Lexicalized and Statistical Parsing of Natural Language Text in Tamil using Hybrid Language Models

M. SELVAM
Assistant Professor, Department of Information Technology, Kongu Engineering College, Perundurai, Erode, Tamilnadu 638052, INDIA
amm_selvam@yahoo.co.in

A.M. NATARAJAN
Chief Executive & Professor, Bannari Amman Institute of Technology, Sathyamangalam, Tamilnadu, 638401, INDIA

R. THANGARAJAN
Assistant Professor, Department of Information Technology, Kongu Engineering College, Perundurai, Erode, Tamilnadu 638052, INDIA
thangs_68@yahoo.com

Abstract:- Parsing is an important process of Natural Language Processing (NLP) and Computational Linguistics which is used to understand the syntax and semantics of a natural language (NL) sentences confined to the grammar. Parser is a computational system which processes input sentence according to the productions of the grammar, and builds one or more constituent structures which conform to the grammar. The interpretation of natural language text depends on the context also. Language models need syntax and semantic coverage for the better interpretation of natural language sentences in small and large vocabulary tasks. Though statistical parsing with trigram language models gives better performance through tri-gram probabilities and large vocabulary size, it has some disadvantages like lack of support in syntax, free ordering of words and long term relationship. Grammar based structural parsing provides solutions to some extent but it is very tedious for larger vocabulary corpus. To overcome these disadvantages, structural component is to be involved in statistical approach which results in hybrid language models like phrase and dependency structure language models. To add the structural component, balance the vocabulary size and meet the challenging features of Tamil language, Lexicalized and Statistical Parsing (LSP) is to be employed with the assistance of hybrid language models. This paper focuses on lexicalized and statistical parsing of natural language text in Tamil language with comparative analysis of phrase and dependency language models. For the development of hybrid language models, new part of speech (POS) tag set with more than 500 tags and dependency tag set with 31 tags for Tamil language have been developed which have the wider coverage. Phrase and dependency structure treebanks have been developed with 3261 Tamil sentences which cover 51026 words. Hybrid language models were developed using these treebanks, employed in LSP and evaluated against gold standards. This LSP with hybrid language models provides better results and covers all the challenging features of Tamil language.

Key-Words:- Dependency Structure, Hybrid Language Model, Lexicalized and Statistical Parsing, Natural Language Processing, Part of Speech, Treebank, Phrase Structure, Trigram Language Model, Tamil Language.

1.0 Introduction

Parsing is important in Linguistics and Natural Language Processing to understand the syntax and semantics of a natural language grammar. Parser is a computational system which processes input sentence according to the productions of the grammar, and builds one or more constituent structures called parse trees which conform to the grammar. Parsing natural language text is challenging because of the problems like ambiguity and inefficiency. A parser permits a grammar to be evaluated against a potentially large collection of test sentences, helping the linguist to identify shortcomings in their analysis.

1.1 Structural Approach

In a language, group of consecutive words act as a constituent. Context Free Grammar (CFG) which is also called phrase structure grammar has been used

to model constituents successfully. However, there are many disadvantages in using CFG for natural languages like ambiguity, left-recursion, repeated parsing of sub-trees. If a sentence is structurally ambiguous, then the grammar assigns more than one parse tree. It will be difficult to use CFG in languages that do not follow strict word order style.

1.2 Statistical Approach

Statistical methods are primarily data driven. The frequencies of patterns as they occur in any training corpora are recorded as probability distributions. These methods with *N*-gram or Trigram approaches mainly focus on short term relationship among words in sentences which depend on large training set and are suitable to model large vocabulary tasks. Whereas grammar based structural methods focus on syntax with long term relationship among words manifested in parse trees which are widely used for small vocabulary tasks. To add the structural component in statistical approach and balance the vocabulary size, LSP can be employed.

1.3 Lexicalized and Statistical Parsing and its Processes

In order to overcome the problem of ambiguity, the CFG is augmented by probabilistic component. A probabilistic context free grammar (PCFG) is a CFG in which each rule is annotated with probability of choosing that rule. PCFG probabilities can be learnt from parsing a training corpus [1]. Even though PCFG can resolve ambiguity by its probabilistic component, still PCFG is insensitive to words. Thus incorporating lexical information in PCFG has become important. The performance of PCFG can be further enhanced by conditioning a rule on the lexical head of its non-terminals [2]. This is known as Lexicalized and Statistical Parsing.

LSP has been enormously successful, but the complexity is increased. LSP is sensitive to individual lexical item and incorporation of these lexical items into features or parameters gives rise to complexity. In this paper attempts have been made to parse the Tamil language sentences by lexicalized and statistical parsing approach with the help of phrase and dependency structure language models. In this approach LSP comprises pre-processing, morphological analysis, tagging, phrasing or applying dependency relations, generation of treebank, training language model and statistical parsing. Language models are highly useful in applications like speech recognition, machine translation, etc [3][4]. A general framework of LSP with various language models is shown in Figure 1.

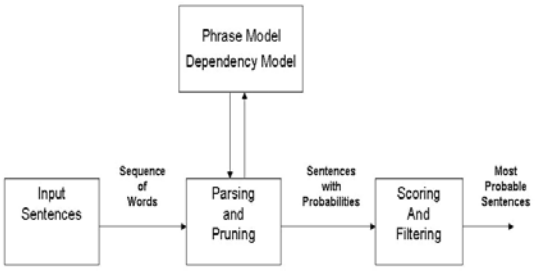

Figure 1. Framework of Lexicalized and Statistical Parser

Structural component is applied by means of phrasing or applying dependency relations after POS tagging, construction of treebank, and training language model. Language model is created with the aid of treebank and statistical parsing is done for test sentences using the language model.

