have evaluated the effectiveness of different representation in Bidirectional Long Short Term Memory (bi-LSTM) model, compared these models with 22 languages under different conditions and proposed a novel bi-LSTM model with auxiliary loss for POS tagging in this research. Their basic bi-LSTM tagging model is context bi-LSTM that takes word embedding as inputs.

Also they used sub token level (character or Unicode byte) embedding of words at the lower level and then concatenated with word embedding to be given as input for context bi-LSTM. In their proposed model, they train the bi-LSTM tagger to predict both the tags of the sequence and a label that represents the log frequency.

All the bi-LSTM models are implemented in CNN/pycnn a flexible Neural Network library. They have used the same hyper parameters for all the models in the experiment process. They have compared their bi-LSTM with TNT [18] a second order HMM with suffix tire and a freely available CRF tagger [19].They have done the experiment on 22 languages. For the multilingual experiments, they have used the data from the Universal Dependencies project v1.2.

The bi-LSTM without sub token information outperforms other two taggers only on three languages. The bi-LSTM model using character embedding only work well than TNT on nine languages. The combined word and character embedding model outperforms all the models on 21 languages. Also by evaluating the proposed and baseline models with different size data set, they came to the decision that TNT is better with little data and bi-LSTM is better with more data and outperforms CRF model. They have analyzed the robustness of the models with the label noise. They found out that bi-LSTM was affected more than the TNT when the noise increased.

- P. Wang et.al [20] have proposed a bidirectional LSTM Recurrent Neural Network with word embedding for POS tagging and achieved a state of the art tagging accuracy. Also they have proposed a novel approach for training word embedding. The LSTM RNN model was built using CURRENT, a machine learning library. To train the word embedding they have used the North American news as the unlabeled data, which contains about 536 million words. They have evaluate the LSTM RNN model with different hidden layer size to choose the best. The proposed model was evaluated with the Wall Street Journal data from Penn Treebank III. The accuracy of 97.40% is achieved for the proposed BLSTM RNN model.

Chapter 3

Design

As mentioned in the research methodology in section 1.6 the first implementation step of this research is the construction of the POS taggers with some approaches that is already used for Tamil language POS tagging. Since POS tagging is a kind of classification problem some supervised classifiers like Decision Tree classifier, Naïve Bayes classifier and Support Vector Machine classifier is used for POS tagging. The common architecture of the classification based POS tagging model is given in the figure 3.1.

As mentioned in the research methodology the next implementation part of this research is the construction of POS tagger based on the deep learning approaches. The LSTM neural network is used to build the POS tagger in this research. The high level architecture for the deep learning based POS tagging model is given in the figure 3.2.

Initially the corpus is given to the preprocessing module. The corpus is preprocessed and stored in an appropriate format, so other modules can read it. Then the preprocessed dataset is given to the sentence tokenization module. Tagged sentences from this module is split into two sets as training dataset and testing dataset. 75% of the corpus is allocated for the training process and the rest 25% of the dataset is used for the evaluation process. The detailed description of each module is given below in this chapter.

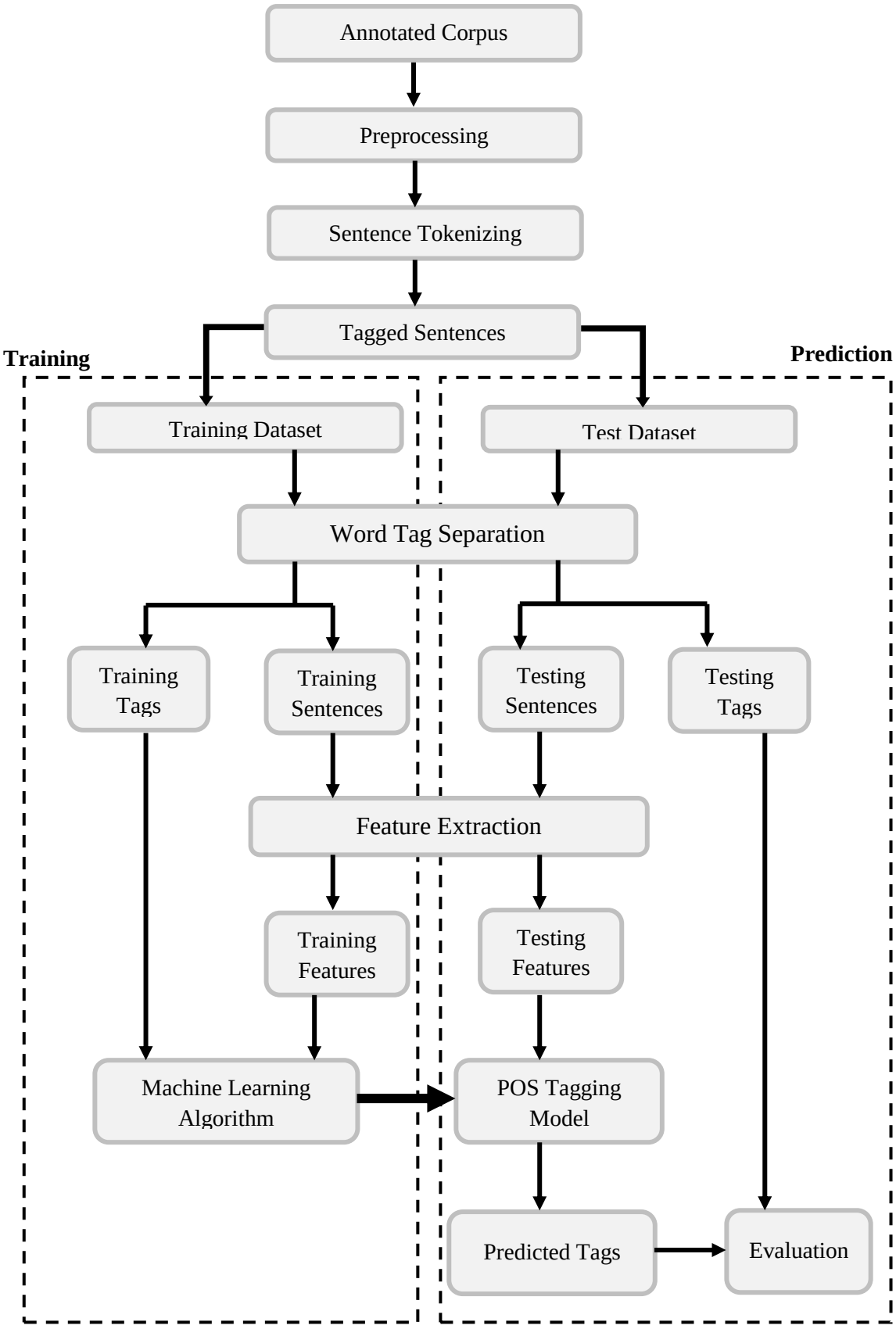

Figure 3.1: Architecture of the classifier based POS tagger

3.1. Annotated Corpus

The corpus that is used in this research is AUKBC Tamil POS Corpus [5], which is developed by the Computational Linguistic Research Group, AUKBC research Centre, MIT Campus of Anna University. This is the largest corpus in Indian languages annotated using the BIS POS tagset. The source of the corpus is a very famous contemporary novel in Tamil "Ponniyin Selvan" written by "Kalki Krishnamoorthy". The corpus was annotated by four trained linguists manually. The original text of the corpus was edited by splitting some words and concatenating some words based on the structure of Tamil words. There are short poems between the proses, which is not annotated with POS tags. The poem phrases are tagged as "poem" as mentioned above.

A complete analysis on the AUKBC Tamil POS Corpus is carried out to study the frequencies of tags occurred in the corpus. The details of the dataset is given below in the table 3.1.

Table 3.1: Details of the dataset

Details	Count
Number of sentences in the corpus	50,876
Number of tokens in the corpus	515,283
Number of unique tokens in the corpus	63,549
Number of words in the corpus (without punctuations and symbols)	414,483
Number of unique words in the corpus	63,509
Number of unique token – tag pair	66,600

Therefore, from this analysis we can come to the conclusion that some words have been tagged with different tags. That is because of the ambiguity of the language. Some of the words may get different tags based on its context.

There are some incorrectly tagged words and symbols. For example, the exclamation mark (!) is tagged as residual symbol (RD_SYM) in some places and residual foreign (RD_RDF) in some places, which is actually a residual symbol.

Table 3.1 shows how the tags in BIS tagset are distributed in the AUKBC Tamil POS corpus. As we can see N_NN (Common Noun) is the most frequently used tag in the corpus.

