International Journal of Computer Applications (0975 – 8887)
Volume 7– No.8, October 2010

Penn Treebank-Based Syntactic Parsers for South Dravidian Languages using a Machine Learning Approach

Antony P J
Research Scholar
CEN Department, Amrita Vishwa Vidyapeetham University, Coimbatore, Tamil Nadu, India

Nandini. J. Warrier
Assistant Professor
CSE Department, Amrita Vishwa Vidyapeetham University, Coimbatore, Tamil Nadu, India

Dr. Soman K P
Professor and Head
CEN Department, Amrita Vishwa Vidyapeetham University, Coimbatore, Tamil Nadu, India

ABSTRACT

With the availability of limited electronic resources, development of a syntactic parser for all types of sentence forms is a challenging and demanding task for any natural language. This paper presents the development of Penn Treebank based statistical syntactic parsers for two South Dravidian languages namely Kannada and Malayalam. Syntactic parsing is the task of recognizing a sentence and assigning a syntactic structure to it. A syntactic parser is an essential tool used for various natural language processing (NLP) applications and natural language understanding. The well known grammar formalism called Penn Treebank structure was used to create the corpus for proposed statistical syntactic parsers. Both the parsing systems were trained using Treebank based corpus consists of 1,000 Kannada and Malayalam sentences that were carefully constructed. The developed corpus has been already annotated with correct segmentation and Part-Of-Speech (POS) information. We have used our own POS tagger generator for assigning proper tags to each and every word in the training and test sentences. The proposed syntactic parser was implemented using supervised machine learning and probabilistic context free grammars (PCFG) approaches. Training, testing and evaluations were done by support vector method (SVM) algorithms. From the experiment we found that the performance of our systems are significantly well and achieves a very competitive accuracy.

Keywords

Penn Treebank, Dravidian Languages, Syntactic Parser, Part-Of-Speech, Support Vector Methods

1. INTRODUCTION

Syntactic parsing of sentences is considered to be an important intermediate stage for semantic analysis in natural language processing (NLP) application such as information retrieval (IR), information extraction (IE) and question answering (QA). The study of structure of sentence is called syntax and it attempts to describe the grammatical order in a particular language in term of rules which detail an underlying structure and a transformational process. Syntax provides rules to put together words to form components of sentence and to put together these components to form meaningful sentences [1]. In natural language processing, syntactic parsing or more formally syntactic analysis is the process of analyze and determine the structure of a text which is made up of sequence of tokens with respect to a given formal grammar. Because of the substantial ambiguity present in the human language, whose usage is to convey different semantics, it is much difficult to design the features for natural language processing tasks. The main challenge is the inherent complexity of linguistic phenomena that makes difficult to represent the effective features for the target learning models.

Literature shows that the rule based grammar refinement process is extremely time consuming and difficult and failed to analyze accurately a large corpus of unrestricted text. Hence, most modern parsers are based on statistical or at least partly statistical, which allows the system to gather information about the frequency with which various constructions occur in specific contexts. Any statistical approach requires the availability of aligned corpora which are: large, good-quality and representative.

Probabilistic context free grammars (PCFG), maximum entropy techniques, and neural networks based learning are some of the approaches that have proven their efficiency for parsing and other natural language processing. The researchers have to agree on the grammar that is to be used for analyzing the syntactic structure in natural language parsing. For some parsing systems use lexical functional grammar called NP-complete grammar. Head-driven phrase structure grammar is another linguistic formalism which has been popular in the parsing community. A less complex grammar formalism called Penn Treebank structure is much popular in the field of natural language syntactic analysis.

Penn Treebank corpora have proved their value both in linguistics and language technology all over the world. At present a lot of research has been done in the field of Treebank based probabilistic parsing successfully. The main advantage of Treebank based probabilistic parsing is its ability to handle the extreme ambiguity produced by context-free natural language grammars. Information obtained from the Penn Treebank corpora has challenged the intuitive language study for various natural language processing purposes [2]. South Dravidian languages are morphologically rich in which a single word may carry different sorts of information. The different morphs composing a word may stand for, or indicate a relation to other elements in the syntactic parse tree. There for it is a challenging task to the developers in terms of the status of the orthographic words in the syntactic parse trees.

In this paper we propose a Penn Treebank based probabilistic syntactic parsers for two South Dravidian languages namely Kannada and Malayalam. The proposed supervised machine learning systems are implemented using support vector algorithms. Parts of speech tagging is an important stage in the syntactic parsing model and we have used our own POS tagger models [3][4] for assigning tags to each and every words in the sentence.

2. LITERATURE SURVEY

A series of statistical based parsers for English are developed by various researchers namely: Charniak-1997, Collins-2003, Bod et al. - 2003 and Charniak and Johnson- 2005 [5][6]. All these parsers are trained and tested on the standard benchmark corpora called Wall Street Journal (WSJ). A probability model for a lexicalised Probabilistic Context Free Grammar (PCFG) was developed by Charniak in1997. In the same time Collins also describes three generative parsing models, where each model is a refinement on the previous one, and achieving improved performance. In 1999 Charniak introduced a much better parser called maximum-entropy parsing approach. This parsing model is based on a probabilistic generative model and uses a maximum-entropy inspired technique for conditioning and smoothing purposes. In the same period Collins also present a statistical parser for Czech using the Prague Dependency Treebank. The first statistical parsing model based on a Chinese Treebank was developed in 2000 by Bikel and Chiang. A probabilistic Treebank based parser for German was developed by Dubey and Keller in 2003 using a syntactically annotated corpus called 'Negra'. The latest addition to the list of available Treebank is the 'French Le Monde' corpus and it was made available for research purposes in May 2004. Ayesha Binte Mosaddeque & Nafid Haque wrote Context Free Grammar (CFG) for 12 Bangla sentences that have taken from a newspaper [7]. They used a recursive descent parser for parsing the CFG.