1.3.1 Lexicalization

Punctuations and special characters in the sentences are removed and sentence beginning and ending markers are placed during pre-processing. POS tags are formed with morphological analysis in mind. Every word is assigned with a POS tag. In the phrase structure model, POS tag-word pair forms the leaves of the parse tree of a sentence. Phrase structure treebank is generated by grouping words into the phrases and constituents and phrases into parse trees for each and every sentence of the corpus. In the dependency model, dependency relations between tokens are marked and labeled with dependency tags. Collection of dependency annotated sentences form the dependency treebank. For building hybrid language model either phrase or dependency structure treebank can be employed.

1.3.2 Building Language Model

Language model is trained using phrase or dependency structure treebank with suitable technique which generates features and associated

probabilities among the head words. In the phrase structure model trigram approach is applied among the head words [5] of various constituent structures of a sentence which balances syntax and semantics. In the dependency model scoring is done over the edges or relations between the head words. These language models are hybrid in nature which contain probabilities among the head words which balance memory and processing time.

1.3.3 Statistical Parsing

Statistical approach is applied with the head words in both of the models with different parameters and better performance is achieved compared with simple trigram model in terms of syntax and semantics, long term relationship and free ordering of words. [6]. LSP with phrase structure language model supports long term relationship and free ordering of words to some extent only. LSP with dependency structure language model supports the same to the greater extent.

1.4 Features of Tamil Language

Grammar of Tamil language is agglutinative in nature. Suffixes are used to mark noun class, number and case. Tamil words consist of a lexical root to which one or more affixes are attached. Most of the Tamil affixes are suffixes which can be derivational or inflectional. The length and extent of agglutination is longer in Tamil resulting in long words with large number of suffixes.

In Tamil, nouns are classified into rational and irrational forms. Human comes under the rational form whereas all other nouns are classified as irrational. Rational nouns and pronouns belong to one of the three classes: masculine singular, feminine singular and rational plural. Irrational nouns belong to one of two classes: irrational singular and irrational plural. Suffixes are used to perform the functions of cases or post positions. Tamil verbs are also inflected through the use of suffixes. The suffix of the verb will indicate person, number, mood, tense and voice.

Tamil is consistently head-final language. The verb comes at the end of the clause with a typical word order of Subject Object Verb (SOV). However, Tamil language allows word order to be changed making it a relatively word order free language. Other Tamil language features are plural for honorific noun, frequent echo words, and null subject feature i.e. not all sentences have subject, verb and object.

To cater these challenging needs, LSP employs hybrid language model developed from phrase or dependency structured treebank. Phrase structured treebank is developed with POS tag set of Tamil language which needs greater coverage for all nouns, verbs, other POS and their inflections. Dependency structure treebank is developed with POS and dependency tags applied to tokens and relations between tokens respectively. Tamil language is resource deficient in all forms of treebank and associated tools. Since treebank construction is labor intensive, at least, a medium sized vocabulary treebank is to be employed to train the language model.

2.0 Language Model

Language model is the heart of the parser which provides the ways and means to predict the words and sentences confined to the patterns and grammar of a language. *N*-gram and Trigram models are the examples of statistical model and simple phrase structure model is the example of structural model.

2.1 Statistical Model

In *N*-gram language model, each word depends probabilistically on the *n-1* preceding words. This is expressed as shown in equation (1).

$$p(w_{o,n}) = \prod_{i=0}^{n-1} p(w_i \mid w_{i-n+1}, \ldots, w_{i-1}) \qquad (1)$$

When *N* is big memory and processing power requirement is high. Good results are obtained by *N=3*. This is called tri-gram language model, where each word depends probabilistically on previous two words and is shown in equation (2)

$$p(w_{o,n}) = \prod_{i=0}^{n-1} p(w_i \mid w_{i-1}, w_{i-2}) \qquad (2)$$

Trigram language model is most suitable due to the capacity, coverage and computational power. For shaping the trigram model into a greater level of suitability some advanced and optimizing techniques like smoothing, caching, skipping, clustering, sentence mixing, structuring and text normalization can be applied. Through these techniques marginal improvements in perplexity can be obtained. Even though statistical model gives better performance, proper meaning can not

be derived for the compound sentences due to the tri-gram hits which capture local dependencies.

2.2 Structural Model

Grammar based structural model is purely rule driven approach which is suitable for small vocabulary task. The grammar is applied in the form of productions and associated probabilities. Simple phrase structure model [7][8] will generate parse trees. Probabilities will disambiguate a correct parse from others. Simple structural model can overcome the disadvantages of statistical model to some extent.

2.3 Hybrid Model

Significant improvements can be achieved if structural information is applied in the statistical model [9]. In the phrase structure model trigrams are obtained among immediate heads of various constituents of the sentence. In the dependency structure model [10] probabilities are computed over edges which represent the dependency relations between the modifier and head word of the edges in a sentence.

3.0 Immediate Head Parsing

LSP with immediate head parsing technique is basically lexicalized in nature which conditions probabilities on the lexical content of the sentences being parsed. All of the properties of the immediate descendants of a constituent *c* are assigned probabilities that are conditioned on the lexical head of *c* [11][12]. For example, in Figure.2 the probability that the S expands into NP PP VP is conditioned on the head of the VP (**எடுக்காது** ['eh T uh k k aa T h uh'])[1] selected from sub-heads **பஞ்சு** ['p a eh n eh ch uh'][1] (the head of the NP), **தண்ணீரை** ['T h a N N iy r ay'][1] (the head of the PP) and **எடுக்காது** ['eh T uh k k aa T h uh'][1] (the head of VP).