Table 3.2: Frequency of tags distribution in the dataset

Tag	Tag Count of Total Words	Tag Count of Unique Words
N_NN	121124	26226
N_NNP	31045	4658
N_NST	1332	49
PR_PRP	46284	1022
PR_PRF	2135	84
PR_PRL	220	19
PR_PRC	64	7
PR_PRQ	258	24
DM_DMR	6211	26
DM_DMQ	14159	1934
V_VM_VF	49573	10272
V_VM_VNF_COND	3269	1042
V_VM_VNF_INF	9953	2311
V_VM_VNF_RP	18176	2684
V_VM_VNF_RP_PSP	178	764
V_VM_VNF_VBN	36271	4720
V_VM_VNG	44	36
N_NNV	39	24
JJ	10086	1578
RB	22845	4917
PSP	10001	251
CC	5	2
CC_CCD	2613	13
CC_CCS	10648	129
RP_INJ	1773	52
RP_INTF	622	12
RP_NEG	10967	2187
RP_NEG_PSP	86	40
QT_QTF	1304	17
QT_QTC	5645	206
QT_QTO	413	116
RD_PUNC	77969	14
RD_ECH	1053	329
RD_SYM	24958	12
RD_RDF	8	6
poem	1038	815

3.1.1 Tagset

To develop a POS tagger and POS tagged corpus, it is necessary to define a tagset. Different languages have different tagset. Also different corpus may have different tagset depend on the purpose. When we define a tagset for the POS tagging task we need to decide whether to consider only the lexical aspect or to mark plurality and gender distinctly.

The tagset used in the AUKBC Tamil POS corpus is the Bureau of Indian Standard (BIS) tagset. This is a national standard tagset for Indian languages. This tagset consider only the lexical categories of the language. BIS tagset have hierarchy of two levels. The first level has the categories of highest level POS classes. There are 11 tags in this top level.

- Noun
- Verb
- Pronoun
- Demonstrative
- Adjective
- Preposition
- Conjunction
- Particles
- Quantifier
- Residuals

The next level has some necessary subtypes of the POS tags of the top level. Overall, there are 37 tags in this BIS tagset. The tag called ‘poem’ is used to tag the small poems in the corpus in addition to the tags defined in the BIS tagset.

3.2. Preprocessing

Preprocessing is done based on the standards followed by the corpus. The data in the corpus is in tab separated format. There are six columns in the corpus that represent the volume number of the book, chapter number of the volume, sentence number in the chapter, word number in the sentence, word and the POS tag respectively.

An ordered dictionary is used to store the word - tag pair as the value and combination of volume number, chapter number and the sentence number as the key of the dictionary. This combination is used as the key to uniquely identify the sentences in the corpus. The output format of this preprocessing module is given in the table 3.3.

Input Data:

1	1	1	1	முதலாவது	QT_QTO
1	1	1	2	அத்தியாயம்	N_NN
1	1	2	1	ஆடித்திருநாள்	N_NN
1	1	3	1	ஆதி	N_NN
1	1	3	2	அந்தமில்லாத	RP_NEG
1	1	3	3	கால	N_NN
1	1	3	4	வெள்ளத்தில்	N_NN

Table 3.3: The output format of the preprocessed data

Sentence	Key of the Dictionary	Value of the Dictionary
1	1-1-1	முதலாவது QT_QTO
	1-1-1	அத்தியாயம் N_NN
2	1-1-2	ஆடித்திருநாள் N_NN
3	1-1-3	ஆதி N_NN
	1-1-3	அந்தமில்லாத RP_NEG
	1-1-3	கால N_NN
	1-1-3	வெள்ளத்தில் N_NN

3.3. Sentence Tokenizing

After the preprocessing the corpus is given to the sentence tokenization module. All the tagged sentences from the corpus is extracted in this module. The ordered dictionary is used to get the sentence list. The key of the dictionary is used to separate the sentences. A list of tagged sentence list is given as the output of this sentence tokenization module.

Then the tagged sentences are split into training and testing dataset. The training dataset contains 38,705 sentences and testing dataset contains 12,902 sentences.

3.4. Word Tag Separation

The tagged sentence contain the word and the target tag of the word. So it must be separated before giving it to the feature extraction module since the features are extracted from the words only. So this module separate the words from the tags and store it in a matrix preserving the sentence boundaries. The tags also stored in a matrix. This process is done for both training and testing sentences.

3.5. Feature Extraction

In this step, some useful features from the words in the given sentences are extracted. The features used in this module are

- The target word
- Previous word of the target word
- Next word of the target word
- Last one suffix of the target word
- Last two suffixes of the target word
- Last three suffixes of the target word

3.6. Machine Learning Algorithm

Learning process of the model is done in this step. Once the features are extracted from the training dataset, those features are given to the classification based machine learning algorithm to be trained. The Decision Tree classification algorithm, Naïve Bayes algorithm and Support Vector Machine algorithm are used in this research. The feature vector and the target POS tag is given as input to the model for training. Once the training is completed the model is ready to do the classification task, which is POS tagging in this case. The trained model with fine-tuned parameters will be used as the POS tagger in the prediction module to determine the POS tags of the given words.

3.7. POS Tagging Model

The actual POS tagging work, which means the task of identifying the correct tag for the given word is done in this module. The testing features are given to this POS tagging model as the input. The predicted POS tag for each word is given as the output from this module. Predicted tags and the target testing tag is given to the evaluation module to evaluate the performance of the model.

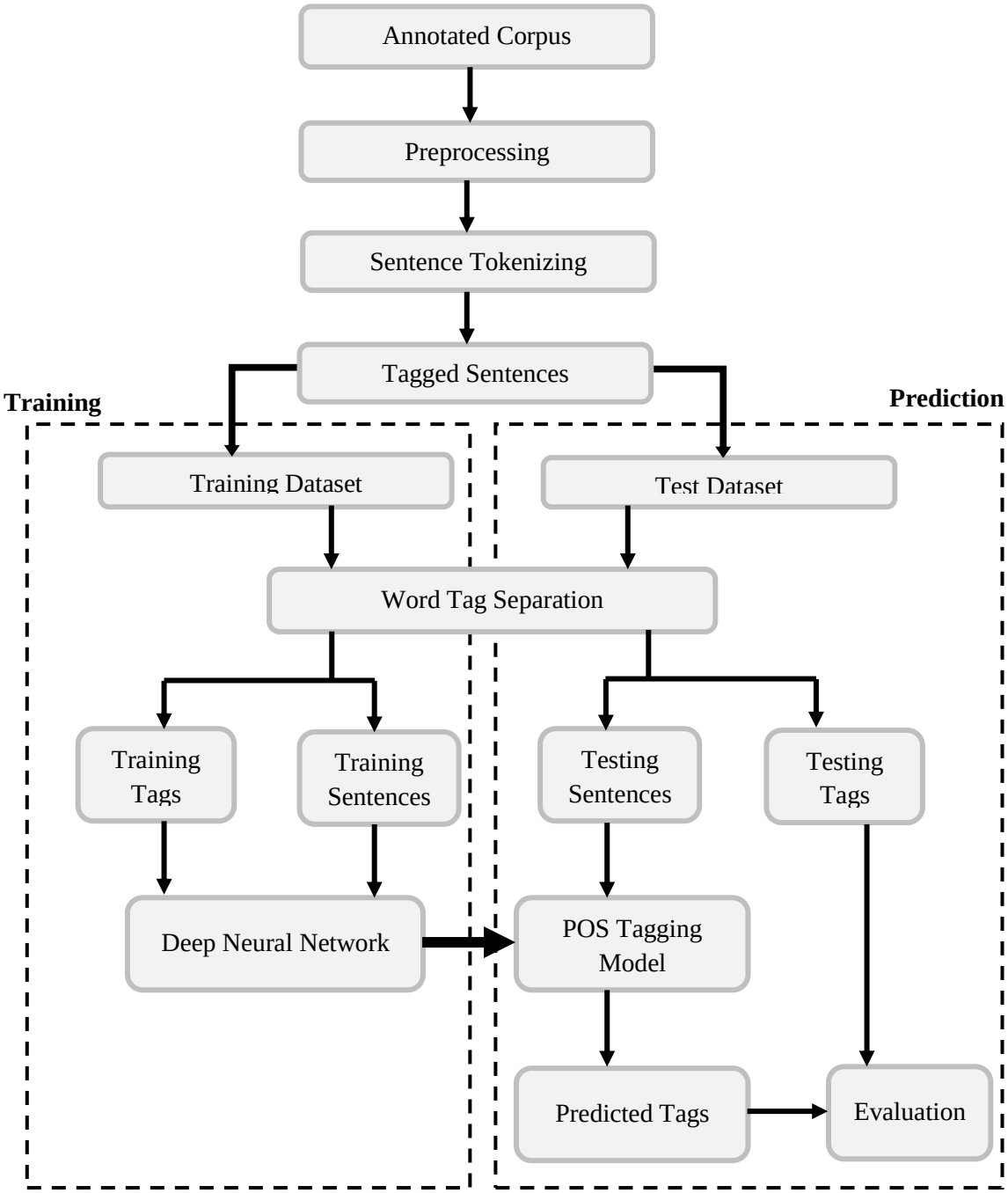

Figure 3.2: Architecture of the Deep Learning based POS tagger

Figure 3.2 gives the high level architecture of the deep learning based POS tagging model. The preprocessing module, sentence tokenizing module, word tag separation module and the POS tagging module of this model is same as the classification based POS tagging model mentioned above. But feature extraction module and the machine learning algorithm module is replaced with deep neural network module.