Comparing with foreign languages, a very little work has done in the area of natural language processing for Indian languages. B.M. Sagar, Shobha G and Ramakanth Kumar P. worked on Solving the Noun Phrase and Verb Phrase Agreement in Kannada Sentences [8]. Bala sundara Raman L, Ishwar S, Sanjeeth Kumar Ravindranath, implemented Natural Language constructs for 'Venpa' class of Tamil Poetry using Context Free Grammar [9]. G.V. Singh and D.K. Lobiyal, attempted to check the grammar for Hindi sentences with compound, conjunct or complex verb phrases [10]. Selvam M, Natarajan. A M, and Thangarajan R implemented a structural parsing of Tamil using phrase structure hybrid language model for almost 700 sentences [11]. AUKBC-NLP team under the supervision of Professor Rajendran S prepared a morphological parser and a shallow parser for Tamil [12]. To the best of our knowledge apart from these there is not much research literature available for development in computational processing of Indian languages at present.

3. SOUTH DRAVIDIAN LANGUAGES

Among the four major South Dravidian languages such as Kannada, Malayalam, Tamil and Telugu are having almost 40, 35, 70 and 71 million speakers respectively [13]. These languages have their own independent scripts and long documented histories. Verbs have a negative as well as an affirmative voice. Gender classification is made on the basis of rank instead of sex, with one class including beings of a higher status and the other beings of an inferior status. Nouns are declined, showing case and number. In South Dravidian languages a great use is made of suffixes with nouns and verbs. Also all these four Dravidian languages have their own alphabets, related to the Devanagari alphabet used for Sanskrit. Even though Kannada and Malayalam are languages of rich in historical literary, they are resource poor when viewed through the prism of computational linguistics [14]. In this paper most of the descriptions are based on Kannada and Malayalam languages.

3.1 Structure of South Dravidian languages

Unlike English language Kannada and Malayalam are syntax of relatively free word order language [15]. This can be easily illustrated with the example 'India defeated Pakistan in Lahore' as shown in table 1.

Table 1. Word order in Kannada and Malayalam languages

Case	Kannada	Malayalam
Case 1	ಭಾರತವು ಪಾಕಿಸ್ತಾನವನ್ನು ಲಾಹೋರಲ್ಲಿ ಸೋಲಿಸಿತ್ತು.	C´y]mInkvXms\ emtlmdnÂ tXmÂ]n¨p.
Case 2	ಪಾಕಿಸ್ತಾನವನ್ನು ಭಾರತವು ಲಾಹೋರಲ್ಲಿ ಸೋಲಿಸಿತ್ತು.	]mInkvXms\ C´y emtlmdnÂ tXmÂ]n¨p.
Case 3	ಭಾರತವು ಲಾಹೋರಲ್ಲಿ ಪಾಕಿಸ್ತಾನವನ್ನು ಸೋಲಿಸಿತ್ತು.	C´y emtlmdnÂ]mInkvXms\ tXmÂ]n¨p
Case 4	ಲಾಹೋರಲ್ಲಿ ಭಾರತವು ಪಾಕಿಸ್ತಾನವನ್ನು ಸೋಲಿಸಿತ್ತು.	emtlmdnÂ C´y]mInkvXms\ tXmÂ]n¨p

In all the cases, the subjects are 'ಭಾರತ' (bhArata) and 'C´y' (inthya) , the objects are ' ಪಾಕಿಸ್ತಾನ' (pAkistAna) and ']mInkvXm³' (pAkisthAn) and the locative is 'ಲಾಹೋರ್', 'emtlmÀ'(lAhOr). From the above example, it is clear that word order does not determine the functional structure in South Dravidian languages and permits scrambling. But normally South Dravidian languages follow Subject-Object-Verb orders in contradiction with English language..

3.2 Complexity and Ambiguity

The highly agglutinative languages like Kannada and Malayalam, nouns and verbs get inflected. Many times we need to depend on syntactic function or context to decide upon whether the particular word is a noun or adjective or adverb or post position [3][4]. This leads to the complexity in Kannada and Malayalam syntactic parsing. A noun may be categorized as common, proper or compound. Similarly, verb may be finite, infinite, gerund or contingent. Contingent is a special form of verb found only in Kannada and not found in other Dravidian languages. Other parts of speech were also divided into their own subcategories. Parts-of –speech ambiguity is the another important issue that have to be carefully analyse while designing a syntactic parser.

Table 2. Ambiguity in Kannada and Malayalam languages

Case	Kannada	Malayalam
Case 1	ಸೀತೆ ದೀಪದ ಬತ್ತಿ ಬದಲಿಸಿದಳು. (Seete deepada batti badalisidaLu)	അവന് കാലി തൊഴുത്തില് ജനിച്ചു (avan kAli tozhuthil janichu)
Case 2	ಬಾವಿಯ ನೀರು ಬತ್ತಿ ಹೋಯಿತ್ತು. (baaviya neeru batti hooyittu)	t]mIäv Imen Bbn (pOcket kAli Ayi)

For example, Kannada word 'ಬತ್ತಿ' (batti) and the Malayalam word 'കാലി' (kAli) in the following sentences in table 2 gives different parts of speech. The words 'ಬತ್ತಿ'(batti) and 'കാലി'(kAli) are nouns in the first case whereas in the second case these words act as verbs. Also, the parts of speech are not just the noun, pronoun, verb and adverb. There are clearly many more categories and sub-categories.While developing the syntactic parsers we have taken all the above factors into consideration.

3.3 PNG and Tense Markers

The Person–Noun-Gender (PNG) and the tense marker concatenated to the verb stems are the two important aspect of verb morphology in South Dravidian languages. Usually in South Dravidian languages, verbs follow the regular pattern of suffixation. PNG markers play an important role in word formation except in Malayalam. Depends on the noun case associated with noun phrase (NP), the PNG marker of corresponding verb in the verb phrase (VP) may change.