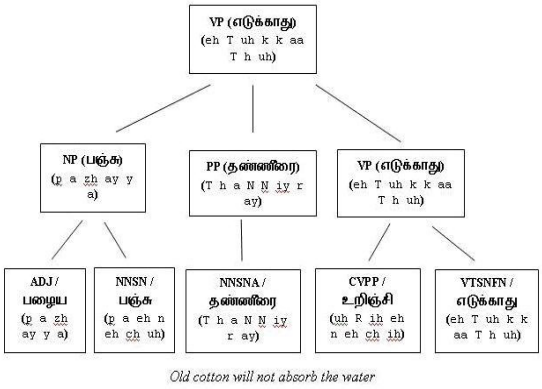

Figure 2. Parse Tree with Lexical Heads of Constituents

3.1 Calculating Parse Probabilities

This parsing model assigns a probability to a parse by a top-down process of considering each constituent *c* and predicting the pre-terminal *t(c)*, lexical head *h(c)* and expansion *e(c)* for each *c*. The probability of a parse is given by the equation (3)

$$p(\pi)=\prod_{c\in\pi} p(t(c)\mid l(c),H(c)).p(h(c)\mid t(c),l(c),H(c)).\ p(e(c)\mid l(c),t(c),h(c),H(c)) \quad (3)$$

where *l(c)* is the label of *c* (whether it is a noun phrase (NP), verb phrase(VP), etc.) and *H(c)* is the relevant history of *c*. *H(c)* consists of the label, head and head-part-of-speech for the parent of *c*: *m(c)*, *i(c)*, and *u(c)* respectively. One exception is the *e(c)* distribution, where *H* only includes *m* and *u*. Equation (3) is written as shown in equation (4)

$$p(\pi)=\prod_{c\in\pi} p(t\mid l,m,u,i).p(h\mid t,l,m,u,i).p(e\mid l,t,h,m,u) \quad (4)$$

A bonus multiplicative factor for constituents that end at the right boundary of the sentence and a penalty for the constituents which do not end at right boundary [5] are given.

3.2 Finding Best Parse among N Parses

LSP is generative in which parser tries to find the parse of a sentence *s* defined by

$$\arg\max_{\pi} p(\pi\mid s)=\arg\max_{\pi} p(\pi,s) \quad (5)$$

Language model *p(s)* is defined by assigning a probability to all possible sentences in the language by computing the sum

$$p(s) = \sum_{\pi} p(\pi, s) \quad (6)$$

4.0 Development of Phrase Structure Language Model

Phrase structure language Model is the combination of structural and statistical model. After applying POS tag for each and every lexicon in the bottom level, structural component is added by means of phrasing the constituents in the sentences.

4.1 Pos Tag-Set

Parts of Speech in Tamil language take different forms and inflections as shown in Table 1.

Table 1: POS Forms and Morphological Inflections

POS & Others	Forms	Morphological inflections
Noun	Simple Noun Proper Noun Participle Noun Adjective Noun Positive Tensed Verbal Noun Negative Tensed Verbal Noun Un-tensed Verbal Noun	Number ▪ Singular ▪ Plural Gender ▪ Male ▪ Female ▪ Neutral ▪ Common ▪ Oblique
Verb	Simple Verb Transitive Verb Intransitive Verb Causative Verb Infinitive Verb Imperative Verb Reportive Verb	Person ▪ First ▪ Second ▪ Third Number ▪ Singular ▪ Plural Gender ▪ Male ▪ Female ▪ Neutral ▪ Common Tense ▪ Present ▪ Past ▪ Future Passive Honorific Negative Interrogative Suffix
Adverb	Simple Adverb	
Adjective	Simple Adjective Participle Adjective	Tense ▪ Present ▪ Past ▪ Future Negative
Preposition	Simple preposition Noun+ cases	Cases ▪ Accusative ▪ Dative ▪ Instrumental ▪ Sociative ▪ Locative ▪ Ablative ▪ Benefacive ▪ Genetive ▪ Vocative ▪ Clitics ▪ Selective Negative
Conjunction	Simple Conjunction Coordinating conjunction	Wh words ▪ What ▪ Who ▪ Whose ▪ When ▪ Where ▪ Whom ▪ Which ▪ How
	Participle	Verbal Participle Conditional participle ▪ Positive ▪ Negative
Interjection	Simple Interjection	
Others	Echo words	Same Different
	Determiner Quantifier Complementizer Ordinal Optative	

Morphological inflections on nouns include gender and number. Prepositions take either independent form or noun combined form with various cases like accusative, dative, instrumental, sociative, locative, ablative, benefactive, genitive, vocative, clitics and selective [13][14][15]. Some other forms of verbs are transitive, intransitive, causative, infinitive, imperative and reportive. Adjective tags are generated along with tense and positive or

negative participles. Other parts of speech take simpler forms.

4.2 POS Tagging and Phrasing

Every sentence in the corpus is segmented into sequence of tokens and each and every token is applied with the appropriate tag for the application of direct meaning to the words. The tags and lexicons are bracketed for all the pairs. Phrasing is done among the words to form syntactic phrases and constituents. Phrases are classified based on the parts of speech as shown in Table 3.

