Journal of Information Technology Management

Print ISSN: 2008-5893
Online ISSN: 2423-5059

Generation of Syntax Parser on South Indian Language using Bottom-Up Parsing Technique and PCFG

M. Rajani Shree*

*Corresponding Author, Research Scholar, Visvesvaraya Technological University, Belagavi, Karnataka, Assistant Professor, BNMIT, Bengaluru, Karnataka. E-mail: rajanichandu.aeroit@gmail.com

Shambhavi B. R.

Associate Professor, Department of Information Science and Engineering, BMSCE, Bengaluru, Karnataka. E-mail: shambhavibr.ise@bmsce.ac.in

Abstract

In our research, we provide a statistical syntax parsing method experimented on Kannada texts, which is an official language of Karnataka, India. The dataset is downloaded from TDIL website. Using the Cocke-Younger-Kasami (CYK) parsing technique, we generated Kannada Treebank dataset from 1000 annotated sentences in the first stage. The Treebank generated in this stage contains 1000 syntactically structured sentences and it is used as input to train the syntax parser model in the second stage. We have adopted Probabilistic Context Free Grammar (PCFG) while training the parser model and extracting the Chmosky Normal Form (CNF) grammar from a Treebank dataset. The developed syntax parser model is tested on 150 raw Kannada sentences. It outputs with the most likely parse tree for each sentence and this is verified with golden Treebank. The syntax parser model generated 74.2% precision, 79.4% recall, and 75.3% F1-score respectively. The similar technique may be adopted for other low resource languages.

Keywords: Natural Language Processing; Artificial Intelligence; Syntax Parser; CYK Parsing Algorithm; Probabilistic Context Free Grammar.

Journal of Information Technology Management, 2023, Vol. 15, Special Issue, pp. 19-33
Published by University of Tehran, Faculty of Management
doi: https://doi.org/ 10.22059/jitm.2022.89408
Article Type: Research Paper
© Authors

Received: February 26, 2022
Received in revised form: May 07, 2022
Accepted: August 18, 2022
Published online: October 19, 2022

Introduction

Finding the right syntactic structure is the most important component of understanding any sentence. As a result, determining any sentence's syntactical structure is helpful in understanding its meaning. Syntax parsing is one of the most essential processes in Artificial

Intelligence (AI) for extracting the meaning from natural language speech or text. High end NLP applications like semantic parsing, sentiment analysis, word sense disambiguation, text summarization and many others require intensive syntactical analysis.

Even though much research has been done on several Indian languages, there is still a research lag on some Indian languages that are not English. Researchers developed a statistical parser for the Bangla language using PCFG (Probabilistic Context Free Grammar) and CYK approaches (A. Khatun and M. M. Hoque, 2017). On 2025 test sentences with a maximum of 9 words, they achieved an average accuracy of 93 percent. Only 1891 of the 2025 input sentences were correctly processed, leaving the remaining 134 sentences unparsed. Before being fed into their model, the words were given with POS labels (Kumar, D et al., 20221).

However, in our proposed study, we feed raw sentences to the parser model, which tokenizes each phrase into words and assigns POS labels based on the Treebank dataset's knowledge. In test phrases, the maximum number of words that we have given is 12. The strings of the grammar during execution are represented as integers in our work. This not only reduces the amount of RAM used, but it also increases performance due to the fact that the two integers can be compared in a single clock cycle (Basheer, S et al., 2020). Furthermore, in grammatical rules, string comparison is more computationally intensive. Implementing the PCFG, on the other hand, can significantly speed up the parsing process.

We worked efficiently to create a statistical parser that could parse natural text syntactically and the results are promising. On Kannada text, the CYK parsing algorithm has been used, which correctly parses each phrase and generates the syntactically structured text that corresponds to it. We generated a Kannada Treebank dataset from 1000 tagged phrases using CYK parsing algorithm and this is the first step towards our next implementation. This is a train dataset that will be used to build the syntax parser model. Similarly, we constructed a Treebank dataset with 150 labeled sentences and referred to it as the "Golden Treebank Dataset."