4. SYNTACTIC PARSING

Syntactic analysis is the process of analyzing a text or sentence that is made up of a sequence of words called tokens, and to determine its grammatical structure with respect to a given grammatical rules.

4.1 Syntactic Tree Structure

The different parts-of-speech tags and phrases associated with a sentence can be easily illustrated with the help of a syntactic structure. Figure 1 below shows the output syntactic tree structure produced by a syntactic parser for the Kannada input sentence ರಾಮ ಚೆಂಡನ್ನು ಎಸೆದನು ' Rama threw the ball'.

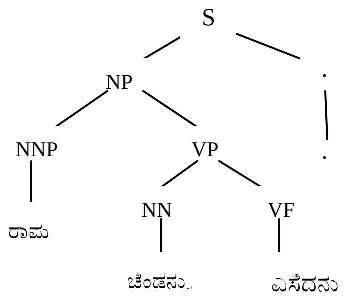


Figure 1. Syntactic tree structure

For a given sentence, the corresponding syntactic tree structure conveys the following information.

4.1.1 Part-of-Speech-Tags

The syntactic trees help in identifying the tags of all the words in a given sentence as shown in table 3.

4.1.2 Identification of Phrases

The syntactic tree also helps in identifying the various phrases and the organization of these phrases in a sentence. In the above example there are two phrases as shown in table 3.

4.1.3 Useful Relationship

The syntactic tree structure also helps in identifying the relationships between different phrases or words. Indirectly it identifies the relationship between different functional parts like subject, object, and verb in a sentence. In the given example, the subject part is ರಾಮ, object is ಚೆಂಡು and the verb is ಎಸೆದನು. Given a rule S ->NP VP in the above example, the NP (noun phrase) is the subject of the verb within the VP (verb phrase). In this case ' ರಾಮ ' (rAma) is the subject of 'ಎಸೆದನು ' (esedanu). Similarly the rule VP ->NP VP indicates an object-verb relationship. In our example 'ಚೆಂಡು' (ceMDu) is the object of verb 'ಎಸೆದನು ' (esedanu).

Table 3. Example: Parts-of-speechs and Phases in a sentence

Parts-of-speech		Phrases	
Node (Word)	**Tag**	**Phrase**	**Name of Phrase**
ರಾಮ	NNP	ರಾಮ	Noun phrase
ಚೆಂಡನ್ನು	NN	ಚೆಂಡನ್ನು ಎಸೆದನು	Verb phrase
ಎಸೆದನು	VF		

5. BRACKETING GUIDELINES FOR KANNADA PENN TREEBANK CORPUS

Penn Treebank corpora have proved their value both in linguistics and language technology all over the world. Information obtained from the Penn Treebank corpora has challenged the intuitive language study for various natural language processing purposes [16]. The main effort for developing a Penn Treebank based corpus is to representing the text in the form of a 'Treebank', where tree structures represent syntactic structures of phrases and sentences. This is followed by an application of a parsing model to the resulting Treebank. There for with the availability of Treebank of annotated sentences it is easy to develop natural language syntactic parser and other NLP application tools. Our effort is to create well balanced Treebank based corpus with almost all possible inflections. We have created a corpus with all types of sentences, some of which are illustrated as follow:

5.1 Simple Declarative Sentence

Consider a simple declarative sentence ರಾಮ ಚೆಂಡನ್ನುಎಸೆದನು (rama ceMDannu esedanu ' *Rama threw the ball'*). The figure 2 shows an example for the 'Penn tree syntax' and the figure 1 shows the corresponding parse tree for this sentence.

(S (NP (NNP ರಾಮ) (VP (NN ಚೆಂಡನ್ನು) (VF ಎಸೆದನು))) (. .))

Figure 2. Penn Treebank format of a Declarative sentence.

5.2 Imperative Sentences

(S (NP (NNP SBJ) (VP (NN ಚೆಂಡನ್ನು) (VF ಎಸೆದನು))) (! !))
[SBJ throw the ball !]

Figure 3. Penn Treebank format of an Imperative sentence

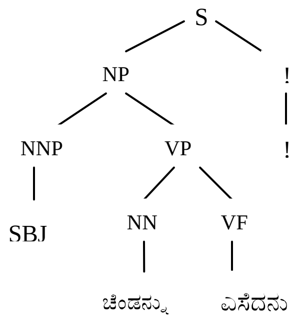


Figure 4. Parse tree for the figure 3.

Imperatives are formed from the root of the verb and usually given a null subject-SBJ, as shown in figures 3 and 4. Unlike Malayalam, depends on the type of noun case that is associated with the SBJ, the PNG markers associated with the Kannada verb also changes.

5.3 Compound Sentences

The relationship of conjoining in Kannada may be any one of: (i) additive indicated by ‘ಮತ್ತು’(mattu) or ‘ಊ’(U) (ii) alternative indicated by ‘ಅಥವ’ (adhava) or ‘ಇಲ್ಲಲಿಲ್ಲವೆ’(illalillave) and (iii) adversative indicated by ‘ಆದರೆ’(Adare). Coordination may be take place either at phrase or clause level. Figures 5 and 6 illustrate an example of Treebank format and parse tree for a compound sentence Coordinated at phrase level.

(S (NP (NN ಹುಡುಗಿಯರೂ) (NN ಹುಡುಗರೂ))
(VP (NN ಚೆಂಡನ್ನು) (VP (ADV ಎಸೆದು) (VF ಹಿಡಿಯುತ್ತಿದ್ದರು))) (. .))
[Girls and boys threw and caught the ball]

Figure 5. Penn Treebank format of a compound sentence coordinated at phrase level.

