2014 International Conference on Advances in ICT for Emerging Regions (ICTer): 197 - 203

A Shallow Parser for Tamil

I. Ariaratnam, A. R. Weerasinghe, C. Liyanage

University of Colombo School of Computing, Sri Lanka.

avifancy@gmail.com

arw@ucsc.cmb.ac.lk

cml@ucsc.cmb.ac.lk

Abstract— **This research is an attempt to build a shallow parser designed to assign a partial structure to natural language sentences in order to recover useful syntactic information from Sri Lankan Tamil sentences. It uses a combination of a maximum entropy based part-of-speech (POS) tagger which automatically labels each word in a sentence with the appropriate POS tag, and a rule-based chunker which segments the sentences into syntactically correlated word groups, without the need for a large annotated corpus. To do this, we developed a POS tagset consisting of 20 POS tags using expert input, manually annotated a corpus of approximately 12500 words, and identified 390 chunk patterns to extract the chunks. Our POS tagger and chunker demonstrated promising f-measures of 81.72% and 78.3% respectively. Our combined shallow parser gives an f-measure of 66.6% owing to error propagation.**

Index Terms—**Chunking, Maximum Entropy Model, Partial Parsing, Part-of-speech Tagging, Shallow Parsing, Tamil Language Processing**

I. INTRODUCTION

Natural language plays a most significant role in the communication between human beings. The electronic version of natural language text has become a key phenomenon in the process of communication with the revolution of electronic and information media in global communication. On the Internet, there is a lot of textual data available in human languages. This imposes a need to find an easy way to access the relevant, useful information and understand it in context without ambiguities. Since the electronic version of the information is in the form of natural languages, language specific knowledge is crucial for processing information. This necessitates the need of dealing with computational aspects of human languages.

Understanding of natural language sentences requires an extensive knowledge about the world and the ability to manipulate such knowledge. Though algorithms capable of parsing simple sentences for determining syntax exist, they mostly fail in parsing long sentences, since no complete grammar is available for many human languages.

"Shallow Parsing" is an attempt to address this problem. Instead of trying to determine a full parse tree for a given sentence, it is designed to assign a partial structure to such sentences in order to recover a limited but useful amount of syntactic information. It does not solve all the semantically significant ambiguities; instead it identifies the most important phrases from sentences. The key idea is that this is often sufficient for many NLP applications such as information extraction, information retrieval, and machine translation.

The initial processing step of shallow parsing is part-of-speech (POS) tagging, a process of automatically labeling each word in a natural language sentence to the corresponding POS tag such as noun, verb, adjective, adverb, pronoun, preposition or other lexical class marker [7]. The POS tag of a word is not only based on the word itself but also depends on the context of the sentence; hence the difficulty comes into play when assigning appropriate POS tags to words in a sentence. For example, a word object can be treated as a noun or a verb, depending on the context in which it is used.

After the completion of the POS tagging process, a next step is the division of the natural language sentence into chunks consisting of grammatical units such as noun phrases, verb phrases, and prepositional phrases [4]. Usually such chunk structures are non-recursive, i.e., one particular word can only be a member of a single chunk, while all the words in a sentence need not belong to a chunk or phrase. Extracting the chunks from the sentences is a useful level of analysis for several tasks such as named entity recognition, relation extraction, information retrieval, and machine translation.

II. RELATED WORK

Tamil is a Dravidian language and one of the classical languages spoken by the people from South India and Sri Lanka. It is also spoken in Malaysia, Canada, South Africa, Fiji, Germany, USA, Netherlands, and Mauritius as well as other emigrant communities around the world. Tamil has an agglutinative grammar structure where suffixes are used to denote number, case, tense, and other grammatical categories. Tamil words are made up of a lexical root followed by one or more affixes. Most of the Tamil affixes are suffixes, which are inflectional because of its morphological richness. Although the Tamil language is a relatively free word order language, there often is a default order which is dominant.

Northern Sri Lankan Tamil dialects are distinct from the dialects of modern Tamil spoken in the Tamil Nadu and Kerala states of India. Sri Lankan Tamil dialects retain many words and grammatical forms that are not in everyday use in India and use many other words slightly differently. Sri Lankan Tamil dialects are less influenced by Sanskrit and western languages, although there are western and Sanskrit loan words in day to day usage. In general, Sri Lankan Tamil dialects are considered to be more conservative than their continental counterparts.

According to the literature, the following are the main attempts carried out for chunking in Tamil.

978-1-4799-7732-1/14/$31.00 ©2014 IEEE

- A chunker for Tamil named "Vaanavil" was developed by K. Saravanan et al. in 2004 [6]. It uses a look-ahead approach to handle free word order. It deals with ambiguity using 15 heuristic rules. It uses the morphological analyzer to obtain the root word.
- In 2006, a noun phrase chunker for Tamil was developed using a rule-based approach by L. Sobha et al. from AU-KBC Research Center, India [5].
- In 2009, V. Dhanalakshmi et al. from Amrita University, Coimbatore, India developed a chunker for Tamil using a machine learning technique, Support Vector Machines. This research reported an overall accuracy 95.82% [1].