3.8. Deep Neural Network

In deep neural networks features are extracted automatically by the network itself with hidden layers. So we don't need to do the feature extraction explicitly. Here the deep neural network Long Short Term Memory (LSTM) model is used for this POS tagging purpose. The trained parameters of this network will be used by the POS tagger in the prediction module with testing dataset. The proposed LSTM network model is given in the figure 3.3.

3.8.1. LSTM neural network model

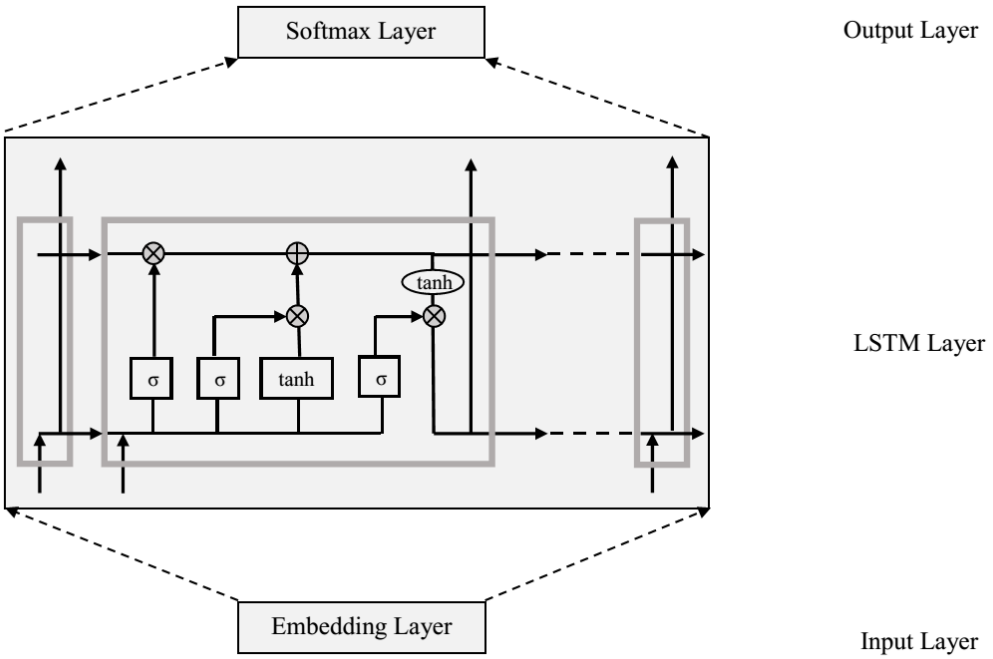

Figure 3.3: LSTM Network for POS tagging

Given a sentence with the target tags to predict the tag probability distribution of each word. The input word sequence is converted into an integer sequence before given to the network model. The integer sequence, which represent the sentence is fed into the network model.

3.8.1.1. Embedding Layer

The first layer of the model, which is the embedding layer make the one hot vector representation of the input sequence with the size of the input dimension. And then reduce the dimension of this one hot vector representation of the input sequence to the size of the embedding. The input dimension of this embedding layer is the size of the word vocabulary.

3.8.1.2. LSTM Layer

Long Short Term Memory (LSTM) model is a variation of Recurrent Neural Networks (RNN). RNN solves the issue in traditional neural networks by keeping track of previous events. RNNs have loops in the network which allows the information to persist. The context information of a language model can be learned by RNNs, but up to some extent. When the context information needed for the task grows the RNNs became unable to learn to connect the information. This long term dependencies can be learned by the LSTM networks.

LSTMs are explicitly designed to avoid the long term dependency problem. The default behavior of this networks is to remember information for long periods of time. All the LSTMs have the form of chain of repeating modules of neural network. The repeating module of the LSTM network have four layers interacting in a special way. Those layers are three sigmoid layer and one tanh layer.

The key to LSTM is the cell state. LSTM networks have the ability to add or remove information to the cell state using gates. The forget-gate and an add-gate is used by the network to learns when to forget something and when to update the internal storage.

3.8.1.3. Output Layer

The output layer is a softmax layer. The dimension of this layer is the number of tag types in the corpus. This layer outputs the tag probability distribution of the input word.

All the weights are trained using Stochastic Gradient Descent algorithm to maximize the likelihood on the training data.

3.9. Evaluation Metrics

The performance of the models built for POS tagging is evaluated with the metrics Accuracy, Precision, Recall and F1-score.

Accuracy is the most basic performance measure, which is simply the ratio of correctly identified observations to the total observations.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Precision is the ratio of correctly identified positive observations to the total identified positive observations. Precision shows the ability of the classifier not to label a negative observation as positive.

$$Precision = \frac{TP}{TP + FP}$$

Recall is the ratio of correctly identified positive observations to all the observations in the actual positive class. It is called as sensitivity also. Recall shows the ability of the classifier to identify all the positive samples correctly.

$$Recall = \frac{TP}{TP + FN}$$

Here TP – True Positives

TN – True Negatives

FP – False Positives and

FN – False Negatives

F1-score is the weighted average of the precision and recall. It consider both false positives and false negatives. F1-score is better when the target classes have uneven distribution.

$$F1 - Score = \frac{2 * Precision * Recall}{Precision + Recall}$$

Chapter 4

Implementation

This chapter provides the details of the implementation that have been done in this research. The environmental setup of the implementation and the details of the parameters used for the model is given in this chapter.

As clearly mentioned in the research design there are two main implementation parts in this research. The first one is the implementation of a baseline model with the AUKBC Tamil POS corpus which is based on the classification models. And the second is the implementation of the deep learning based model for Tamil POS tagging.

4.1. Implementation Environment

All the implementations are done using the Python language. Python language is used because it is one of the fastest languages and has many libraries that support the implementation of NLP applications. Python version 3.5.2 is used for all the implementations.

Python language have libraries that help to implement machine learning applications. Scikit-learn [21] library of Python is used to implement the classification based POS tagging models. Scikit - learn is a simple, efficient and robust library mainly

used for the implementation of Machine Learning applications. Also it is an open source library. This library is built upon the SciPy (Scientific Python) package.

Python language have a library called ‘Keras’ which is the python deep learning library. Keras is a high level neural networks API, written in python and capable of running on top of TensorFlow or Theano. To build the LSTM model Keras library is used with ‘Theano’ backend in this research.

Scikit-learn library have built in package for evaluation process called metrics. This metrics package have all the evaluation metrics mentioned in section 3.9 implemented. This ‘sklearn.metrics’ package is used in this research to evaluate all the models.

4.2. Preprocessing

Preprocessing is done as explained in the design chapter. Python ordered dictionary is used to store the preprocessed dataset. The combination of volume number, chapter number and sentence number is used as the key of the dictionary and word-tag pair is used as the value of the dictionary. Then the dictionary of word tag pair is then given to a function to get the tagged sentences from the corpus.

The list of tagged sentences is further given to the feature extraction module. The features mentioned in the chapter 3 is extracted from each of the words in the sentences and represented as a feature vector. The ‘CountVectorizor’ of the scikit-learn feature extraction module is used to represent the extracted features in the vectorized form. Then this feature vector and the target POS tag is given to the machine learning algorithm to be trained.

4.3. Baseline Model

Three models have been built as baseline model. Decision Tree classifier, Naïve Bayes classifier and Support Vector Machine classifier are used to build these models. The same features have been used for all of these models to maintain the consistency.