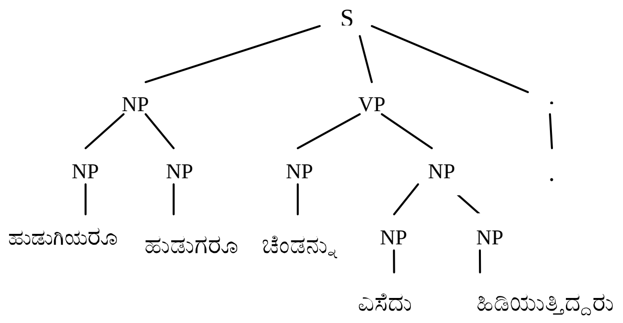


Figure 6. Parse tree for the figure 5.

6. SUPPORT VECTOR METHODS

SVM is a machine learning algorithm for binary classification, which has been successfully applied to a number of practical problems, including NLP. SVM is based on strong mathematical foundations and results in simple yet very powerful algorithms.In their basic form, a SVM learns a linear hyperplane that separates the set of positive examples from the set of negative examples with maximal margin[17][18]. The SVMs’ superiority over other classifiers is its ability to maximize the margin between classes. This learning bias has proved to have good in terms of generalization bounds for the induced classifiers. The linear separator is defined by two elements: a weight vector ‘w’ and a bias ‘b’ which stands for the distance of the hyperplane to the origin. The classification rule of a SVM is based on the equations (1) and (2).

$$sign\big(f(x,w,b)\big) \tag{1}$$

$$f(x,w,b) = \langle w \cdot x \rangle + b \tag{2}$$

Being ‘x’ the example to be classified. In the linearly separable case, learning the maximal margin hyperplane (w, b) can be stated as a convex quadratic optimization problem with a unique solution: minimize ||w||, subject to the constraints as indicated by the equation (3), one for each training example:

$$y_i(\langle w \cdot x_i \rangle b) \geq 1 \tag{3}$$

7. CONTEXT FREE GRAMMARS (CFG)

Context-free grammars, sometimes called a phrase structure grammar play a central role in the description of natural languages. In general a CFG [19] is a set of recursive rewriting rules called productions that are used to generate patterns of strings and it consists of the following components:

- A finite set of **terminal symbols** (Σ).
- A finite set of **non-terminal symbols** (**NT**).
- A finite set of **productions (P).**
- A **start symbol (S)**.

Consider an example for simple declarative sentence ರಾಮು ಚೆಂಡನ್ನು ಎಸೆದನು (rAmu ceMDannu esedanu). The components and the derivation of this sentence using CFG are shown in table 4. The tag assigned to each of the word is based on the POS tagger generator on table 6.

Table 4. Example: Context free grammar (CFG)

Production Rules (P)	**Derivation of Sentence**	
	Derivation	**Rule used**
S →NP VP	S →NP VP	S →NP VP
NP →NNP	S →NNP VP	NP →NNP
VP→NP VP	S → ರಾಮು VP	NNP→ ರಾಮು
NP→ NN	S → ರಾಮು NP VP	VP→NP VP
VP→VF	S → ರಾಮು NN VP	NP→ NN
NNP→ ರಾಮು	S → ರಾಮು ಚೆಂಡನ್ನು VP	NN→ ಚೆಂಡನ್ನು
NN→ ಚೆಂಡನ್ನು	S → ರಾಮು ಚೆಂಡನ್ನು VF	VP→VF
VF→ ಎಸೆದನು	S → ರಾಮು ಚೆಂಡನ್ನು ಎಸೆದನು	VF→ ಎಸೆದನು

Start Symbol: S, Terminal Symbols (Σ): {ರಾಮು, ಚೆಂಡನ್ನು, ಎಸೆದನು}, Nonterminal Symbols (NT): {S, NP, VP, NNP, NN, VF}

8. PROBABILISTIC CONTEXT FREE GRAMMAR (PCFG)

The problem of CFG is that it misses the probabilistic model which is needed in order to disambiguate between parses. A Probabilistic Context Free Grammar (PCFG) is a probabilistic version of a CFG where each production has a probability [20]. Probabilities of all productions rewriting a given non-terminal must add to 1, defining a distribution for each non-terminal. The simplest way to gather statistical information about a CFG is to count the number of times each production rule is used in a corpus containing parsed sentences. This count is used in order to estimate the probability of each rule being used. In our case, we estimate the rules probabilities using the relative frequency of the rule in the training set. For a generic rule “$A \to B\ C$”, this means that every time we find the symbol A, it can be substituted with the symbol B and C. Its conditional probability is defined as in equation (4):

$$P(A \to BC \mid A) = \frac{freq(A \to BC)}{freq(A)} \tag{4}$$

Once we have the probability of the production rules in a PCFG, the probability of a parse tree for a particular sentence can easily be calculated by multiplying the probabilities of the rules that built its sub-trees. The advantage of PCFG based syntactic parser model is that, for any two or more different sentences that have same pos tag sequence, but have different syntactic tree structure, then the sentence structure that has more probability would be considered or correctly parsed.

Table 5. Example: Probability context free grammar (PCFG)

Derivation of Sentence		Probability of rule used
Derivation	**Rule used**	
S →NP VP	S →NP VP	1.000
S →NNP VP	NP →NNP	0.050
S → ರಾಮು VP	NNP→ ರಾಮು	0.030
S → ರಾಮು NP VP	VP→NP VP	0.015
S → ರಾಮು NN VP	NP→ NN	0.025
S → ರಾಮು ಚೆಂಡನ್ನು VP	NN→ ಚೆಂಡನ್ನು	0.010
S → ರಾಮು ಚೆಂಡನ್ನು VF	VP→VF	0.015
S → ರಾಮು ಚೆಂಡನ್ನು ಎಸೆದನು	VF→ ಎಸೆದನು	0.015

Table 5 shows the total probability derivation of our previous sentence ರಾಮು ಚೆಂಡನ್ನು ಎಸೆದನು (rAmu ceMDannu esedanu) 'Rama threw the ball'. Total probability for derivation of sentence is calculated by multiplying the probabilities used to derive the sentence.