For English and other European languages, many successful POS taggers and chunkers have been developed compared with Indic languages. Comparatively little research has been carried out for implementing a chunker for Tamil because of the richness in vocabulary, morphological structure, and limited availability of annotated corpora. Only a small portion of the work has been reported so far and very few linguistic resources publicly available. Also no attempts have been made for chunking Sri Lankan Tamil.

III. Data Preparation

In recent years, machine learning techniques have become popular for solving many NLP tasks. The reason is that the algorithms used can automatically and efficiently learn from natural language data given a correctly annotated training corpus. The main resource needed for the machine learning process is such linguistically annotated data. Unfortunately, there is no such annotated corpus publicly available for Tamil. Due to this unavailability, we focused on developing an annotated corpus for Sri Lankan Tamil, with the view to creating a tagging model capable of labeling each word in a Tamil sentence to the appropriate POS tags depending on its context. For the task of identifying the basic structural relations between groups of such tagged words, we used a rule-based approach owing to data sparseness issues.

A. Corpus selection

At present, there is no annotated corpus publicly available for Sri Lankan Tamil. Due to this unavailability, a raw text corpus of size approximately 500,000 words collected from Sri Lankan Tamil newspapers such as Veerakesari, Thinakural, and Tamil Mirror, prepared by the Language Technology Research Laboratory (LTRL) at the University of Colombo School of Computing (UCSC), Sri Lanka, was used to prepare a POS annotated corpus.

B. Pre-processing (Normalization)

Sentence segmentation was performed as the first pre-processing step on the raw text corpus to split the data into sentences. Then the sentences which have 20 or less words were extracted in order to ease the process of POS tagging regardless of the domains such as politics, sports, art, and literature. Then, manual pre-editing of the corpus was performed to get rid of unwanted white spaces and spelling mistakes.

C. Proposed POS tagset

Based on the POS tag sets described in the literature for Tamil, and considering the standard tagset of POS annotation for all Indian languages[1], our own POS tagset which consists of 20 POS tags was developed in consultation with a linguist. This list, shown in TABLE I, was designed according to the final objective of building a shallow parser for Tamil.

Here, only grammatical categories were considered and not grammatical features, since a morphological analyzer for Tamil under development at the UCSC is expected to handle such features.

TABLE I
OUR POS TAGSET

Description	POS Tag	Example
1. Noun	NN	மனிதன்
2. Compound Noun	NNC	வாழ்க்கைப் பதிவு
3. Proper Noun	PRP	மாலா
4. Verbal Noun	VBN	வாழ்க்கை
5. Pronoun	NNPC	இது
6. Adjective	ADJ	உயரமான
7. Adverb	ADV	வேகமாக்
8. Quantifier	QTF	இரண்டு
9. Particle	PAR	தான்
10. Post Position	PPO	பக்கத்தில்
11. Finite Verb	VF	வாருங்கள்
12. Non-finite Verb	VNF	படிக்கின்ற
13. Infinitive Verb	VINT	சிரிக்க
14. Gerund Verb	VBG	படித்தல்
15. Auxiliary Verb	VAX	முடியும்
16. Determiner	DET	அந்த
17. Punctuations	PU	!
18. Conjunction	CNJ	அதனால்
19. Residuals	RD	Foreign words
20. Unknown words	UNK	Others

D. Manual tagging

A relatively small sized POS annotated corpus consisting of approximately 12500 words was manually created and then divided into two sets: 10000 words as training data and 2500 words as test data. This POS tagged corpus is clean and completely annotated. To verify the correctness of the annotation of the words, post-editing of the tagged corpus was also performed.

IV. POS Tagger

Maximum Entropy (ME) modeling, a popularly used statistical modeling technique, was employed to predict the most probable POS tag sequence for each sentence in the test data. It calculates the probabilities based on the history extracted from the training data, stating some relations among features and outcome. The ME principle states that given a set of observations/histories, the most probable underlying probability distribution is that which has the least bias - namely, maximum entropy - while confirming the statistical properties measured on the set of observations/histories.

It has been proven empirically that these ME models serve as a successful supervised machine learning method to classification problems in linguistics, in which contexts are

[1] http://shiva.iiit.ac.in/SPSAL2007/iiit_tagset_guidelines.pdf

used to predict linguistic classes. The reasons for the adoption of this model are:

- It permits the presence of diverse sources of evidence without producing fragmentation and without necessarily assuming independence between the predictors.
- It is simple and effective for even relatively sized training datasets.

A. Probability model

The probability model $P(t/h)$ characterizes the conditional probability of a POS tag $t \in T$ where T is the set of allowable tags, given context/history $h \in H$ where H is the set of possible word and tag contexts.