4.3.1. Decision Tree Classifier

Decision Tree is a supervised learning model used for classification. Decision Tree classifier asks a series of carefully crafted questions about the features of the test record; Every time it receives an answer, and a follow-up question is asked until a conclusion about the target class label is reached. In Decision Tree classification, the dataset is divided into smaller and smaller subsets while the decision tree is built incrementally. The final result of the training is a tree with decision nodes and leaf nodes. The decision nodes can have two or more child nodes and the leaf nodes represent the target classes of the problem.

Scikit-learn library have a built in function for Decision Tree classifier in the 'sklearn.tree' package. This function is used to build the Decision Tree based POS tagger with the AUKBC Tamil POS corpus. The extracted features from the training words and the target POS tag are given to the Decision Tree classifier model and the model is trained for POS tagging.

4.3.2. Naïve Bayes Classifier

Naïve Bayes is a kind of supervised classification algorithm that uses the Bayes Theorem. This model predicts membership probabilities for each class such as the probability that given data point belongs to a particular class. The target class with the highest probability is considered as the most likely class. Naïve Bayes classification model make an assumption that all the features are unrelated to each other. Presence or absence of a feature does not influence the presence or absence of any other feature.

Multinomial Naïve Bayes classifier is a type of Naïve Bayes classifier that uses the data in multinomial distribution. This model is mostly used for text classification. Scikit-learn library have the built in package for Naïve Bayes classification models. The 'MultinomialNB' function from the 'sklearn.naïve_bayes' package is used to build the Naïve Bayes classifier based POS tagger.

4.3.3. Support Vector Machine Classifier

Support Vector Machine is a supervised machine learning algorithm that is mainly used for classification problems. In this algorithm, each data point is plotted as a point in n-dimensional space with the value of each feature being the value of a particular coordinate. Then the classification is performed by finding the hyper plane that differentiate the two classes very well.

The Scikit-learn library have the built in package for SVM classification. The SVM in Scikit-learn library support both dense and sparse sample vectors as input. Sparse vector is given in this research by considering the memory of the device.

Scikit-learn library supports SVM model for multi class classification using the classes SVC, NuSVC and LinearSVC. The LinearSVC is the implementation of the Support Vector Classification for the case of linear kernel. This LinearSVC implements the 'one-versus-rest' multiclass strategy for training n_class models. This strategy was used in a previous work [11] for Tamil POS tagging. The LinearSVC class is used to build the SVM based POS tagger in this research.

4.4. Deep Learning Model

The initial preprocessing to the dataset for this model also same as the previous baseline model. In deep learning based model feature extraction model is not necessary. So the input sentence itself need to be represented in numeric value. To obtain a numeric value for each unique word in the corpus, the unique word list from the corpus is obtained and sorted in the order of frequency distribution. The index of the word in the sorted list is used to represent the word. A list of integer which represent the sentence is given as the input value for the neural network model. Long Short Term Memory neural network is used in this research for POS tagging as the deep learning model.

4.4.1. LSTM neural network

To build this LSTM model as explained in the section 3.8.1, Python 'Keras' library is used with 'Theano' backend. All the deep neural networks can be built with 'model' package in the Keras library. The models are defined as a sequence of layers. The sequential model is created using the 'Sequential' function and each layer is added to the model one at a time as necessary.

The input layer of this model is the word embedding layer. The input dimension of the embedding layer is the vocabulary size, which is 53,907 and the output dimension of the embedding layer is set to 20.

The next layer of the model is the LSTM layer. Output dimension of the LSTM layer is set to 100. The activation function used in this layer is the 'tanh' function. The 'hard sigmoid' activation function is used in the recurrent step of this LSTM layer.

The final layer of this model is a fully connected layer. The output dimension of this layer is the number of POS tags available in the corpus. The 'softmax' activation function is used in this output layer for the multi class classification.

The model is trained with the Stochastic Gradient Descent (SGD) optimizer and the loss of categorical cross entropy. SGD is initialized with learning rate 0.01, decay 1e-6 and momentum 0.9.

Each sentence is given to the model as the input. The variable length input is handled by the 'train_on_batch' function available in the model class. In this LSTM model sentences with only one word is eliminated from the dataset at training and testing phase.

Chapter 5

Results and Evaluation

This chapter gives the detail description of the experiments and the results obtained for each experiments during this research. The experiment is done for each of the POS tagging models explained in the previous chapters. The experiments were done with the AUKBC Tamil POS corpus as mentioned above.

The results of the classification based models were compared to select the best baseline model for the AUKBC corpus. And then the LSTM model is compared with the baseline model to check the performance of the deep learning models for Tamil POS tagging over the traditional machine learning approaches.

The experiment with different size of the dataset is also done in this research to study the behavior of the models with small size dataset.

5.1. Evaluation of the POS taggers

As mentioned in the implementation chapter above baseline models and the deep learning model is built for POS tagging. Three classification based models have been built as baseline models. The models was evaluated with the AUKBC Tamil POS corpus. In total, there are around 51,607 sentences in the dataset. There are 38,705 sentences in the training dataset and 12,902 sentences in the test dataset. Results obtained for each of the models are given below in the following sections.

5.1.1. Result of Decision Tree POS Tagger

The above mentioned features are extracted and given to the Decision Tree classifier to train and the result of the evaluation is given in the figure 5.1 and the table 5.1.

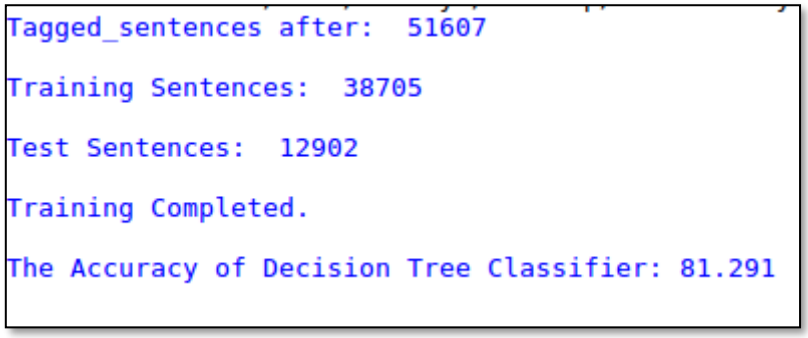

Figure 5.1: Accuracy of the Decision Tree Classifier based POS tagger

The accuracy of 81.29% is obtained for the Decision Tree classifier based POS tagger when the model is trained and tested with the above mentioned dataset.

The table 5.1 gives the classification report for the Decision Tree based POS tagger. This classification report gives the Precision, Recall, F1- Score and the Support value of each of the POS tag and the average value for each score. The support value give the number of occurrences of the tag in the corpus.

The average precision obtained for this Decision Tree classifier based model is 0.82, average recall is 0.81 and the average F1-score is 0.81. The Common noun (N_NN) is the mostly available tag in the dataset appeared 29,984 times, which is classified with precision 0.66, recall 0.86 and F1-score 0.75. When analyzing the result in the classification report it is clear that most of the tags have been identified with F1-score more than 50% but Proper noun (N_NNP), Wh–word pronoun (PR_PRQ), Residual echo words (RD_ECH), Conditional verb (V_VM_VNF_COND) and Gerund (V_VM_VNG) tags are identified with F1-score less than 50%. And Verbal noun (N_NNV), POEM, Reciprocal pronoun (PR_PRC) and Relative pronoun (PR_PRL) was not identified at all, which got 0% of precision, recall and F1-score. The tags Conjunctions (CC), Demonstratives (DM), Quantifiers (QT), Personal pronouns (PR_PRP), Negative particles (RP_NEG), Intensifier particles (RP_INTF), Finite verbs (V_VM_VF), Verbal

participle (V_VM_VNF_VBN), Relative participle (V_VM_VNF_RP) and Residual symbols (RD_SYM) are identified with more than 80% of F1-score. The Residual punctuation (RD_PUNC) tag is identified correctly, which have 100% precision, recall and F1-score. Some of the tags are appeared only few times in this dataset. Some of those have been identified either correctly or not at all. The support of those tags is very low. Therefore, this could be the reason for such loss.