Total probability = 1.0 * 0.05 * 0.030 * 0.015 * 0.025 * 0.010 * 0.015 * 0.015

9. INSIDE-OUTSIDE ALGORITHMS (IOA)

Similar to the HMM's forward and backward algorithm, probability of nodes in a PCFG parse forest as the product of the inside and outside probabilities (IO probability) for the node *Ni* [21]. This can be easily understand by considering an example, for the grammar rule '*NP → DET NN'* over the input 'the man*'*. The corresponding node's IO probability is equal to the probability of all derivations which include the '*NP →DET NN'* category over this subset of the input. For production *i→ jk*, the probability of the rule is determined using the equation (5):

$$P(i \rightarrow ij) = \frac{freq(i \rightarrow ij)}{\sum_{j,k} freq(i \rightarrow jk)} \qquad (5)$$

Consider a CFG grammar G as a tuple {*NT*, *Σ*, *P*, *R*}, where *NT* and *Σ* elements represent the set of nonterminal and terminal symbols of the grammar respectively. The element *P* represents the set of production rules, while *R* represents the nonterminal category that is considered the top grammar category. For given input sequence of terminals of the grammar $\{a1, ...aT\}$, we denote ***e***(*s*, *t*, *Ni*) and ***f*** (*s*, *t*, *Ni*) are the inside and outside probabilities respectively for a node *Ni*, that spans input items *as* to *at* inclusively.

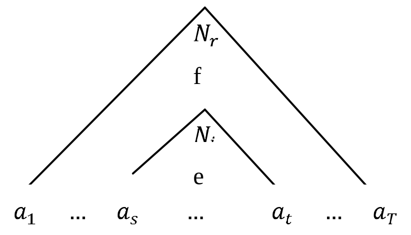


Figure 7. The inside (e) and outside (f) regions for node *Ni*)

Figure 7 illustrates the corresponding nodes in the parse forest used when calculating the inside and outside probabilities for *Ni*. Nonleaf nodes in the figure represent *NT* categories, and *Nr* is the root node whose category *r* is in the set *R*. Leaf nodes represent S categories of the grammar, that is, the input sequence $\{a1, ...aT\}$.

9.1 Inside Probability

The inside probability *e*(*s*, *t*, *Ni*) represents the probability of sub-analyses that are rooted with mother category *i* for this sentence over the word span *s* to *t*. Each production is of the form *i → jk*

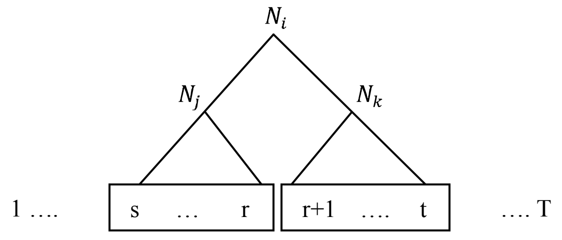


Figure 8. Inside probabilities for node Ni.

where each set of daughter nodes *Nj* and *Nk* span from *as* to *ar* and *ar*+1 to *at*, respectively. Figure 8 illustrates this structure for node *Ni*. Inside probability of each node corresponds to the product of all CFG rules that are applied to create the sub-analysis as shown in equation 6.

$$e(s,t,N_i) = \sum_{j,k}\left[P(i \rightarrow jk) \sum_{r=s}^{t-1} e(s,r,N_j)\, e(r+1,t,N_k)\right] \quad (6)$$

9.2 Outside Probability

On the other hand the outside probability *Ni*, *f* (*s*, *t*, *Ni*), for a node Ni is calculated using all the nodes for which the node is a daughter (sub-analysis). This calculation includes the inside probability of the other daughter nodes of which *Ni* is a member. This means, category *i* could appear in two different settings: *j→ik* or *j→ki*, as shown in figure 9.

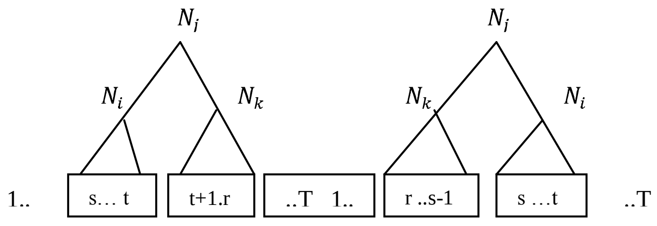


Figure 9. Outside probabilities for node Ni

The outside probability of *Ni* is calculated using the outside probability of the mother node (*Nj*) multiplied by the product of inside probabilities of the daughters other than *Ni* i.e. *Nk*. In each instance when *Ni* is a daughter of a node, the outside probability *f* (*s*, *t*, *Ni*) for a given sentence is calculated using the equation (7).

$$f(s,t,N_i) = \sum_{j,k}\left[f(s,r,N_j)P(j \rightarrow ik) \sum_{r=t+1}^{T} e(t+1,r,N_k)\right] + \sum_{j,k}\left[f(r,t,N_j)P(j \rightarrow ki) \sum_{r=1}^{s-1} e(r,s-1,N_k)\right] \quad (7)$$

10. IMPLEMENTATION

The proposed statistical syntactic parser for Kannada and Malayalam were based on Probabilistic Context free grammar (PCFG) and implemented using supervised machine learning approach with SVM algorithms. PCFG is basically a context free grammar (CFG) with probabilities associated with each rule, indicating how probable a production rule is. The SVM learning algorithm is used to create trained model which is used to identify the syntactic tree structure of new test sentences. When a sentence is given to be parsed, initially the pos tag of the words in the sentence is found out using Amrita pos tagger. In the subsequent steps the SVM classifier, using Inside-Outside Algorithm to find out the most probable parse structure of the given sentence.

10.1 Architecture of Proposed Syntactic Parser for Dravidian Languages

The architecture of the proposed syntactic parser is shown in figure 10.