The probability model is represented as [3]:

$$P(t/h) = \frac{1}{Z(h)} \prod_{j=1}^{k} \alpha_j^{f_j(t,h)} \quad (1)$$

where, each parameter α_j corresponds to a feature f_j and $Z(h)$ is a normalization function. The problem of POS tagging can be formally stated as follows: given a sequence of words $w_1, w_2, \ldots w_n$, we need to find the most probable sequence of POS tags $t_1, t_2, \ldots t_n$, which satisfies:

$$P\left(\frac{t_1, t_2 \ldots t_n}{w_1, w_2 \ldots w_n}\right) = \prod_{i=1,2\ldots n} P(t_i/h_i) \quad (2)$$

where h_i is the context available for the word w_i.

The ME model assigns POS tags, one sentence at a time. The process of assigning POS tags necessitates a search procedure to compute the candidate tag sequences for the given sentence, and the tag sequence with the highest probability is chosen by the Beam search algorithm as the final tag sequence.

B. Implementation

Implementation of the POS tagger based on ME modeling was initiated with the baseline model introduced by Adwait Ratnaparkhi [2] using the Java-based Apache OpenNLP[2] toolkit.

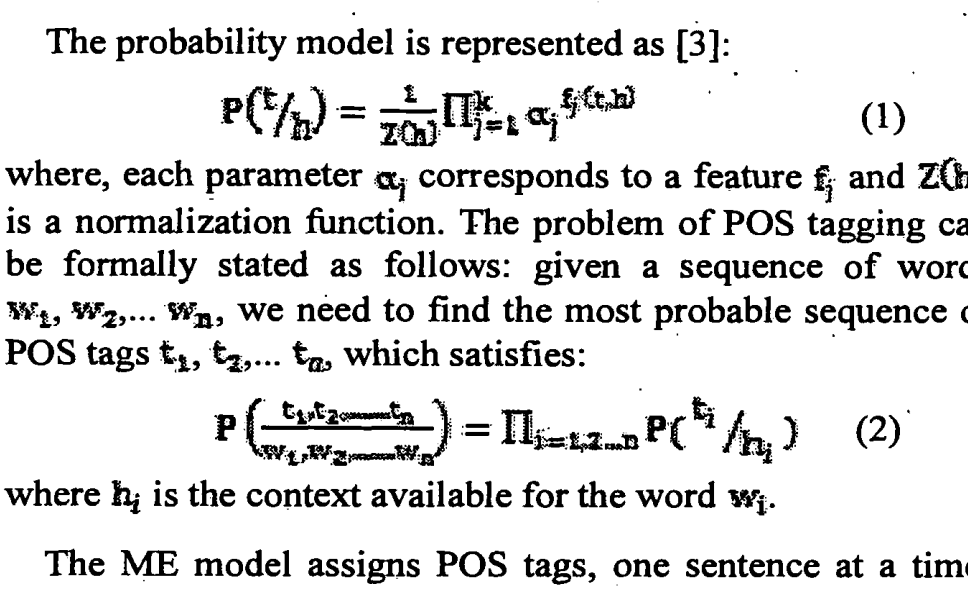

Fig. 1. Part-of-speech tagger based on Maximum Entropy model

Fig.1 represents the overall architecture of the POS tagger based on the ME model. Manually POS annotated training data and the selected feature set for Tamil were given to the ME trainer for the purpose of creating a model file. The ME model uses the Beam search algorithm as a disambiguation algorithm to decide the most likely sequence of POS tags which has the higher probability, one sentence at a time. A tag dictionary can list the possible POS tags for each "closed-class" word appearing within the training data. Thus, the search procedure for known words generates only the tags given by the tag dictionary. On the other hand, for unknown words, it generates all the tags in the tagset. The model file and tag dictionary were given to the Maximum Entropy tagger for predicting the POS tags for the untagged test data to produce the POS annotated output data.

Finally the output data from the tagger was evaluated against the model file produced from the manually tagged test data, for measuring the accuracy of the POS tagger.

C. Features for Tamil

Selecting appropriate features particular to a language generally involves identifying a set of significant characteristics which provide strong evidence for the classification of POS tags. To find out the most appropriate set of features for Sri Lankan Tamil, a comprehensive analysis was carried out by looking at the POS annotated corpus. The following are the features that were incorporated in the implementation of the tagger. The precision of the POS tagger was measured by incorporating different groupings from the following set of features.

1) Context of the word:

A key problem in the area of NLP is word sense disambiguation. With the use of contextual information, some of the word level ambiguities can be resolved. This appears to be a useful static feature which considers the previous and next subsequent words of a particular word in order to find out the POS tag of that particular word precisely. For example in the sentence,

அவன் காயத்திற்கு தையல் போட்டான்.