Table 5.1: Classification Report of the Decision Tree Classifier based POS tagger

Tags	Precision	Recall	F1-Score	Support
CC_CCD	0.99	0.95	0.97	636
CC_CCS	0.95	0.88	0.91	2,769
DM_DMQ	0.84	0.86	0.85	3,326
DM_DMR	0.94	0.99	0.96	1,558
JJ	0.72	0.66	0.69	2,513
N_NN	0.66	0.86	0.75	29,984
N_NNP	0.59	0.36	0.45	9,186
N_NNV	0.00	0.00	0.00	9
N_NST	0.59	0.75	0.66	369
POEM	0.00	0.00	0.00	108
PR_PRC	0.00	0.00	0.00	20
PR_PRF	0.70	0.56	0.62	619
PR_PRL	0.00	0.00	0.00	59
PR_PRP	0.92	0.76	0.83	12,792
PR_PRQ	1.00	0.29	0.45	136
PSP	0.91	0.70	0.79	2,528
QT_QTC	0.95	0.82	0.88	1,125
QT_QTF	0.98	0.96	0.97	267
QT_QTO	0.69	0.74	0.71	94
RB	0.76	0.74	0.75	6,062
RD_ECH	0.50	0.07	0.13	174
RD_PUNC	0.99	1.00	1.00	19,846
RD_SYM	1.00	0.99	0.99	5,920
RP_INJ	0.64	0.61	0.62	437
RP_INTF	0.98	0.74	0.84	160
RP_NEG	0.95	0.79	0.86	2,909
RP_NEG_PSP	0.00	0.00	0.00	10
V_VM_VF	0.89	0.87	0.88	12,229
V_VM_VNF_COND	0.75	0.33	0.46	848
V_VM_VNF_INF	0.72	0.59	0.65	2,746
V_VM_VNF_RP	0.83	0.79	0.81	4,806
V_VM_VNF_RP_PSP	0.75	0.48	0.58	498
V_VM_VNF_VBN	0.82	0.86	0.84	9,437
V_VM_VNG	0.24	0.78	0.37	9
Average / Total	0.82	0.81	0.81	134,189

5.1.2. Result of Naïve Bayes POS Tagger

The same features used for Decision Tree classifier based POS tagger is used for the Naïve Bayes classifier also. The accuracy of this model is given in the figure 5.2 and the table 5.2.

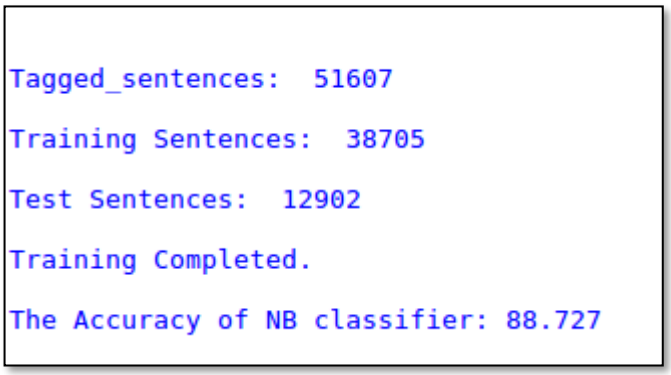

Figure 5.2: Accuracy of the Naïve Bayes Classifier based POS tagger

The accuracy of 88.73% is obtained for the Naïve Bayes classifier based POS tagger with the above mentioned dataset. The table 5.2 gives the classification report for the Naïve Bayes classifier based POS tagger.

The average precision obtained for this model is 0.89, average recall is 0.89 and the average F1-score is 0.88. The Common noun (N_NN) is the mostly available tag in the dataset appeared 29,984 times, which is classified with precision 0.81, recall 0.93 and F1-score 0.86. When analyzing the result in the classification report it is clear that most of the tag has been identified with F1-score more than 50%. But there are few tags which have F1-score less than 50% and some of them are 0%. The support of those tags is very low. So this could be the reason for such loss.

Some of the POS tags get more than 80% of F1-score. Those are Conjunctions (CC), Demonstratives (DM), Quantifiers (QT), Common noun (N_NN), Verbal noun (N_NNV), Personal pronouns (PR_PRP), residual symbol (RD_SYM), Negation (RP_NEG) and Interjection (RP_INJ) particles, Finite verb (V_VM_VF) and Verbal (V_VM_VNF_VBN) and Relative (V_VM_VNF_RP) particle. The Residual punctuation (RD_PUNC) tag is identified correctly with 100% of precision, recall and F1-score.

Table 5.2: Classification Report of the Naïve Bayes Classifier based POS tagger

Tags	Precision	Recall	F1-Score	Support
CC_CCD	1.00	0.62	0.77	636
CC_CCS	0.91	0.80	0.85	2,769
DM_DMQ	0.91	0.83	0.87	3,326
DM_DMR	0.98	0.99	0.98	1,558
JJ	0.83	0.73	0.78	2,513
N_NN	0.81	0.93	0.86	29,984
N_NNP	0.95	0.72	0.82	9,186
N_NNV	0.00	0.00	0.00	9
N_NST	1.00	0.01	0.02	369
POEM	0.00	0.00	0.00	108
PR_PRC	0.00	0.00	0.00	20
PR_PRF	1.00	0.10	0.18	619
PR_PRL	0.00	0.00	0.00	59
PR_PRP	0.88	0.96	0.92	12,792
PR_PRQ	0.00	0.00	0.00	136
PSP	0.82	0.62	0.71	2,528
QT_QTC	0.99	0.82	0.90	1,125
QT_QTF	0.96	0.88	0.91	267
QT_QTO	0.00	0.00	0.00	94
RB	0.79	0.78	0.78	6,062
RD_ECH	1.00	0.03	0.07	174
RD_PUNC	1.00	0.99	1.00	19,846
RD_SYM	0.98	1.00	0.99	5,920
RP_INJ	0.93	0.84	0.88	437
RP_INTF	0.97	0.36	0.53	160
RP_NEG	0.98	0.77	0.87	2,909
RP_NEG_PSP	0.00	0.00	0.00	10
V_VM_VF	0.89	0.98	0.93	12,229
V_VM_VNF_COND	0.97	0.31	0.47	848
V_VM_VNF_INF	0.89	0.77	0.82	2,746
V_VM_VNF_RP	0.89	0.93	0.91	4,806
V_VM_VNF_RP_PSP	0.95	0.12	0.21	498
V_VM_VNF_VBN	0.87	0.94	0.90	9,437
V_VM_VNG	0.00	0.00	0.00	9
Average / Total	0.89	0.89	0.88	134,189

5.1.3. Result of Support Vector Machine POS Tagger

The features used for the above models have been used for the SVM classifier based model also. The result obtained for the SVM is given in the figure 5.3 and the table 5.3.

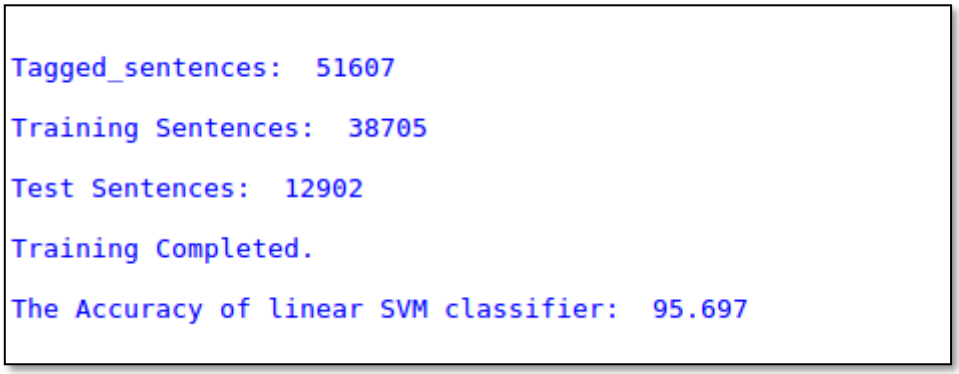

Figure 5.3: Accuracy of the SVM Classifier based POS tagger

The accuracy of 95.70% is obtained for the SVM classifier based POS tagger with the above mentioned dataset. The table 5.3 gives the classification report for the SVM classifier based POS tagger.

The average precision obtained for this model is 0.96, average recall is 0.96 and the average F1-score is 0.96. The Common noun (N_NN) tag is the mostly appeared tag in the dataset, which is classified with precision 0.93, recall 0.96 and F1-score 0.95. When analyzing the result in the classification report it is clear that most of the tags have been identified with F1-score more than 90%. Adjective (JJ), Reciprocal pronoun (PR_PRC), Relative pronoun (PR_PRL) and Conditional verb (V_VM_VNF_COND) tags have been identified with F1-score more than 80%. Verbal noun (N_NNV), POEM, Wh–word pronoun (PR_PRQ), Gerund (V_VM_VNG) have been identified with F1-score less than 50%. Relative demonstrative (DM_DMR) and Residual punctuation (RD_PUNC) was correctly identified with 100% of precision, recall and F1-score.