The CYK algorithm is a fast bottom-up parsing algorithm that eliminates some of the limitations of previous bottom-up methods. Intermediate solutions are generated and stored in a two-dimensional matrix during the process of parsing each text with the CYK approach. Only intermediate results that contribute to the whole parsing process will be pursued. To use CYK parsing technique, we need a special type of grammar known as Chomsky Normal Form (CNF), because the CYK parser only works with CNF grammar. Any component of the input sentence spanning i to j can be divided at k and structure can then be formed using sub-solutions spanning i to k and the following sub-solutions spanning k to j, according to the CYK parsing algorithm. During parsing, a matrix of size (n+1 * n+1) is created, where n is the number of words in the input. If a root node is identified at [0, n] inside the two-dimensional matrix, the CYK algorithm is successful for any text. A CYK parsing algorithm takes O (n3) seconds to execute.

The square of the number of non-terminals is exactly proportional to the complexity of the inner most loops formed during the execution of the CYK algorithm. The algorithm's efficiency decreases as the number of production rules increases. The complexity of the parsing process increases as the number of non-terminals in the production rules increases at a constant rate L = r2, where r is the number of non-terminals in the production rules. The basic CYK method necessitates the use of a binary grammar, which is a Chomsky Normal Form grammar. The algorithm can be tweaked to accommodate any CFG. Converting a grammar to a CNF grammar, on the other hand, is simpler and more efficient than parsing with an arbitrary grammar.

CNF is a restricted form of context-free grammar where the RHS of each production rule is restricted to be either two non-terminals or only one terminal and no empty productions are allowed. Also there can be no mixed rules that contain non-terminal as well as terminal like NP → N_NNP ವಿಧಾನ and no right hand side of the production rule contains more than two non-terminals like S → NP VP DT.

We have covered the background work in section 2, the dataset in section 3, the entire approach in section 4, and the results in section 5.

Literature Review

In the paper "A karaka based method to parsing of Indian languages", (Akshar Bharati and Rajeev Sangal, 1990), Linguistics researchers have created a concept of parsing Indian languages using a Karaaka based approach. The Kaaraka structure was used to define the grammar. The words from the sentences were collected and run through a morphological analyzer to retrieve the root and other grammatical information. The gathered data was then sent into a local word grouper, which created word groups solely on the basis of local data. The output from the local word group was fed into the core parser, which was tasked with determining the Kaaraka role for verbs. The Hindi language has been used to implement this method. According to Madhuri A Tayal et al. (Madhuri A Tayal et al., 2014), determining the syntactic structure of any sentence is useful in determining the meaning of the sentence, which is accurate in the majority of circumstances. They developed English grammar rules, tested the rule-based algorithm on real English sentences to see if the sentences were syntactically valid, and got an accuracy of 81 percent. The initial stage was to determine the type of sentence, such as simple, active, passive, complex, and so on. Match all terms of a sentence with the defined grammatical rules in the second step, which is determined by the type of sentence.

The authors presented a syntax analyzer for Kannada using the Kaaraka framework (Prathibha R J & Padma M C., 2019). They put their work through its paces on the EMILLI corpus and received a score of 75% accuracy. The connection made between the nouns and verbs included in a phrase is known as kaaraka. The authors used the Paaninian Grammar Model to create a rule-based syntax analyzer for Kannada texts. The subject-verb agreement was used to check the grammatical soundness of each sentence. Identifying and analysing the link between subject and verb, as well as gender, person (first, second, third), and number, is a difficult process.

Swati Ramteke et al. (2014) used Hindi wordnet to create a rule-based Lexicon parser for doing syntactic and semantic analysis on Devanagari script. The parser used semantic associations to create the parse trees. The lexical parser achieved an average accuracy of 89 percent when tested on new 150 sentences. They are able to achieve good accuracy because both syntactic and semantic analysis is performed on input. However, only a few numbers of POS were employed, including as nouns, pronouns, verbs, adverbs, and adjectives. They have implemented a parser that takes simple sentences, checks for sentence accuracy, and outputs results as either correct or incorrect sentence (Umamaheswaran, S et al, 2021).