He put stitches to the wound.

the word "தையல்" can take two forms, girl (noun) and stitch (noun/verb). The ambiguity between the two forms can be resolved when the word "காயத்திற்கு" (wound) is met. To resolve these kinds of ambiguities, defining a word-based feature within a contextual window is required.

2) POS tags of previous words:

This is a dynamic feature which captures some information from the POS tags of previous words of the word under consideration.

For example in the sentence,

அவன் தையலை அழைத்தான்

He called the girl / He called stitching

the word "தையல்" takes two POS tags, noun (girl) and verb (stitching). But based on the context of the sentence, it acts as a noun with the meaning of girl.

3) Prefix and suffix information:

Especially for highly inflectional languages such as Tamil, considering the prefix and suffix information of the current word can have a significant impact on the accuracy of the POS tagger.

- Suffix information - series of characters appearing at the end of the current word

[2]OpenNLP toolkit is available at: http://opennlp.apache.org/index.html

- Prefix information - series of characters appearing at the beginning of the current word

If a particular word is less than or equal to the number of suffix/prefix characters, this feature is undefined.

4) Tag dictionary (Lexicon):

This specifies the POS tags which a particular token can have. Two important advantages:

- Inappropriate tags cannot be assigned to tokens which appear in the tag dictionary.
- Beam search algorithm has to consider a smaller number of possibilities and thus can search faster.

It has been observed that for an unknown word, the POS information extracted from the lexicon is given more priority than the POS information assigned to that word by a ME model.

5) Digit information:

If all the characters in a token are digits, this feature is set to true, else it is set to false. This feature enables the tagger to identify and assign "QTF" (Quantifier) POS tag to the token which contains digits.

6) Symbol information:

If all the characters in a token are symbols, the feature is set to true, else it is set to false. This feature enables the tagger to identify and assign "RD" (Residual) POS tag to tokens which contain symbols.

In the ME model, a heuristic rule states that any word occurring less than 5 times in the training data can be considered as a rare word. Since the size of our POS annotated corpus is small, the cutoff point was set to 1 instead of the usual cutoff of 5.

V. CHUNKER

According to the literature, not much work has been done for chunking Tamil compared to the work conducted for Tamil POS tagging. The rule-based approach is very labor intensive, tedious, slow, and error-prone. A rule-based approach was employed for implementing a chunker for Tamil using Natural Language Toolkit 2.0 (NLTK), due to the unavailability of a large enough chunked corpus for Sri Lankan Tamil. Further, in phrasal and clausal construction, Tamil behaves as a fixed word order language though it is a free word order language. A rule-based approach was taken with a strong belief that it would result in a dataset usable for future research without preparing a chunk-annotated corpus in a manual way.

Noun phrase and verb phrase chunk were the two main categories focused on owing to their frequency.

1) Noun Phrase:

A noun phrase (NP) is a phrase whose head can be noun / compound noun / proper noun / verbal noun / pronoun. Noun qualifiers such as determiners, adjectives, quantifiers serve as optional pre-modifiers in a noun chunk and the head noun serves as the end of phrase marker. It includes non-recursive noun phrases and adjectival phrases. Examples for noun phrases are as follows:

1. அந்த/DET அழகிய/ADJ விசாலமான/ADJ பூங்கா/NN
2. முதலாவது/QTF முழு/ADJ சூரிய/NNC கிரகணம்/NNC
3. மாணிக்கவாசப்/PRP பெருமான்/PRP

2) Verb Phrase:

Verb phrases are usually categorized into verb finite chunks and verb non-finite chunks. Verb finite chunks contain a main verb and its auxiliaries with the chunk tag VFP. Verb non-finite chunks consist of non-finite verbs and infinitive verbs with the chunk tag VNFP. Examples for verb phrases are as follows:

1. பெற்று/VNF வருகிறது/VF
2. ஏற்படுத்துகிறது/VF
3. சிந்திக்கும்/VNF
4. வாங்க/VINT முடியாத/VNF