Table 5.3: Classification Report of the SVM Classifier based POS tagger

Tags	Precision	Recall	F1-Score	Support
CC_CCD	0.99	1.00	0.99	636
CC_CCS	0.98	0.99	0.98	2,769
DM_DMQ	0.95	0.96	0.95	3,326
DM_DMR	1.00	0.99	1.00	1,558
JJ	0.89	0.85	0.87	2,513
N_NN	0.93	0.96	0.95	29,984
N_NNP	0.95	0.90	0.92	9,186
N_NNV	0.50	0.11	0.18	9
N_NST	0.70	0.77	0.73	369
POEM	0.62	0.27	0.37	108
PR_PRC	1.00	0.80	0.89	20
PR_PRF	0.97	0.90	0.93	619
PR_PRL	0.96	0.75	0.84	59
PR_PRP	0.99	0.99	0.99	12,792
PR_PRQ	0.98	0.31	0.47	136
PSP	0.90	0.92	0.91	2,528
QT_QTC	0.96	0.95	0.95	1,125
QT_QTF	0.98	0.98	0.98	267
QT_QTO	0.96	0.95	0.95	94
RB	0.92	0.90	0.91	6,062
RD_ECH	0.60	0.45	0.51	174
RD_PUNC	1.00	1.00	1.00	19,846
RD_SYM	0.99	0.99	0.99	5,920
RP_INJ	0.95	0.99	0.97	437
RP_INTF	0.99	0.89	0.94	160
RP_NEG	0.97	0.92	0.95	2,909
RP_NEG_PSP	0.62	0.50	0.56	10
V_VM_VF	0.97	0.98	0.97	12,229
V_VM_VNF_COND	0.85	0.83	0.74	848
V_VM_VNF_INF	0.95	0.91	0.93	2,746
V_VM_VNF_RP	0.96	0.95	0.95	4,806
V_VM_VNF_RP_PSP	0.92	0.87	0.89	498
V_VM_VNF_VBN	0.96	0.96	0.96	9,437
V_VM_VNG	0.27	0.33	0.30	9
Average / Total	0.96	0.96	0.96	134,189

In Decision Tree classifier model and Naïve Bayes classifier model some of the tags have not been identified at all, which got 0% of precision, recall and F1-score. But SVM classifier model identified those tags with less precision, recall and F1-score. So this shows that SVM classifier based POS tagger performed better than those two classifier based model. So the SVM model is selected as the base line model for this research. The deep learning model is expected to give better result than this SVM classifier model.

5.1.4. Result of LSTM POS Tagger

Long Short Term Memory (LSTM) model was built for POS tagging as mentioned in the implementation section. The whole AUKBC Tamil POS corpus was given to this model. The corpus is divided into training and testing dataset. The training dataset contains 38,705 sentences and testing dataset contains 12,902 sentences. The model was trained with one, three and five epochs separately.

5.1.4.1. One Training Epoch

The result obtained for model with one training epoch is given in the table 5.4.

Table 5.4: Classification Report of the LSTM POS tagger with one training epoch

Tags	Precision	Recall	F1-Score	Support
CC_CCD	0.96	1.00	0.98	636
CC_CCS	0.96	0.99	0.97	2,759
DM_DMQ	0.96	0.91	0.93	3,007
DM_DMR	1.00	0.99	1.00	1,556
JJ	0.77	0.83	0.80	2,262
N_NN	0.9	0.97	0.96	25,738
N_NNP	0.97	0.89	0.93	8,337
N_NNV	0.00	0.00	0.00	4
N_NST	0.57	0.52	0.54	369
PR_PRC	0.00	0.00	0.00	20
PR_PRF	0.76	0.94	0.84	609
PR_PRL	0.00	0.00	0.00	57
PR_PRP	0.99	0.99	0.99	12,654
PR_PRQ	0.00	0.00	0.00	134
PSP	0.90	0.84	0.87	2,510
QT_QTC	0.97	0.96	0.96	1,095
QT_QTF	0.99	0.98	0.99	266
QT_QTO	0.31	0.58	0.40	59
RB	0.89	0.87	0.88	5,116
RD_ECH	0.00	0.00	0.00	162
RD_PUNC	1.00	0.99	1.00	19,846
RD_SYM	0.97	1.00	0.98	5,920
RP_INJ	0.94	0.98	0.96	434
RP_INTF	0.65	0.61	0.63	160
RP_NEG	0.91	0.90	0.91	2,478
RP_NEG_PSP	0.00	0.00	0.00	6
V_VM_VF	0.93	0.96	0.95	10,467
V_VM_VNF_COND	0.71	0.60	0.65	625
V_VM_VNF_INF	0.89	0.88	0.89	2,241
V_VM_VNF_RP	0.88	0.91	0.89	4,398
V_VM_VNF_RP_PSP	0.91	0.03	0.05	359
V_VM_VNF_VBN	0.92	0.97	0.95	8,594
V_VM_VNG	0.00	0.00	0.00	3
Average / Total	0.94	0.95	0.94	122,881

The accuracy of 94.51% is obtained for the LSTM model with one training epoch. The average precision obtained for this model is 0.94, recall is 0.95, and F1- score is 0.94. When analyzing the classification report given in the table 5.4 it is clear that most of the tags were identified with F1-score more than 50% except Ordinal quantifier (QT_QTO) tag which got the F1-score of 40%. Also some of the tags have not been identified at all, which give the 0% precision, recall and F1-score. Those tags are Verbal Noun (N_NNV), Reciprocal Pronoun (PR_PRC), Relative pronoun (PR_PRL), Wh–word pronoun (PR_PRQ), Residual echo word (RD_ECH) and Gerund (V_VM_VNG). These tags were appeared in the corpus very less amount of time. That is not enough to be learned by the LSTM model. Because of that these tags could not be identified.

5.1.4.2. Three Training Epochs

The result obtained for the LSTM model with three training epochs is given in the table 5.5.

The accuracy of 96.43% is obtained for the LSTM model with three training epochs. Table 5.5 gives the classification report of this model. The average precision obtained for this model is 0.96, recall is 0.96, and F1- score is 0.96. Most of the tags have been correctly identified with F1- score more than 90%. The tags Adjective (JJ), Preposition (PSP), Reflexive pronoun (PR_PRF), Relative pronoun (PR_PRL) and Conditional verb (V_VM_VNF_COND) have the F1- score between 80% and 90%. The Location noun (N_NST) tag is identified with 56% of F1- score and Ordinal Quantifier (QT_QTO) tag is with 77% of the F1- score. Residual echo word (RD_ECH) tag has very low F1- score 18%. The tags Verbal Noun (N_NNV), Reciprocal Pronoun (PR_PRC) and Gerund (V_VM_VNG) are not identified at all, which have precision, recall and F1-score 0%. The reason behind this is the number of tags appeared in the corpus. The above mentioned three tags have been appeared less than fifty times in the corpus. So this amount is not enough to be learned by the LSTM model with three training epochs.

Table 5.5: Classification Report of the LSTM POS tagger with three training epochs

Tags	Precision	Recall	F1-Score	Support
CC_CCD	0.99	1.00	0.99	636
CC_CCS	0.98	0.99	0.98	2,759
DM_DMQ	0.95	0.95	0.95	3,007
DM_DMR	1.00	0.99	1.00	1,556
JJ	0.79	0.89	0.84	2,262
N_NN	0.97	0.97	0.97	25,738
N_NNP	0.97	0.93	0.95	8,337
N_NNV	0.00	0.00	0.00	4
N_NST	0.58	0.54	0.56	369
PR_PRC	0.00	0.00	0.00	20
PR_PRF	0.81	0.98	0.88	609
PR_PRL	0.98	0.75	0.85	57
PR_PRP	0.99	1.00	0.99	12,654
PR_PRQ	0.87	0.25	0.38	134
PSP	0.92	0.85	0.88	2,510
QT_QTC	0.95	0.95	0.95	1,095
QT_QTF	0.98	0.98	0.98	266
QT_QTO	0.66	0.93	0.77	59
RB	0.91	0.92	0.91	5,116
RD_ECH	0.43	0.12	0.18	162
RD_PUNC	1.00	0.99	1.00	19,846
RD_SYM	0.97	1.00	0.98	5,920
RP_INJ	0.95	0.98	0.96	434
RP_INTF	0.99	0.91	0.94	160
RP_NEG	0.97	0.96	0.97	2,478
RP_NEG_PSP	0.00	0.00	0.00	6
V_VM_VF	0.97	0.97	0.97	10,467
V_VM_VNF_COND	0.85	0.87	0.86	625
V_VM_VNF_INF	0.96	0.96	0.96	2,241
V_VM_VNF_RP	0.95	0.94	0.94	4,398
V_VM_VNF_RP_PSP	0.74	0.78	0.76	359
V_VM_VNF_VBN	0.98	0.98	0.98	8,594
V_VM_VNG	0.00	0.00	0.00	3
Average / Total	0.96	0.96	0.96	122,881

When comparing the results obtained by the LSTM model with one and three training epochs it is clear that some of the tags which were not identified by the model with one training epoch are identified by the model with three training epochs, but with less accuracy. Those tags are Relative pronoun (PR_PRL), Wh–word pronoun (PR_PRQ) and Residual echo word (RD_ECH). So if the number of training epochs are increased the model might be able to identify all the tags more precisely.