For parsing Indonesian language, Cahyani et al. (2020) used comparable strategies to those suggested in this study, such as PCFG and Viterbi-CYK algorithm. Using PCFG on production rules, the Indonesian Corpus was initially pre-processed. The Viterbi-CYK algorithm was then used to parse and create the parse tree using sentences. The final output is the parse tree with the highest probability value. They were able to successfully test their theory on 129 Indonesian sentences. In 2017, another study (A. Khatun & M. M. Hoque) was published; they introduced a statistical parser for Bangla texts. The CYK Algorithm was used in conjunction with PCFG to build the most likely parse tree for each sentence. The statistical parser model was put to the test using a total of 2025 sentences, divided into three categories: simple, complex, and compound sentences. In 2018, a similar implementation was done on Bengali sentences by the same Authors (A. Khatun & M. M. Hoque), using the CYK method for parsing and PCFG to calculate the probabilities of the grammar rules. However, the testing dataset was expanded to 3500 sentences, which included all types of sentences, and an average accuracy of 85 percent was achieved.

Monika and Deepak (2015) stated that developing a qualitative parser for agglutinative and morphologically rich natural languages is a difficult undertaking. The majority of Indian languages are agglutinative. In their study, they compared the efficiency of numerous syntactic parser models constructed for a variety of Indian languages. They also discovered that for several Indian languages, the size of the annotated dataset utilized to train the syntax parser model is quite limited. In order to create effective parser models for several resource-poor Indian languages, the size of the annotated dataset must be considerably increased. In a similar way, Manisha Prajapati and Archit Yajnik (2018) conducted another survey on syntax parsing. The two main parser approaches, Rule-based and Statistical-based parsers, have been thoroughly compared and discussed. They also extracted and compared the results of several parsing approaches used on a variety of Indian languages.

In the year 2010 (B M Sagar et al.), a CFG analysis of simple Kannada texts was conducted. Only simple Kannada sentences were used to demonstrate the process of producing CFG by the authors. They next evaluate whether the input raw sentences can be processed successfully or not using the supplied CFG in the next phase. Top-down and bottom-up strategies were used to parse the grammar. (Praveen Sundar, P.V et al., 2020) has also attempted to develop simple CFG and grammar formation rules for Marathi sentences. The defined CFG was then applied to Marathi sentences, parsing each input sentence according to their defined grammar and determining whether the sentence was accepted or not (Shalini, A et al., 2018). They tested both top-down and bottom-up parsing strategies on the supplied grammar and concluded that the top-down parser is better for parsing the rules. The researchers (N. Chatterjee & S. Goyal, 2007) used an innovative way to parse English sentences. They derived parse information from texts that had already been parsed. From the constituent words of parsed text, knowledge was retrieved in two stages: link information and phrase templates. The input sentences were correctly parsed using these two, with an accuracy of more than 80%. The authors in 2019 (Altynbek Sharipbay et al.) suggested a syntactic analysis for simple Kazakh phrases based on the text's semantics. Formal grammar, such as CNF, parse trees, and ontological models, were used to derive the semantic link. Only simple sentences were used for syntactical analysis, and the results were impressive. Using the CYK parsing technique, Elliot Glaysher et al. (2006) attempted to minimize the parsing time required to create the complete parse tree. Because the CYK parsing technique examines for all conceivable combinations of grammatical productions, generating the parse trees for each sentence might take a long time. The researchers changed the CYK parsing code and streamlined the parsing procedure to avoid this. This was accomplished by just evaluating partial parsing process decisions rather than waiting for the process to finish (Karthikeyan, T et al., 2019).

Dataset

The Indian Government Resource Center TDIL (Technology Development for Indian Languages) provides the tagged Kannada dataset and this is used to generate the Kannada Treebank dataset in our implementation work. To assign POS (Parts of Speech) tags to each Kannada word, they used the BIS (Bureau of Indian Standard) tagset standard. To prepare the dataset for use, it was preprocessed and unnecessary symbols were removed. This dataset contains 1150 sentences from the Agriculture and Health domains. For developing the syntax parser model, 1000 sentences are taken as the training set and 150 sentences are considered as the test set.