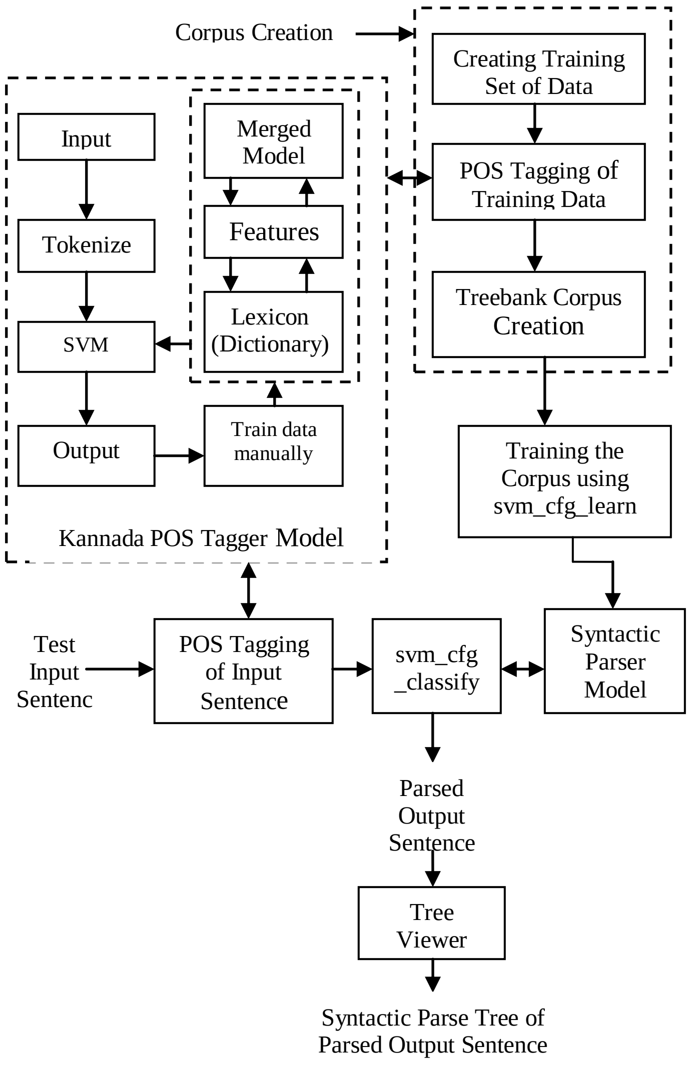


Figure 10. Architecture of Proposed Syntactic Parsing System

The proposed syntactic parser model consists of the following steps:

1. Creating training set of sentences.
2. POS Tagging the training sentences.
3. Format the syntactic structure of the training sentences.
4. Training the system using svm_cfg_learn module of SVM.
5. Testing the system with parser model created in the previous step using svm_cfg_classify module of SVM.
6. Display the output of input test sentence in syntactic tree form using Tree Viewer.

In any statistical system, the corpus creation is a major task which consumes considerable time. The first three steps in the proposed system were used to create the Treebank based corpus. A brief description of each of these steps is as follow:

10.1.1 Creating training set of sentences

The proposed Kannada and Malayalam Treebank corpora consist of 1,000 random diverse Kannada and Malayalam sentences. These sentences were carefully constructed by taking care of various factors for generating good corpora.

10.1.2 POS Tagging the training sentences

The next step was to assign parts-of speech tags to each and every word in the sentences using the POS tagger model. Parts-of speech tagging is an important stage in our Treebank based syntactic parsing approach. We have used our own POS tagger for assigning proper tags to each and every word in a sentence. POS tagger plays an important role in Natural language applications like speech recognition, natural language parsing, information retrieval and information extraction. We have developed statistical part-of-speech Taggers for Kannada and Malayalam languages using SVM algorithms based on Amrita tagset. These pos taggers were used for assigning syntactic tags to the words in the training and testing sentences. Table 6 shows the set of parts-of-speech syntactic tags that were used in our corpus for generating the syntactic parser. More detailed information on the POS tagset and guidelines concerning its uses are found in [3][4].

Table 6. Kannada POS tagset

Tag	Description	Example [Meaning in English]
<NN>	Noun	ಹುಡುಗ (huDuga) [boy]
<NNC>	Compound Noun	ಎತ್ತಿನ ಬಂಡಿ (ettina banDi)
<NNP>	Proper Noun	ಕರ್ನಾಟಕ (karnataka)
<NNPC>	Compound Proper Noun	ಅಬ್ದುಲ್ ಕಲಾಂ (Abdul kalam)
<CRD>	Cardinals	ಒಂದು (ondu) [one]
<ORD>	Ordinals	ಒಂದನೆ (ondane) [first]
<PRP>	Pronoun	ಅವನು (avanu) [he]
<ADJ>	Adjective	ಸುಂದರವಾದ [beautiful]
<ADV>	Adverb	ವೇಗದಲ್ಲಿ [speedly]
<VNAJ>	Verb Nonfinite Adjective	ಬಂದಹುಡುಗ [the boy who came]
<VNAV>	Verb Nonfinite Adverb	ಬಂದುಹೋದನು[came and went back]
<VBG>	Verbal Gerund	ಬರುವ (baruva) [coming]
<VBC>	Verb Contingent	ಬರುವೇನು (baruvEnu)[might come]
<VF>	Verb Finite	ಬರೆದೆನು (baredenu) [wrote]
<VAX>	Auxiliary Verb	ನೋಡುತ್ತಿದ್ದೇನೆ [was + ing]
<VINT>	Verb Infinite	ನೋಡಲು (nODalu) [to see]