4.3 Phrase Structure Treebank

Phrase structure treebank is a corpus with linguistic annotation beyond the word level. The annotation is typically a syntax tree which is manually checked and corrected [16] as shown in Figure. 3.

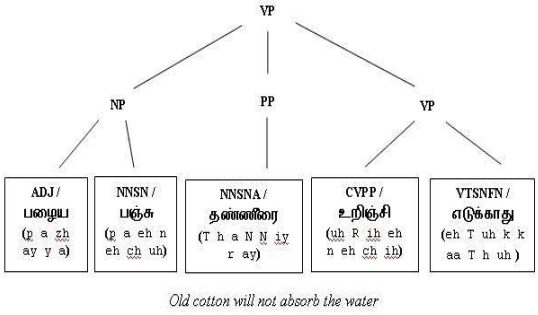

Figure 3. Syntax Tree

Treebank provides training material for Machine Learning in NLP systems [17]. It is used to build gold standards for the evaluation of NLP systems. It supports in the experimentation of linguistics against other linguistic theories. It provides material for human grammar exploration and learning.

Simple bracketed version of treebank is generated by phrasing all the sentences in the text corpus. This is basically constituency based format. This is done using automated tools like morphological analyzer, POS tagger and phrasing tool with manual corrections. All the annotated sentences will take form[1] as shown below.

(S1 (S (NP (ADJ பழைய)
(NNSN பஞ்சு))
(PP (NNSNA தண்ணீரை))
(VP (CVPP உறிஞ்சி)
(VTSNFN எடுக்காது))))

Transliterated Equivalent sentence

(S1 (S (NP (ADJ 'p a zh ay y a')
(NNSN 'p a eh n eh ch uh'))
(PP (NNSNA 'T h a N N iy r ay'))
(VP (CVPP 'uh R ih eh n eh ch ih')
(VTSNFN 'eh T uh k k aa T h uh'))))

4.4 Phrase Structure Language Model

Phrase structure language model is trained using phrase structured treebank. By means of immediate head parsing technique heads are selected from various constituents and trigram approach is applied among the heads. For all the parameters of constituent *c* discussed in Section 3.1 feature files are created and updated during the training process. All the feature files together constitute this hybrid language model.

5.0 Dependency Parsing

Dependency representations are more efficient and very simple than phrase structures. It has the additional advantage of encoding the information about predicate arguments. It is suitable for the applications like relation extraction, machine translation, etc. Thus, dependency parsing uses a syntactic representation whose computational complexity will allow exploring discriminative training, while at the same time providing a usable representation of language for many natural language processing tasks. Dependency structure for a sentence is a directed graph originating out of a unique and artificially inserted root node. Dependency graph is a weakly connected directed graph where each word has exactly one outgoing edge except the root which has no outgoing edge. There is no cycle i.e if there are *n* words in the sentence including root then the graph has exactly $n-1$ edges. Dependency graphs which satisfy the tree constraints are called dependency trees.

5.1 Issues in Dependency Relations

When constructing a dependency structure there are many issues to address. Definition of the head and modifier in a relation is most important. Some classes of relations are relatively easy to define. For

instance, both subjects and objects are modifying a verb or sets of verbs. Similarly, adjectives and adverbs play the role of modifier. However, in Tamil language the preposition are attached to nouns and complementizer governs the verb.

5.2 Dependency Relationship in Tamil Language

Whenever two words are connected by a dependency relation, we say that one of them is the head and the other is the dependent, and that there is a link connecting them. In general, the dependent is in the form of modifier, object or complement. The head plays the larger role in determining the behavior of the pair. In our dependency representation the source of the edge represents the modifier and destination points to the head word. The dependency structure of a sample Tamil sentence[1] is shown in Figure 4.

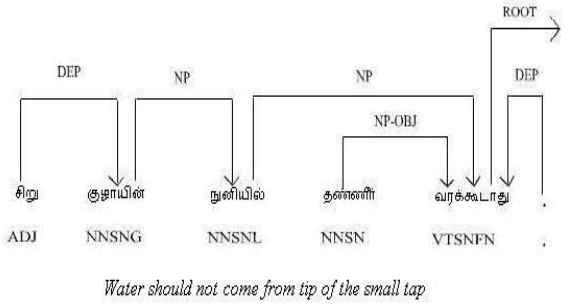

Figure 4: Dependency Tree for a sample Tamil sentence

Here the each word is annotated with POS tag to know the lexical information of the sentence. And the each word's dependent relation is also annotated. Here the word **சிறு** (ch ih R uh)[1] depends on **குழாயின்** (k uh zh aa y ih n)[1] as DEP (simply dependent), **குழாயின்** (k uh zh aa y ih n)[1] depends on **நுனியில்** (nn uh n ih y ih l)[1] as NP (Noun Phrase), **நுனியில்** (nn uh n ih y ih l)[1] depends on **தண்ணீர்** (T h a N N iy r)[1] as NP, **தண்ணீர்** (T h a N N iy r)[1] depends on **வரக்கூடாது** (v a r a k k uw T aa T h uh)[1] as NP-OBJ (Noun Phrase Object) and finally **வரக்கூடாது** (v a r a k k uw T aa T h uh)[1] is the root word.