5.1.4.3. Five Training Epoch

From the above conclusion the model is trained with five training epochs. The result obtained for model with five training epoch on the LSTM model is given in the table 5.6.

Table 5.6: Classification Report of the LSTM POS tagger with five training epochs

Tags	Precision	Recall	F1-Score	Support
CC_CCD	0.98	1.00	0.99	636
CC_CCS	0.98	0.99	0.98	2,759
DM_DMQ	0.96	0.96	0.96	3,007
DM_DMR	1.00	0.99	1.00	1,556
JJ	0.79	0.89	0.84	2,262
N_NN	0.96	0.98	0.97	25,738
N_NNP	0.97	0.93	0.95	8,337
N_NNV	0.00	0.00	0.00	4
N_NST	0.62	0.62	0.62	369
PR_PRC	0.86	0.60	0.71	20
PR_PRF	0.81	0.99	0.89	609
PR_PRL	1.00	0.77	0.87	57
PR_PRP	0.99	1.00	0.99	12,654
PR_PRQ	1.00	0.40	0.57	134
PSP	0.92	0.86	0.89	2,510
QT_QTC	0.96	0.96	0.96	1,095
QT_QTF	0.98	0.98	0.98	266
QT_QTO	0.69	0.93	0.79	59
RB	0.93	0.92	0.92	5,116
RD_ECH	0.55	0.25	0.34	162
RD_PUNC	1.00	0.99	1.00	19,846
RD_SYM	0.97	1.00	0.98	5,920
RP_INJ	0.95	0.99	0.97	434
RP_INTF	0.98	0.91	0.94	160
RP_NEG	0.97	0.97	0.97	2,478
RP_NEG_PSP	0.67	0.33	0.44	6
V_VM_VF	0.98	0.97	0.97	10,467
V_VM_VNF_COND	0.93	0.88	0.91	625
V_VM_VNF_INF	0.97	0.96	0.96	2,241
V_VM_VNF_RP	0.95	0.94	0.95	4,398
V_VM_VNF_RP_PSP	0.88	0.89	0.88	359
V_VM_VNF_VBN	0.98	0.98	0.98	8,594
V_VM_VNG	0.00	0.00	0.00	3
Average / Total	0.97	0.97	0.97	122,881

The accuracy of 96.74% is obtained for the LSTM model with five training epochs. Table 5.6 gives the classification report of the LSTM model with five training epochs. The average precision obtained for this model is 0.97, recall is 0.97, and F1-score is 0.97. The precision, recall and F1-socre for each of the tag obtained with this

five training epoch is almost same as the result obtained from the three training epoch model. The tag Reciprocal pronoun (PR_PRC) and the tag RP_NEG_PSP is not identified with the three training epoch model. But it is identified with this model with 71% and 44% of F1-score respectively. Also slight improvement in some tags is achieved with this model. Those tags are Adjective (JJ), Reflexive pronoun (PR_PRF), Location noun (N_NST), Wh–word pronoun (PR_PRQ), Ordinal Quantifier (QT_QTO), Wh-word demonstrative (DM_DMQ), Relative pronoun (PR_PRL), Preposition (PSP), Cardinal quantifier (QT_QTC), Adverb (RB), Interjection particles (RP_INJ), Conditional verb (V_VM_VNF_COND) and Relative participle (V_VM_VNF_RP).

The LSTM model give better performance when it was trained with five training epochs. Also this model give slightly better performance than the SVM model which is the baseline of this research. The improvement of 1.038% is achieved with the LSTM model in this research.

5.2. Experiment with Different size of the Dataset

In addition to the above mentioned experiments the implemented models was evaluated with different size of the dataset. For this purpose 25%, 50% and 75% of the corpus is used. This experiment is done to study the behavior of the models with different size of the dataset. Sentences are randomly selected from the original corpus to create this different size corpus. All of the above mentioned models have been trained and tested with the randomly selected corpus. The 25% of the corpus contains 12,901 sentences, 50% contains 25,803 sentences and 75% contains 38,703 sentences. The results obtained for each of the model with the 25% of the corpus is given in the table 5.7. The result for the 50% corpus is given in the table 5.8 and with 75% is given in the table 5.9.

Table 5.7: Results for 25% of the corpus

Model	Precision	Recall	F1-Score	Accuracy
Decision Tree Classifier	0.85	0.85	0.84	84.54
Naïve Bayes Classifier	0.86	0.85	0.84	85.08
SVM Classifier	**0.95**	**0.95**	**0.95**	**95.40**
LSTM with one epoch	0.73	0.75	0.72	75.16
LSTM with three epoch	0.90	0.90	0.90	89.88
LSTM with five epoch	0.92	0.91	0.91	90.78

Table 5.8: Results for 50% of the corpus

Model	Precision	Recall	F1-Score	Accuracy
Decision Tree Classifier	0.86	0.85	0.85	85.33
Naïve Bayes Classifier	0.88	0.88	0.87	87.93
SVM Classifier	**0.97**	**0.97**	**0.96**	**96.53**
LSTM with one epoch	0.88	0.88	0.87	87.93
LSTM with three epoch	0.94	0.94	0.94	93.64
LSTM with five epoch	0.94	0.94	0.94	93.92

Table 5.9: Results for 75% of the corpus

Model	Precision	Recall	F1-Score	Accuracy
Decision Tree Classifier	0.85	0.85	0.85	85.16
Naïve Bayes Classifier	0.90	0.90	0.89	89.63
SVM Classifier	**0.97**	**0.97**	**0.97**	**97.41**
LSTM with one epoch	0.92	0.92	0.92	92.15
LSTM with three epoch	0.95	0.95	0.95	94.94
LSTM with five epoch	0.95	0.95	0.95	95.27

In all these dataset the baseline SVM model outperform all other models. But the LSTM model also not perform worse with these dataset. The proposed model got more than 90% of the accuracy with all the datasets. The LSTM model with five training epochs got 90.78% of accuracy with 12,901 sentences, 93.92% of accuracy with 25,803 sentences and 95.27% of accuracy with 38,703 sentences. So this LSTM model could be able to use with some small size dataset also to get good performance.

Chapter 6

Conclusion

Part of Speech tagging is one of the basic and important task for many Natural Language Processing applications. The accuracy of the POS tagger have influence in the performance of many NLP applications. So having a good POS tagger is a crucial task for all the languages. Tamil is one of the morphologically rich, agglutinative and free word order language widely used by native Tamil speakers. Currently there are no freely available state of the art POS taggers for Tamil language. There were few researches have been done for Tamil language POS tagging but none of them have the accuracy compared to other language state of the art POS taggers. Therefore, this research is done to improve the accuracy of the POS tagger for Tamil language using one of the deep learning approaches that have been used in other language state of the art POS taggers. The LSTM model is used to build the POS tagger for Tamil language and evaluated with the AUKBC Tamil POS corpus, which is the large manually annotated POS corpus for Tamil language.

Few POS taggers have been built for Tamil language as mentioned in the literature review. Most of them have used the statistical approaches. So the accuracy of those taggers depend on the corpus they have used to evaluate. Since there is no work in the literature which used the AUKBC Tamil POS corpus, to evaluate the performance of the proposed model, a baseline model with one of those approaches is necessary. Therefore, the baseline model is also built in this research. Three classification based

models have been built for baseline model using the Decision Tree classifier, Naïve Bayes classifier and Support Vector Machine classifier. SVM classifier based POS tagging model is selected as the baseline by considering the result obtained with the corpus.