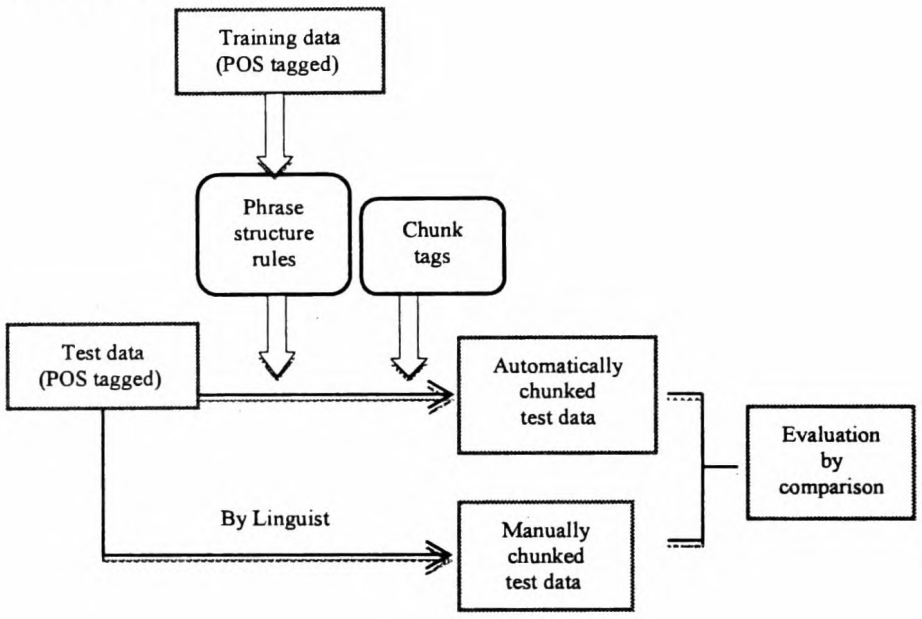

Fig. 2. Rule-based Chunker

Fig. 2 represents the architecture of our rule-based chunker. The POS tagged corpus was taken and divided into training and test data sets as usual. As an initial step, the following repetitive phases were been applied iteratively to the training data until adding new rules did not considerably increase the precision of the chunker.

1. Define/Refine phrase structure rules using efficient regular expressions for extracting noun and verb phrases to develop a regular expression parser.
2. Run the NLTK Regular Expression Parser on the training set which makes use of regular expressions over sequences of POS tags with the intention of identifying the range of text to be chunked.
3. Check the validity of the defined rules with the previous run.

After this, the phrase structure rules and chunk tags were used by a regular expression parser to chunk the test data which were already tagged with appropriate POS tags. Then the chunked test data were evaluated against a manually chunked version of the test data to measure the precision, recall, and f-measure values.

The following are examples of the 390 defined regular expressions rules for annotating the phrase level chunks.

A. Noun phrase chunk rules:

{<DET | NNPC>?<QTF>*<ADJ>*<NN>}
{<DET | NNPC>?<QTF>*<ADJ>*<NNC>+}
{<DET | NNPC>?<QTF>*<ADJ>*<VBN>}
{<DET | NNPC>?<ADJ>*<QTF>*<NN>}
{<DET | NNPC>?<ADJ>*<QTF>*<NNC>+}
{<DET | NNPC>?<ADJ>*<QTF>*<VBN>}
{<DET>?<QTF | ADJ>*<NN>}
{<DET>?<QTF | ADJ>*<NNC>+}
{<DET>?<QTF | ADJ>*<VBN>}
{<DET>?<ADJ>*<PRP>+}
{<NNPC>}

B. Verb phrase chunk rules:

Verb finite phrase - {<VNF | VINT>*<VF | VAX>+}
Verb non-finite phrase - {<VNF | VINT>+}

Finally, to represent the chunks in the data, the IOB notation was employed which encodes the chunk type, for instance I-NP for noun phrase words and I-VP for verb phrase words. As usual it denotes first word of the chunk by the notation B-Chunk and other words of the chunk by the notation I-Chunk. To indicate a word that is outside any chunks it uses the label "O".

VI. Evaluation and Results

A. POS Tagger

Our POS tagger based on the ME model was trained using a small sized annotated corpus of approximately 10,000 words and was evaluated against the test corpus of size approximately 2,500 words distinct from the training data.

A series of experiments was carried out with different sizes of training data (namely 8K, 9K and 10K words) to get an idea of the relative performance based on the size of the corpus while increasing the size of the training data.

1) Word and tag contextual features

To find a fair context that enables the model to correctly predict the POS tag of the current word, 23 different feature sets were prepared.

TABLE II shows the accuracies for the different combinations of word and tag level contexts relative to the size of the training data where W_i is current word and T_i is current POS tag. The 15th and 23rd word and tag contextual features improve the accuracy uniformly over three different sizes of the training data. Though the 20th feature set shows high accuracy, the improvement in the accuracy as we increase the size of the training data is insignificant. Thus we do not shortlist this feature as a good candidate. As such, only the 15th and 23rd features were taken for the next level of experimentation.