<CNJ>	Conjunction	ಮತ್ತು (mattu) [and]
<CVB>	Conditional Verb	ನೋಡಿದರೆ (nODidare) [if see]
<QW>	Question Words	ಏಕೆ (Eke) [why]
<COM>	Complimentizer	ಎಂಬ (enba) [s , es]
<NNQ>	Quantity Noun	ಸ್ವಲ್ಪ (swalpa) [litte]
<PPO>	Post Position	ತನಕ (tanaka) [till]
<DET>	Determiner	ಆ (A)
<INT>	Intensifier	ತುಂಬಾ (tunbA) [very]
<ECH>	Echo Words	ಅಪ್ಪಿತಪ್ಪಿ (appi tappi) [by mistake]
<EMP>	Emphasis	ಮಾತ್ರ (matra) [only]
<COMM>	Comma	,
<DOT>	Dot	.
<QM>	Question Mark	?
<RDW>	Reduplication Words	ಪಟಪಟ (paTa paTa) [continuously]

10.1.3 Format the syntactic structure of the training sentences

The next step was to find out the syntactic structure of each and every sentence in the corpus manually by resolving various ambiguities and dependencies. The proposed statistical corpus was based on well known Penn Treebank corpora, so that the syntactic format of each and every training sentence manually created. The sentences in the training corpus were divided into various phrases and phrases are further divided to one or more words as described in the section 5.

10.1.4 Training the system using svm_cfg_learn

SVMcfg is a flexible and extensible tool for learning models in a wide range of domains. SVMcfg is an implementation of the Support Vector Methods (SVM) algorithm for learning a weighted context free grammar. The weight of an instantiated rule can depend on the complete terminal sequence, the span of the rule, and the spans of the children trees. Another important property of the SVMcfg is easy to add attributes that reflect the properties for the particular domain at hand. The SVMcfg mainly consists of two modules called learning module namely *svm_cfg_learn* and classification module namely *svm_cfg_classify*. These modules are used respectively for learning and classification for a set of data.

SVMcfg uses the learning module called *svm_cfg_learn* for learning the training corpus. The usage of this module is much like the svm_light module and the syntax is as follow:

svm_cfg_learn -c 1.0 train.data model

Which train SVM on training set *train.data* and output the learned grammar to the two model files called *model.svm* and *model.grammar* by setting the regularization parameter C to 1.0. In the proposed systems, 238 and 234 different rules were extracted for Kannada and Malayalam from the training data of 1000 sentences. Since the svm_cfg_leran module utilized the probabilistic context free grammar formalism, the module also finds out the probabilities of each and every rule as explained in section 8. Testing the system using svm_cfg_classify

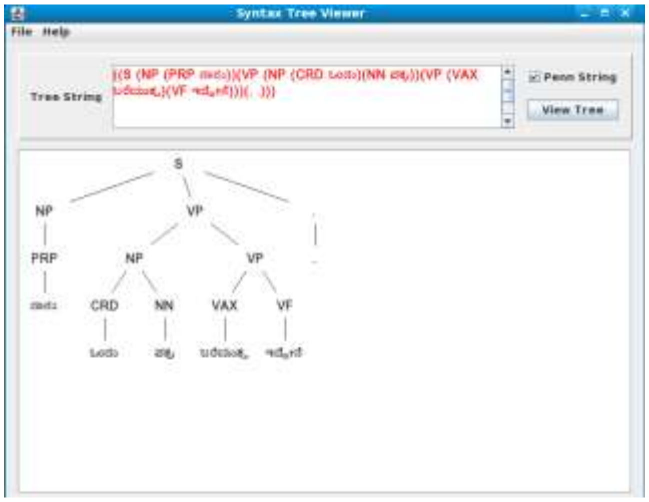
The trained model created in the previous step was used to predict the syntactic tree structure of new test sentences. The test file containing the test sentences were given to the POS tagger model to assigning syntactic tags to each and every word in the sentence. The result of the POS tagger was given to the svm_cfg_classify. Svm_ cfg_classify analyze the syntactic structure of test sentences by referring the model files that were created by svm_cfg_learn. Svm_ cfg_classify module makes predictions about the syntactic structure of test sentences based on probabilistic context free grammar formalism and inside-outside algorithms. Both PCFG and IOA are explained in the sections 8 and 9. The syntax of svm_cfg_classify is as follow:

svm_cfg_classify test.data model predictions

For all test examples in test.data, the predicted parse trees are written to a file called *predictions*.

10.1.5 Display the output using Tree Viewer

NLP or Linguistic researchers who work in syntax often want to visualize parse trees or create linguistic trees for analyzing the structure of a language. The syntactic parse tree of the test sentence is created and displayed by using 'Syntax Tree Viewer' software developed using Java language. Figure 11 shows the output screen shot for a test sentence 'ನಾನು ಒಂದು ಪತ್ರ ಬರೆಯುತ್ತ ಇದ್ದೇನೆ' (nAnu oMdu patra bareyutta iddEne-I am writing a letter).



```
((S (NP (PRP ನಾನು))(VP (NP (CRD ಒಂದು)(NN ಪತ್ರ))(VP (VAX
ಬರೆಯುತ್ತ)(VF ಇದ್ದೇನೆ)))(. .)))
```

Figure 11. Output Screenshot

11. EVALUATION AND RESULT

Even though we have developed a small sized corpus with 1000 distinguished sentences, the result obtained was well promising and encouraging. The performance of the system was evaluated using svm_cfg_classify module and the incorrect outputs were noticed. On contrast to the rule based approach, the systems performance was considerably increased by adding the input sentences to the training corpus whose corresponding outputs were incorrect during testing and evaluation. The graph in figure 12 shows the performance of proposed syntactic parser. We trained the systems with corpus size of 250, 500, 750 and 1000 sentences respectively. Performances of the systems were evaluated with the same set of 100 distinguished sentences that were out of corpus. From the experiment we found that the performances of our systems are significantly well and achieves very competitive accuracy by increasing the corpus size.