5.3 Maximum Spanning Tree

Suppose $x = x_1, \ldots, x_n$ is an input sentence, and y is a dependency tree for sentence x. Taking y as the set of tree edges, $(i, j) \in y$ if there is a dependency in y from word x_i to word x_j . The score of a dependency tree is calculated as the sum of the scores of all edges in the tree. The score of an edge is the dot product between a high dimensional feature representation of the edge and a weight vector is shown in equation 3.

$$s(i, j) = \mathbf{w}.\mathbf{f}(i, j) \tag{3}$$

The score of a dependency tree y for a sentence is given in equation 4.

$$s(x, y) = \sum_{(i,j)\in y} s(i, j) = \sum_{(i,j)\in y} \mathbf{w}.\mathbf{f}(i, j) \tag{4}$$

Assuming an appropriate feature representation as well as a weight vector **w**, dependency parsing is the task of finding the dependency tree y with highest score for a given sentence x.

A directed graph is represented as $G = (V, E)$ by its vertex set $V = \{v_1, \ldots, v_n\}$ and set $E \subseteq [1:n] \times [1:n]$ of pairs *(i, j)* of directed edges $v_i \rightarrow v_j$. Each edge has a score *s(i, j)* and does not necessarily equal *s(j, i).* An example[1] of dependency graph is shown in Figure 5(a). A Maximum Spanning Tree (MST) of G is a tree $y \subseteq E$ that maximizes the value $\sum_{(i,j)\in y} s(i, j)$ such that every vertex in v appears in y. The maximum projective spanning tree of G is constructed only with projective edges relative to some total order on the vertices of G. For each sentence x, a directed graph is defined as $G_x = (V_x, E_x)$ where

$$V_x = \{x_0 = root, x_1, \ldots, x_n\}$$
$$E_x = \{(i, j) : i \neq j, (i, j) \in [0:n] \times [1:n]\}$$

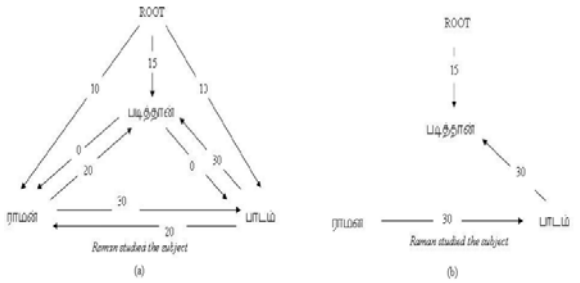

Figure 5: Dependency Graph and its Maximum Spanning Tree

G_x is a graph with the words of a sentence and the dummy *root* symbol as vertices and a directed edge between every pair of distinct words and from the root symbol to every word. Dependency trees for x and spanning trees for G_x coincide, since both kinds of trees are required to be rooted at the dummy root and reach all the words in the sentence. Finding a (projective) dependency tree with highest score is equivalent to finding a maximum (projective) spanning tree in G_x. The example[1] of maximum spanning tree is shown in Figure 5(b).

It is shown that treating dependency parsing as the search for the highest scoring maximum spanning tree in a graph gives efficient algorithms for both projective and non-projective trees [18]. Since dependency tree represents the relations between words, the long term relation is easily obtained. It gives higher efficiency and there is no need of high volume training set.

5.4 Projective Dependency Parsing

A dependency tree is projective when the words are in linear order, preceded by the root and the edges can be drawn above the words without crossings or equivalently, a word and its descendants form a contiguous substring of the sentence. Figure 4 is an example of projective dependency tree. In English, projective trees are sufficient to analyze most of the types of sentences.

5.5 Non-Projective Dependency Parsing

For free-word order languages like Tamil, non-projectivity is a common phenomenon since the relative positional constraints on dependents is much less rigid. Rich inflectional morphological language like Tamil reduces reliance on word order to express grammatical relations and allows non-projective dependencies that need to be represented and parsed efficiently. An example[1] of non-projective dependency graph is shown in Figure 6.

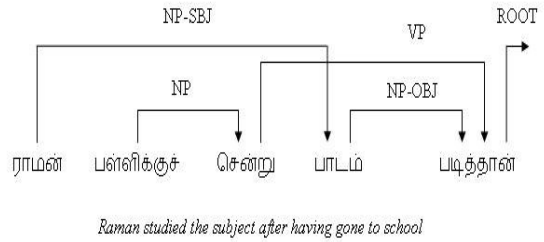

Figure 6: A Non-Projective Dependency Graph

6.0 Development of Dependency Language Model

Development of dependency language model consists of generation of POS and dependency tags, POS tagging, marking and labeling of dependency relations, generation of treebank and training language model using treebank.

6.1 Generation of Pos Tag Set

Generation of POS tags with the analysis of morphological inflections has been discussed in Section 4.1

6.2 Pos Tagging

In the training of dependency language model MST format is used for all input sentences. POS tags corresponding to the words are listed in the sequence in the second line of the format. This[1] is shown in Figure 7.

WORD	→	காட்சி	நடத்துகிறவா	மோதிரத்தை	வாங்க	வேண்டும்	.
POS-TAG	→	NNSN	NNSN	NNSNA	OPT	VTSNF	.
LABEL	→	NP	NP-SBJ	NP-OBJ	VP	ROOT	DEP
DEPENDENCY	→	2	4	4	5	0	5

Performer of the show should get a ring

Figure 7: Sentence with POS tags, Dependency Relations and Labels

6.3 Generation of Dependency Tag Set

For applying the dependency relations, tags are needed in the clause and phrase levels [16]. Tags are used for various clauses like declarative, inverted declarative, direct and indirect questions,

subordinating conjunction and inverted yes/no or *wh*-questions. Also for phrases like noun, verb, adverb, adjective, conjunction, interjection, preposition and *wh*-phrases tags are needed. Some more tags are needed for subject, object, dependent word, root and unknown words. This tag set covers all the language constructs. Limited tag set is sufficient for this dependency analysis. More emphasis behind the words can be given by POS tagging. Sample dependency tag set is shown in Table 4.