2) Prefix and suffix information

2.1 15th feature set (W_{i-2}, W_{i-1}, W_i, W_{i+1}, T_{i-1})

TABLE II
POS TAGGER RESULTS BASED ON THE WORD AND TAG CONTEXTUAL FEATURES

No.	Feature Set	8K	9K	10K
1	*Wi*−1, *Wi*	72.56%	73.96%	74.44%
2	*Wi*−2, *Wi*−1,*Wi*	72.80%	74.04%	74.80%
3	*Wi*−3,*Wi*−2, *Wi*−1,*Wi*	72.56%	74.16%	74.56%
4	*Wi*−4,*Wi*−3,*Wi*−2, *Wi*−1,*Wi*	72.68%	73.76%	73.92%
5	*Wi*, *Wi*+1	74.64%	75.88%	76.52%
6	*Wi*, *Wi*+1, *Wi*+2	74.84%	75.80%	76.44%
7	*Wi*, *Wi*+1, *Wi*+2,*Wi*+3	74.28%	75.44%	76.08%
8	*Wi*−2, *Wi*−1,*Wi*, *Wi*+1	74.40%	75.76%	76.40%
9	*Wi*−1,*Wi*, *Wi*+1	74.00%	75.20%	76.36%
10	*Wi*−2, *Wi*−1,*Wi*, *Wi*+1, *Wi*+2	75.00%	76.04%	76.16%
11	*Wi*−1,*Wi*, *Wi*+1, *Wi*+2	74.68%	75.84%	76.32%
12	*Ti*−1,*Wi*	72.92%	74.16%	75.04%
13	*Ti*−2,*Wi*	73.00%	74.92%	75.12%
14	*Ti*−2,*Ti*−1,*Wi*	72.16%	73.48%	74.76%
15	*Wi*−2, *Wi*−1, *Wi*, *Wi*+1, *Ti*−1	75.24%	76.04%	76.88%
16	*Wi*−2, *Wi*−1,*Wi*, *Wi*+1,*Ti*−2	74.16%	75.92%	76.56%
17	*Wi*−2, *Wi*−1,*Wi*, *Wi*+1,*Ti*−2,*Ti*−1	74.92%	76.08%	76.60%
18	*Wi*−1,*Wi*, *Wi*+1,*Ti*−1	74.96%	75.96%	76.24%
19	*Wi*−1,*Wi*, *Wi*+1,*Ti*−2	73.84%	75.96%	76.76%
20	*Wi*−1,*Wi*, *Wi*+1,*Ti*−2,*Ti*−1	75.12%	76.24%	76.56%
21	*Wi*,*Wi*+1,*Ti*−1	72.92%	74.16%	75.04%
22	*Wi*,*Wi*+1,*Ti*−2	74.04%	75.32%	76.16%
23	*Wi*, *Wi*+1, *Ti*−2, *Ti*−1	75.16%	75.72%	76.88%

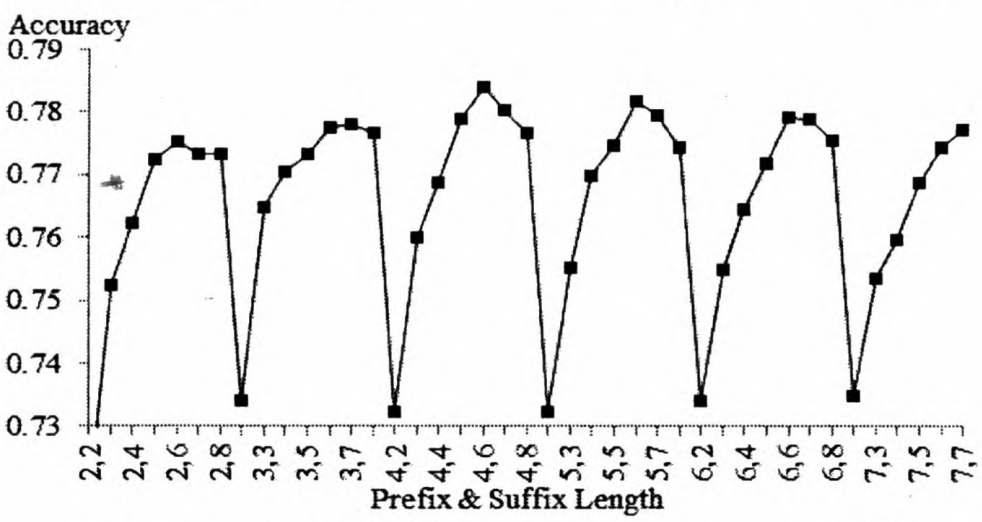

Fig. 3. POS tagger results based on the prefix and suffix information

2.2 23rd feature set (W_i, W_{i+1}, T_{i-2}, T_{i-1})