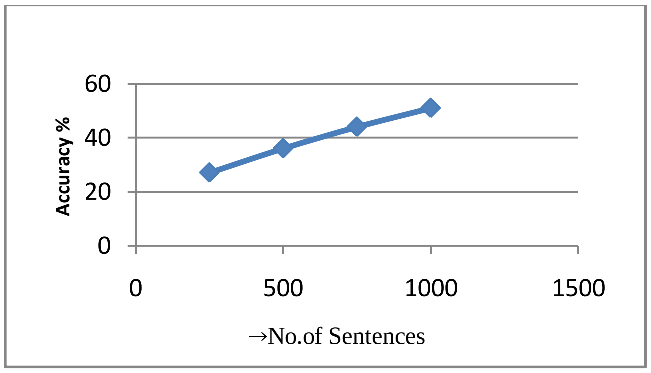


Figure 12. Performance Graph

12. CONCLUSION

From our experience we have noted that development in natural language processing for Indian languages like Kannada and Malayalam are very slow. The main reason for this includes non-availability of large scale data resources and also due to the inherent complexities of the language. The performance of the proposed syntactic parser models can improved by incorporating more syntactical information by increasing more and more sentence types and well-formed large corpus. We are working towards to generate full fledged syntactic parsers for all the South Dravidian languages. In future we can also use these syntactic parsers for tree to tree translation. This will be very useful for bilingual machine translation from English to South Dravidian languages. To the best of our knowledge this is the first attempt of computationally constructing statistical based syntactic parser models for Kannada and Malayalam languages.

13. ACKNOWLEDGMENTS

We acknowledge our sincere gratitude to Prof. M Shankaranarayana Bhat (Head of Kannada department and Principal, Junior College, Sampaje, Coorg, Karnataka, India) and Dr. Muralidhar (Professor, Malayalam Department, Govt. Victoria College, Calicut University, Kerala, and India.) for their excellent support to generate well organized Kannada and Malayalam corpora. We also express our gratitude to Mr. Shivapratab Gopakumar (Assistant. Professor, CEN, AMRITA Vishwa Vidyapeetham, Coimbatore, India) and Mr. Ajith V P (Research Associate, CEN, AMRITA Vishwa Vidyapeetham, Coimbatore, India) for their valuable support and encouragement for developing these prototype versions of Penn Treebank based syntactic parsers for Kannada and Malayalam languages.

14. REFERENCES

[1] Roxana Girju, (2004), “Introduction to Syntactic Parsing”.

[2] Niladri Sekhar Dash, (2004), “Present Indian Need”, Language Corpora.

[3] Antony P J. & Soman K P, (2010) “Kernel Based Part of Speech Tagger for Kannada”, International Conference on Machine Learning and Cybernetics 2010, ICMLC 2010, Qingdao, Shandong, China.

[4] Antony P J, Santhanu P Mohan & Soman K P, (2010), “SVM Based Parts Speech Tagger for Malayalam”, International Conference on-Recent Trends in Information, Telecommunication and Computing (ITC 2010), Kochi, Kerala, India.

[5] Reut Tsarfaty Yoav Goldberg, “Word-Based or Morpheme-Based? Annotation Strategies for Modern Hebrew Clitics”.

[6] Abhishek Arun, (2004), “Statistical Parsing of the French Treebank”, A thesis for Master of Science, Cognitive Science and Natural Language, School of Informatics, University of Edinburgh.

[7] Ayesha Binte Mosaddeque & Nafid Haque, (2004), “Context-Free Grammar for Bangla”, Bangla, Dhaka, Bangladesh.

[8] B.M. Sagar, Shobha G & Ramakanth Kumar P , (2009), “Solving the Noun Phrase and Verb Phrase Agreement in Kannada Sentences ”, International Journal of Computer Theory and Engineering , Vol. 1, No. 3, 1793-8201.

[9] Bala Sundara Raman L, Ishwar S, & Sanjeeth Kumar Ravindranath , (2003), “ Context Free Grammar for Natural Language Constructs – An implementation for Venpa Class of Tamil Poetry ”, I6th International Tamil Internet Conference and Exhibition, Tamil Internet 2003, Chennai,India.

[10] G.V. Singh & D.K. Lobiyal , (1994), “A Computational Grammar For Hindi Verb Phrase ”, IEEE transactions.

[11] Selvam M, Natarajan. A M, and Thangarajan R, (2008), “Structural Parsing of Natural Language Text in Tamil Using Phrase Structure Hybrid Language Model”, International Journal of Computer, Information, and Systems Science, and Engineering.

[12] www.languageinindia.com Vol 6 : 8 August, 2006.

[13] B. A. Sharada (2002), “Transformation of Natural Language into Indexing Language: Kannada - A Case Study**”**, Ph.D. Dissertation, Language in India- Strength for Today and Bright Hope for Tomorrow.

[14] T.N. Vikram & Shalini R Urs, (2007), “Development of Prototype Morphological Analyzer for the South Indian Language of Kannada”, Lecture Notes In Computer Science: Proceedings of the 10th international conference on Asian digital libraries: looking back 10 years and forging new frontiers. Vol. 4822/2007, 109-116.

[15] K Narayana Murthy, “Computer Processing of Kannada Language**”**, University of Hyderabad.

[16] V Tredinnick, (1995), “Bracketing Guidelines for Treebank II Style Penn Treebank Project”.

[17] Jes´us Gim´enez & Llu´ıs M`arquez, (2006), “SVMTtool: Technical manual**”**, v1.3.

[18] V.N. Vapnik, (1998), “Statistical Learning Theory : J.Wiley & Sons”, Inc. New York.

[19] Andrew McCallum , (2007), “Introduction to Natural Language Processing**”**, Lecture 5: Context Free Grammars.

[20] Qaiser Abbas, Nayyara Karamat & Sadia Niazi, (2002), “Development of Tree-bank Based Probabilistic Grammar for Urdu Language**”**, International Journal of Electrical & Computer Sciences IJECS. Vol: 9 No: 9.

[21] Rebecca F. Watson, (2009), “Optimizing the speed and accuracy of a Statistical GLR Parser**”**, Technical Report, University of Cambridge.