6.4 Marking of Dependency Relations

Dependency relations between the words are analyzed and marked by means of word index. Root word is assigned with an index 0. The period (.) acts as a dummy root and assigned with index of the root word. Other words are assigned with index of their respective head words in the fourth line of input sentence in MST format. This is shown in Figure 7.

6.5 Dependency Labeling

Dependency relationship is applied using edge factoring. Through this, unlabeled accuracy can be obtained in the first stage of parsing. For further processing and obtaining grammatical relations among the words, labeling is applied to the relations in the third line of input sentence in MST format as shown in Figure 7. This enables to obtain labeled accuracy in the second stage of parsing.

6.6 Generation of Dependency Treebank

By applying POS tags and marking and labeling of dependency relations, sentences are annotated. The collection of all the annotated statements forms the dependency tree bank. In the proposed work treebank has been created using POS tagger and bootstrapping with manual corrections. Size of the treebank can be increased by means by bootstrapping and employing automated tools which reduce laborious work and time.

6.7 Training the Dependency Language Model

Dependency language model has been trained using this dependency treebank. Input sentence is a tab delimited text file as shown below.

w_1	w_2	...	w_n
p_1	p_2	...	p_n
l_1	l_2	...	l_n
d_1	d_2	...	d_2

Where,

w_i	is the i^{th} word of the sentence
p_i	is the POS tag of i^{th} word
l_i	is the label of the outgoing edge to i^{th} word
d_i	is the integer representing the position of i^{th} word's head

The task of this training algorithm is to calculate the score of each edge in the sentence. This score indicates how likely one word of the edge is a dependent of the other word in this edge. In other words, for each pair of words weight vector is calculated. In each iteration, different scores for the pair of words are generated. The parser is discriminatively trained in which the corpus is reparsed and during each reparsing, those features are defined which allow the model to make decisions better. Discriminatively trained parser scores entire trees rather than making separate parsing decisions unlike generative models

7.0 Experiments

Phrase and dependency hybrid language models have been built and employed in lexicalized and statistical parsing of the Tamil language sentences. In this process POS, phrase and dependency tag sets were generated and phrase and dependency structure treebank were developed

7.1 Proposed POS Tag Set for Tamil Language

Based on rich morphological inflections and POS forms of Tamil language, more than 500 tags have been created. In Tamil Language nouns and verbs take more forms than other languages as suggested in the Table 1. Preposition takes direct and noun combined forms. Adjective takes direct and verb combined forms. For interrogative statements wh-tags were generated. This POS tag set has wider coverage to all Tamil language words. Some of the examples of tags are given in Table 2.

Table 2: Sample POS Tags for Tamil Language

Tag	Description	Example in Tamil with ARPABET Transliteration and meaning
ADJ	Adjective	அழகிய (a zh a k ih y a) (beautiful)
ADJAP	Adjective Past participle	செய்த (ch eh y T h a) (done)
ADV	Adverb	வேகமாக (v ee k a m aa k a) (quickly)
CON	Conjunction	அல்லது (a l l a T h uh) (or)
CVCN	Verbal Conditional negative	செய்யாவிட்டால் (ch eh y y aa v ih T T aa l) (if not done)
DET	Determiner	இந்த (ih nn T h a) (this)
INT	Interjection	ஐயோ (ay y oo) (Alas)
NAPC	Adjective Noun plural common	நல்லவர்கள் (nn a l l a v a r k a L) (good people)
ORD	Ordinal	மூன்றாவது (m uw n R aa v a T h uh) (Third)
PRP	Preposition	உள்ளே (uh L L ee) (inside)
QNT	Quantifier	சில (ch ih l a) (few)
V	Verb	படி (p a T ih) (study)
VC	Verb Causative	கற்பி (k a R p ih) (teach)
VFPA	First Person Plural Past Tense Verb	சென்றோம் (ch eh n R oh m) (we went)
VI	Intransitive verb	திரும்பு (T h ih r uh m p uh) (turn)
VIF	Infinitive Verb	செய்ய (ch eh y y a) (to do)
VSPAN	Second Person Plural Past Tense Negative Verb	செய்யவில்லை (ch eh y y a v ih l l ay y ay) (did not do)
VT	Transitive Verb	திருப்பு (T h ih r uh p p uh) (turn – any object)
VTSNFN	Third Person Singular Neutral Future Tense Negative Verb	எடுக்காது (eh T uh k k aa T h uh) (will not take)

7.2 Proposed Phrase Structures

For applying the syntactic phrases for the sentences, the following phrase tags were suggested. The proposed phrase tag set covers all the constituent structures of Tamil language sentences. This is shown in Table 3.

Table 3: Proposed Phrases

Phrases	Descriptions
NP	Noun Phrase
VP	Verb Phrase
ADVP	Adverbial Phrase
ADJP	Adjective Phrase
PP	Prepositional Phrase
CP	Conjunctional Phrase
IP	Interjectional Phrase
WHNP	Conjunctional Noun Phrase
WHVP	Conjunctional Verb Phrase
WHPP	Conjunctional Prepositional Phrase

7.3 Proposed Dependency Tag Set for Tamil Language

The dependency tag set used to label the dependency relations is shown in Table 4. This tag set has wider coverage for all Tamil language constructs.