6.1. Conclusions about the Research Questions

This research is done with the goal of improving the accuracy of POS tagging for Tamil language. When going through the POS tagging works done for other languages there were many state of the art POS taggers available which used the deep learning approaches. So based on that, the question of 'How deep learning approaches can help to improve the accuracy in Part of Speech tagging for Tamil language?' is raised. So this research is done to address this question. The result obtained for the LSTM model built in this research was able to answer this question. The table 6.1 shows the result obtained for the SVM model and LSTM model.

Table 6.1: The result of SVM and LSTM model

Model	Precision	Recall	F1-Score	Accuracy
SVM	0.96	0.96	0.96	95.70%
LSTM	**0.97**	**0.97**	**0.97**	**96.74%**

When observing the results, it is clear that LSTM model outperform the SVM model on POS tagging for Tamil language. The improvement of 1.04% is achieved by the LSTM model on the AUKBC Tamil POS corpus. So this research ensure that deep learning approaches can be used for Tamil language POS tagging to improve the performance.

The experiment done with the different size of the corpus lead to the conclusion that deep learning approaches give the state of the art accuracy when the size of the corpus is large enough. When the size of the dataset is decreasing the baseline SVM model perform better than the LSTM model as mentioned in the table 6.2. 25%, 50% and 75% of the original corpus is used for this experiment. The 25% of the corpus contains 12,901 sentences, 50% contains 25,803 sentences and 75% contains 38,703 sentences. The accuracy of each of the model with this different size corpus is given in the table 6.2.

Table 6.2: The result of models with different size of corpus

Model	25% of Corpus	50% of Corpus	75% of Corpus	100% of Corpus
Decision Tree	84.54	85.33	85.16	81.29
Naïve Bayes	85.08	87.93	89.63	88.73
SVM	**95.40**	**96.53**	**97.41**	95.70
LSTM with one training epoch	75.17	87.93	92.15	94.51
LSTM with three training epoch	89.88	93.64	94.94	96.43
LSTM with five training epoch	90.78	93.92	95.27	**96.74**

The above table shows the accuracy obtained for each of the models with 25%, 50%, 75% and 100% of the corpus. The baseline SVM classifier based POS tagger outperform all other models when the model is evaluated with the 25%, 50% and 75% of the corpus. The LSTM model give better performance than all other models with the whole dataset.

Even though the proposed LSTM model not give the state of the art POS tagging accuracy with less number of sentences it could be able to give the accuracy more than 90%. So this LSTM model can be used with the corpus which have less amount of sentences to get good performance.

Tamil is an agglutinative, morphologically rich language and free word order language. Languages with these nature lack good POS taggers as well as large annotated corpus. Since this LSTM model give good performance with less number of sentences, this model could be able to use for other morphologically rich and agglutinative languages with less resource.

6.2. Limitations

This research is mainly based on the AUKBC Tamil POS corpus which is annotated form a historical novel. The accuracy of the tagger obtained in this research is limited to the above mentioned corpus only. So the performance of the tagger may vary when it is used to tag the corpus in some other domain.

Tamil language have a complex grammatical structure. So there might be many POS categories to handle the complex structure of Tamil. This research is based on the BIS tagset, which have only 37 POS tags. This tagset only consider the lexical aspects of the language and don't mark the plurality and gender distinctly. Therefore, this might be a limitation.

6.3. Implications for Further Research

This research is mainly focused on whether the deep learning approaches can give better accuracy than the traditional methods that have been used for Tamil language POS tagging. The training and the evaluation of the proposed model have been done only using the AUKBC Tamil POS corpus. So the result is only depends on this corpus. So further improvements are needed for the work done in this research. Some of the different directions are given below to improve POS tagging for Tamil language further.

- Evaluate the model with some other Tamil POS corpus to check the performance and robustness of the proposed LSTM model.

- There are many deep learning approaches that can be used for POS tagging. So other approaches can be tried out with this corpus.

- Hyper parameters of the model is set to some values in this research based on the literature review. So changing the hyper parameters of the LSTM network model and evaluate the performance with the AUKBC Tamil POS corpus can also be done.

References

[1] "Tamil language", Encyclopedia Britannica, 2017. [Online]. Available: https://www.britannica.com/topic/Tamil-language. [Accessed: 06- Oct- 2017].

[2] "Tamil language", En.wikipedia.org, 2017. [Online]. Available: https://en.wikipedia.org/wiki/Tamil_language. [Accessed: 03- May- 2017].

[3] S. Janarthanam, U. Nallasamy, L. Ramasamy and C. Santhoshkumar, "Robust dependency parser for natural language dialog systems in Tamil," *Proceedings of the 5th Workshop on Knowledge and Reasoning in Practical Dialogue Systems,* pp. 1-6, 2007.

[4] S. Rajendran, "Resolution of Lexical Ambiguity in Tamil," *Language in India ,* vol. 14, pp. 271-291, 2014.

[5] L. D. Sobha , G. Sindhuja , L. Gracy, N. Padmapriya, A. Gnanapriya and N. H. Parimala, "AUKBC Tamil Part-of-Speech Corpus (AUKBC-TamilPOSCorpus2016v1).," Computational Linguistics Research Group, AU-KBC Research Centre, Chennai, India, 2016.

[6] V. Ranganathan, "Development of Morphological Tagger for Tamil," *Tamil Internet 2001 Conference, Kuala Lumpur, Malaysia* , pp. 1-4, 2001.

[7] S. L. Pandian and T. V. Geetha, "Morpheme based Language Model for Tamil Part-of-Speech Tagging," *Polibits,* vol. 38, pp. 19-25, 2008.

[8] M. Ramanathan, V. Chidambaram and A. Patro, "An Attempt at Multilingual POS Tagging for Tamil".

[9] P. Arulmozhi and L. Sobha, "HMM based POS Tagger for a Relatively Free Word Order Language," *Research in Computing Science,* vol. 18, pp. 37-48, 2006.

[10] V. Dhanalakshmi, A. Kumar, G. Shivapratap, K. P. Soman and S. Rajendran, "Tamil POS Tagging using Linear Programming," *International Journal of Recent Trends in Engineering,* vol. 2, pp. 166-169, 2009.

[11] V. Dhanalakshmi, M. Anand kumar , S. Rajendran and K. P. Soman, "POS tagger and chunker for Tamil language," *Proceedings of Tamil Internet Conference,* 2009.

[12] S. L. Pandian and T. V. Geetha, "CRF models for tamil part of speech tagging and chunking," *International Conference on Computer Processing of Oriental Languages,* vol. LNAI 5459, pp. 11-22, 2009.

[13] M. Selvam and A. M. Natarajan, "Improvement of rule based morphological analysis and pos tagging in tamil language via projection and induction techniques," *International journal of computers,* vol. 3, no. 4, pp. 357-367, 2009.

[14] P. Arulmozhi, T. R. K. R. Pattabhi and L. Sobha, "A Hybrid POS Tagger for a Relatively Free Word Order Language," *Proceedings of the First National Symposium on Modeling and Shallow Parsing of Indian Languages,* pp. 79-85, 2006.

[15] C. D. Santos and B. Zadrozny, "Learning Character-level Representations for Part-of-Speech Tagging," *In Proceedings of the 31st International Conference on Machine Learning (ICML-14)* , vol. 32, pp. 1818-1826, 2014.

[16] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu and P. Kuksa, "Natural language processing (almost) from scratch," *Journal of Machine Learning Research,* vol. 12, pp. 2493-2537, 2011.

[17] B. Plank, A. Søgaard and Y. Goldberg, "Multilingual part-of-speech tagging with bidirectional long short-term memory models and auxiliary loss," *arXiv:1604.05529v3 [cs.CL]*, 2016.

[18] T. Brants, "TnT: a statistical part-of-speech tagger," *Proceedings of the sixth conference on Applied natural language processing* , pp. 224-231, 2000 April.

[19] B. Plank, D. Hovy, R. Mcdonald and A. Søgaard, "Adapting taggers to Twitter with not-so-distant supervision," *COLING*, pp. 1783-1792, 2014.

[20] P. Wang, Y. Qian, F. K. Soong, L. He and H. Zhao, "Part-of-Speech Tagging with Bidirectional Long Short-Term Memory Recurrent Neural Network," 2015.

[21] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Gramfort, B. Thirion, O. Grisel and J. Vanderplas, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825-2830, 2011.