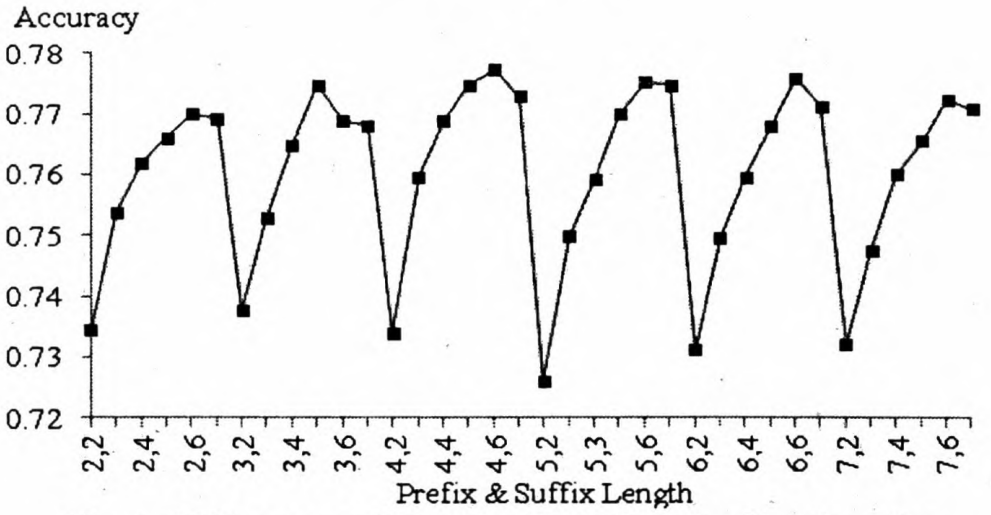

Fig. 4. POS tagger results based on the prefix and suffix information

According to Fig. 3 and Fig. 4, it was concluded a prefix length of 4 and suffix length of 6 are the optimum lengths appropriate for Tamil in both of the feature sets (15th and 23rd) for deciding the correct POS tag of the current word.

3) Tag Dictionary

To further improve the accuracy of the POS tagger, a tag dictionary, which contains only 480 entries (including punctuation marks) collected from the training data, was incorporated. It includes only closed-class words. The reason behind the choice of closed-class words is to increase the probability of assigning correct POS tags to unseen words where there is no obvious bias for the words in the training model by allowing the tagger to choose within the limited categories of POS tags. When it comes to closed-class words, the restricted list of POS tags have only one POS tag which leads to that word being tagged with that particular POS tag with probability one.

TABLE III presents the accuracies for both feature sets after the inclusion of prefix and suffix information, and tag dictionary as the size of the training data increases. For the maximum size of the training data (10K), the highest accuracy of 81.64% was obtained for the 15th feature set.

TABLE III: POS TAGGER RESULTS AFTER THE INCLUSION OF TAG DICTIONARY

Training Data	Feature Set 15: Accuracy	Feature Set 23: Accuracy
8 K words	80.20%	79.64%
9 K words	80.72%	80.40%
10 K words	**81.64%**	80.92%

4) Digit and symbol information

After examining the remaining instances of mis-tagging, another feature was added which identifies digits and symbols in a deterministic way. TABLE IV represents the results obtained after the addition of digit and symbol information as the size of the training data increases.

For the maximum size of the training data (10K), the highest accuracy of 81.72% was obtained again for the 15th feature set. Thus the final accuracy of the POS tagger stands at nearly 82%.

TABLE IV: POS TAGGER RESULTS AFTER THE ADDITION OF DIGIT AND SYMBOL FEATURE

Training Data	Feature Set 15: Accuracy	Feature Set 23: Accuracy
8 K words	80.32%	79.96%
9 K words	80.76%	80.44%
10 K words	**81.72%**	80.92%

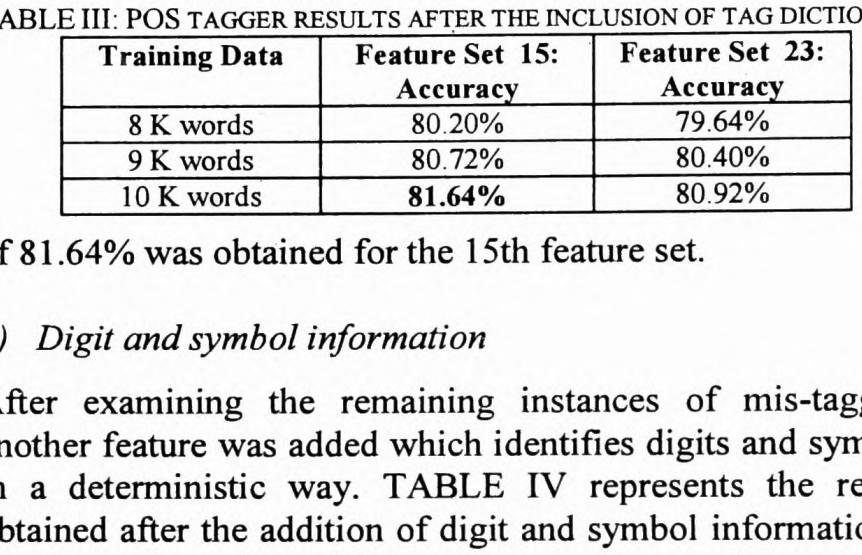

Fig. 5. Performance analysis of the POS tagger

In addition, an evaluation of the POS tagger was performed as we increase the size of the training data with the final set of features. As can be seen in Fig. 5, the accuracy of the POS tagger increases fairly evenly as the size of the training data increases.