Table 4: Sample Dependency Tags used for Tamil Language

Tag	Description
ADJP	Adjective Phrase
ADVP	Adverb Phrase
CONJP	Conjunction Phrase
DEP	Dependent Word
FRAG	Fragment
INTJ	Interjection
NP	Noun Phrase
NP-OBJ	Object as NP
NP-SBJ	Subject as NP
PP	Prepositional Phrase
QP	Quantifier Phrase
ROOT	Root Word
S	Simple declarative clause
SBAR	Subordinating conjunction Clause
SINV	Inverted declarative sentence
VP	Verb Phrase
WHAVP	*Wh*-adverb Phrase
WHNP	*Wh*-noun Phrase
WHPP	*Wh*-prepositional Phrase
X	Unknown, Uncertain

7.4 Generation of Phrase and Dependency Structure Treebanks

Phrase structure treebank has been developed for 3261 sentences which has the size of 51026 words by using our own rule based morphological

analyzer, POS tagger and phrasing tool. Sentences were annotated automatically by the tools with manual corrections. Dependency treebank has been created for the same sentences using POS tagger and bootstrapping with manual corrections. In the dependency tag set 31 tags have been used.

7.5 Training Phrase and Dependency Language Models

Phrase structure language model has been trained using phrase structure treebank which comprises feature files generated for the features quoted in equation (4). The probability values of all the features are initialized and updated during the training process. These values are used later in the parsing process. By considering (CVPP / உறிஞ்சி ['uh R ih eh n eh ch ih'][1]) as constituent *c* in Figure.2, examples of the features are shown in the Table 5.

Table 5: Features in phrase structure language model and their examples

Features	Description	Example
t	Tag of constituent	CVPP
l	Label of the constituent	VP
h	Head of the constituent	உறிஞ்சி (uh R ih eh n eh ch ih)[1]
e	Expansion of constituent	---
M	Label of the parent	VP
i	Head of the parent	எடுக்காது (eh T uh k k aa T h uh)[1]
u	Head-part-of-speech for the parent	VTSNFN

By using dependency treebank, dependency language model has been trained. Features updated during training include directions of attachment, the distance between the words and contextual features. Contextual features are POS tags of words that occur in between parent and child nodes and POS tags of words that surround parent and child nodes to the right and left. Adding contextual features leads to the considerable improvement of the performance of the dependency language model.

8.0 Results and Discussion

Parsing has been done for two set of sentences from trained and test sets with phrase structure language model. Results are evaluated with their respective gold standards. Result of phrase structure language model is shown in Table 6.

Table 6. Results from Phrase Structure Language Model

Details	Trained Sentences	Test Sentences
Total Sentences	600	300
Ref. Words	6126	2728
Hyp. Words	5758	2537
Total Word Accuracy	94 % (5758 / 6126)	93% (2537/2728)
Correct Sentences	438	195
Sentence Accuracy	73% (438/600)	65% (195/300)

With the same training and test sentences parsing has been done with dependency structure language Model. Test sentences use the same MST format as shown in Figure 7. Third and fourth lines should be filled with filler words LAB and 0 respectively for all tokens. This[1] is shown in Figure 8.

இதைக்	கண்ட	சபையோர்	ஆச்சரியப்படுவார்கள்	.
NOSNA	CVPP	NNPN	VTPCF	.
LAB	LAB	LAB	LAB	LAB
0	0	0	0	0

After having seen this, audience will be surprised

Figure 8: Input Sentence format of Test Case

Output is generated in the same MST format which has been used for the training. The sample output[1] is shown in the Figure 9. Parser substitutes the third and fourth line with dependency labels and relations respectively for the test sentences.

இதைக்	கண்ட	சபையோர்	ஆச்சரியப்படுவார்கள்	.
NOSNA	CVPP	NNPN	VTPCF	.
NP	VP	NP-SBJ	ROOT	DEP
2	3	4	0	4

After having seen this, audience will be surprised

Figure 9: Sample output sentence

Results are evaluated with their respective gold standards. Results are shown with labeled and unlabelled accuracies for tokens and sentences in Table 7.

Table 7. Results from Dependency Structure Language Model

Details	Trained Sentences	Test Sentences
Total Sentences	600	300
Total Tokens	6126	2728
Correct Tokens	6060	2547
Unlabeled Token Accuracy	98.92 %	93.37 %
Unlabeled Sentence Accuracy	93.50 %	76.50 %
Labeled Token Accuracy	98.68 %	89.67 %
Labeled Sentence Accuracy	91.83 %	69.50 %

9.0 Conclusion and Future Work

POS and dependency tag sets have been created with more than 500 and 31 tags respectively. 3261 sentences are used for phrase and dependency structure treebanks which have 51026 vocabularies. Phrase and dependency structure language models have been built and 600 trained and 300 test sentences were parsed and evaluated against gold standards. Since Tamil language is the relatively word order free language, LSP with these hybrid language models gives better results and performs well for the application of syntax with semantics, long term relationship and free word order. LSP with phrase structure language model covers the above said features to some extent. LSP with dependency structure language model covers same features to the greater extent. These hybrid language models are very useful for the applications like Speech recognition, Machine translation, Optical character recognition, etc.

Performance can be increased further by providing more and more training to the language models by increasing the size of the treebank in future. Since Tamil language is resource deficient, developing treebanks with greater size is laborious and time consuming even with bootstrapping. By employing the induction technique [19] in which English POS tags are directly projected to Tamil Sentences via word alignment with morphological analysis in English and Tamil languages parallel corpora, very large Tamil Treebank can be developed. Also Cross Lingual Latent Semantic Analysis [20] can be employed with document aligned English and Tamil languages Parallel corpora for generating the Treebank since sentence aligned parallel corpora is also scarcely available. Using these large treebanks accurate hybrid language models can be developed.

Acknowledgement

Language model part of this research is the sponsored work of the project funded by Tamil Virtual University, Chennai, India, under the scheme of Tamil Software Development Funding (TSDF).

The authors would like to thank Central Institute of Indian Languages (CIIL), Mysore, India and Department of Science and Technology, New Delhi, India for providing the Tamil text corpora.

Endnote

1. Transliterated equivalent of Tamil sentences and words used in figures and examples in this manuscript in ARPABET format

Sentence-1	பழைய	பஞ்சு	தண்ணீரை	உறிஞ்சி	எடுக்காது
Transcription-1	p a zh ay y a	p a eh n eh ch uh	Th a N N iy r ay	uh R ih eh n eh ch ih	eh T uh k k a T h uh
Sentence-2	சிறு	குழாயின்	நுனியில்	தண்ணீ	வரக்கூடாது
Transcription-2	ch ih R uh	k uh zh aa y ih n	nn uh n ih y ih l	Th a N N iy r	v a r a k k uw T aa Th uh
Sentence-3	காட்சி	நடத்துகிறவா	மோதிரத்தை	வாங்க	வேண்டும்
Transcription-3	k aa T ch ih	nn a T a Th Th uh k ih R a v a r	m oo Th ih r a Th Th ay	v aa n y k a	v ee N T uh m
Sentence-4	ராமன்	பள்ளிக்குச்	சென்று	பாடம்	படித்தான்
Transcription-4	r aa m a n	p a L L ih k k uh ch	ch eh n R uh	p aa T a m	p a T ih Th Th aa n
Sentence-5	ராமன்	பாடம்	படித்தான்		
Transcription-5	r aa m a n	p aa T a m	p a T ih Th Th aa n		
Sentence-6	இதைக்	கண்ட	சபையோர்	ஆச்சரியப்படுவார்கள்	
Transcription-6	ih Th ay k	k a N T a	ch a p ay y oo r	aa ch ch a r ih y a p p a T uh v aa r k a L	

References:

[1] Chi, Z. and Geman, S, Estimation of Probabilistic Context-Free Grammars. Computational Linguistics 24 2, 1998, 299–306.

[2] Roark B., Probabilistic Top–Down Parsing and Language Modeling, Association for Computational Linguist, 2001.

[3] Daniel M. Bikel, On the Parameter Space of Generative Lexicalized Statistical Parsing Models, Ph.D. Thesis, University Of Pennsylvania, 2004

[4] Daniel Jurafsky & James H. Martin, Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition, 2nd Edition, Pearson Education, 2006

[5] Eugene Charniak, Immediate-Head Parsing for Language Models, Proceeding of ACL, 2001

[6] Collins, M. J. Head-Driven Statistical Models for Natural Language Parsing. University of Pennsylvania, Ph.D. Dissertation, 1999

[7] Chelba, C. and Jelinek, F. Exploiting Syntactic Structure for Language Modeling. In Proceedings for COLING-ACL 98. ACL, Newbrunswick NJ, 1998, 225–231.

[8] Chelba, C. and Jelinek, F. Structured Language Modeling. Computer Speech and Language 14, 2000, 283–332.

[9] Diego Linares Pontificia and Jos E-Miguel Benedi and Joan-Andreu Sanchez, A Hybrid Language Model based on a Combination of N-Grams and Stochastic Context-Free Grammars , ACM Transactions on Asian Language Information Processing, Volume 3, Issue 2, 2004, pp.113-127.

[10] Ciprian Chelba and David Engle, Structure and Performance of a Dependency Language Model, 2000

[11] Ratnaparkhi, A. Learning to parse Natural Language with Maximum Entropy Models. Machine Learning 34 1/2/3, 1999, 151–176.

[12] Charniak, E. A Maximum-Entropy Inspired Parser. In Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics. ACL, New Brunswick NJ, 2000

[13] Bharati, Akshar, Vineet Chaitanya and Rajeev Sangal, Natural Language Processing: A Paninian Perspective, Prentice-Hall of India, New Delhi, 1995

[14] Akshar Bharati, Rajeev Sangal, Vineet Chaitanya, Anncorra : Building Tree-Banks in Indian Languages, COLING 2002 Post Conference Workshops - Proceedings of the 3rd Workshop on Asia Language Resources and International Standardization at Taipei, Taiwan, 2002

[15] Rajendran S, Strategies In The Formation Of Compound Nouns In Tamil, Languages Of India, Volume 4, 2004

[16] Marcus, M. P., Santorini, B. and Marcinkiewicz, M. A. Building A Large Annotated Corpus of English: The Penn Treebank. Computational Linguistics 19, 1993, 313–330.

[17] Charniak, E., Tree-Bank Grammars. In Proceedings of the Thirteenth National Conference on Artificial Intelligence. AAAI Press/MIT Press, Menlo Park, 1996, 1031–1036.

[18] Ryan McDonald and Fernando Pereira, Non-projective Dependency Parsing using Spanning Tree Algorithms, 2001

[19] D. Yarowsky, G. Ngai, and R. Wicentowski. Inducing multilingual text analysis tools via robust projection across aligned corpora. In Proc. HLT, Santa Monica, CA, 2001, pages 109–116.

[20] W. Kim and S.Khudanpur, Cross-Lingual Latent Semantic Analysis for Language modeling, IEEE, 2004, pp.257 - 260.