B. Chunker

The same POS annotated corpus was used to implement the chunker for Tamil. This corpus was divided into a training set consisting of approximately 10,000 words and test set consisting of approximately 2,500 words.

Since there is no Gold Standard available for Tamil for doing the evaluation of the chunker, our own manual chunked version of the test data was used. In the evaluation, the performance of the chunker was measured in terms of IOB accuracy, recall, precision and f-measure.

- IOB Accuracy = Total no. of words which have IOB chunk tags / Total no. of words in the test data
- Recall = No. of chunks correctly identified with proper label / Total no. of correct chunks in the test data
- Precision = No. of chunks correctly identified with proper label / Total no. of chunks identified
- F-measure = The harmonic mean of precision and recall

$$F_{\alpha=0.5} = \frac{1}{(\alpha/precision + (1-\alpha)/recall)} \quad (3)$$

1) Results of the chunker

The accuracy of the chunker independent from our POS tagger results was evaluated. To do this, unchunked test data were chunked by the regular expression parser and the output was compared against the manually chunked version of the test data with the correct POS tags.

The results shown in TABLE V indicate that our complete chunker which extracts noun and verb phrases performs well with an f-measure of 78.3% compared to our noun phrase chunker.

TABLE V: RESULTS OF OUR RULE-BASED CHUNKER

	Noun Phrase	Noun Phrase + Verb Phrase
IOB Accuracy	87.4%	85.9%
Recall	72.8%	79.0%
Precision	74.5%	77.6%
F-measure	73.6%	78.3%

2) Results of the final shallow parser

When the input text to the chunker is POS annotated data obtained by executing our ME based POS tagger, the final results of the shallow parser shown in TABLE VI were obtained.

According to the results of our final shallow parser, 78.9% of the words have accurate IOB chunk tags. Though our chunker reports the results well with an f-measure of 78.3%, we obtained an overall f-measure of 66.6% for the final shallow parser - a combination of the results from our POS tagger and chunker.

TABLE VI: RESULTS OF THE FINAL SHALLOW PARSER

	Noun Phrase + Verb Phrase
IOB Accuracy	78.9%
Recall	67.4%
Precision	65.7%
F-measure	66.6%

VII. CONCLUSIONS

In this research, we attempted to propose a methodology for implementing a robust, wide coverage shallow parser for Sri

Lankan Tamil, with the combination of a machine learning based POS tagger and a rule-based chunker without the need for a large annotated corpus.

The POS tagger developed was based on a Maximum Entropy model and yielded a promising accuracy of nearly 82% for Sri Lankan Tamil. The best contextual feature we found considers the previous two words, the POS tag of the immediately previous word, and the next word of the word to be tagged, as well as its four prefix characters and six suffix characters. In addition a tag dictionary, a digit recognizer, and a symbol parser were incorporated to improve the accuracy of the tagger. The accuracy of the POS tagger can be increased further by increasing the size of the training corpus.

Our rule-based chunker independent from the results obtained by our maximum entropy based POS tagger, demonstrates a good f-measure, 78.3%. Though our chunker reports the results well with the f-measure 78.3%, we obtained an f-measure of 66.6% for the combination of the results obtained by our POS tagger and chunker, since the error from our POS tagger propagates to the chunker, which in turn affects the performance of the final shallow parser.

REFERENCES

[1] V. Dhanalakshmi, K. Anand, K. Soman, and S. Rajendran. "POS tagger and chunker for Tamil language," in *Proceedings of the 8th Tamil Internet Conference*, Cologne, Germany, 2009, pp. 123–128.

[2] A. Ratnaparkhi *et al.*, "A maximum entropy model for part-of-speech tagging," in *Proceedings of the conference on empirical methods in natural language processing*, vol. 1, 1996, pp. 133–142.

[3] A. L. Berger, V. J. D. Pietra, and S. A. D. Pietra, "A maximum entropy approach to natural language processing," *Computational linguistics*, vol. 22, no. 1, pp. 39–71, 1996.

[4] E. F. Tjong Kim Sang and S. Buchholz, "Introduction to the conll-2000 shared task: Chunking," in *Proceedings of the 2nd workshop on Learning language in logic and the 4th conference on Computational natural language learning-Volume 7*. Association for Computational Linguistics, 2000, pp. 127–132.

[5] L. Sobha, and R. V. S. Ram, "Noun Phrase Chunking in Tamil," in *Proceedings of the MSPIL-06, Indian Institute of Technology, Bombay. 2006.* pp. 194-198.

[6] K. Saravanan, P. Ranjani, and T. V. Geetha, "Syntactic Parser for Tamil," in *Proceedings of Sixth Tamil Internet 2003 Conference, Chennai, India.* 2003. pp. 28-37.

[7] D. Jurafsky, J. H. Martin, A. Kehler, K. Vander Linden, and N.Ward, *Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition*. MIT Press, 2000, vol. 2.