Methodology

In the first step of our proposed work, 1000 Kannada annotated sentences are fed through the CYK parsing method to build a training Treebank set, which is then fed into the following step to train the parser model. To produce a golden Treebank dataset for test, 150 Kannada annotated phrases are fed into the CYK Parsing algorithm. For Kannada, a rule-based grammar is constructed to do the same. The grammar includes production rules and is a hybrid of CFG and CNF grammar. The grammatical rules that are not in CNF are converted from CFG to flexible CNF, as indicated in Table 1.

Table 1. Conversion from CFG to flexible CNF

CFG	**CFG converted to flexible CNF**
S → NP VP DT	S → NP+VP DT, NP+VP → NP VP
S → V_VM_VNF NP VP DT	S→ V_VM_VNF+NP+VP DT, V_VM_VNF+NP+VP → V_VM_VNF+NP VP, V_VM_VNF+NP → V_VM_VNF NP
NP → N_NN	NP → N_NN
NP → N_NNP	NP → N_NNP
NP → NP VP	NP → NP VP
NP → CC_CCS RB N_NNP	NP → CC_CCS+RB N_NNP, CC_CCS+RB → CC_CCS RB
NP → CC_CCS QT	NP → CC_CCS QT
QT → QT_QTF	QT → QT_QTF
QT → QT_QTO	QT → QT_QTO
VP → N_NST V_VM_VNF V_VM_VF	VP → N_NST+V_VM_VNF V_VM_VF, N_NST+V_VM_VNF → N_NST V_VM_VNF
VP → JJ N_NN V_VM_VNF	VP → JJ+N_NN V_VM_VNF, JJ+N_NN → JJ N_NN
VP → V_VM_VNF N_NN	VP → V_VM_VNF N_NN
VP → PSP VP	VP → PSP VP

Kannada Treebank dataset which is obtained in the initial step will be converted into CNF Treebank dataset for further processing. Because we need to train the syntax parser model and extract the grammar from a dataset which must be in Chomsky normal format. During the transformation of Treebank dataset into CNF tree like structure, we removed unary rules from all the sub-trees recursively. We can observe in Figure 1 that the unary production like VP → V_VM_VF is converted into VP & V_VM_VF and it is depicted in Figure 2.

One simple example of a tree-like structure of a sentence in original/non-CNF Treebank is given below:

(S

(NP (JJ ಎಳೆಯ) (N_NN ಮಕ್ಕಳು))

(VP

(N_NN ಪ್ರೂನಿಂಗ್)

(VP (N_NN ಹಂತವನ್ನು) (VP (V_VM_VF ದಾಟಿರುವುದಿಲ್ಲ))))

(DT .))

The above syntactically structured text is generated for a sentence “ಎಳೆಯ ಮಕ್ಕಳು ಪ್ರೂನಿಂಗ್ ಹಂತವನ್ನು ದಾಟಿರುವುದಿಲ್ಲ .” The meaning of this sentence in English is “Young kids have not yet crossed the pruning stage.” This sentence is taken from Health domain dataset. The corresponding NLTK Tree structure for this sentence is given in Figure 1.

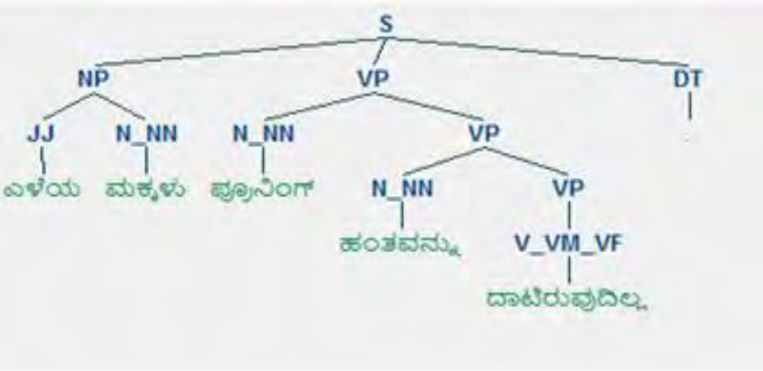

Figure 1**.** non-CNF tree structure

Now the original tree structure which has been shown in Figure 1 is converted into CNF tree and it is projected in Figure 2.

A tree-like structure of a sentence in CNF Treebank is:

(S

(NP (JJ ಎಳೆಯ) (N_NN ಮಕ್ಕಳು))

(S|NP
(VP
(N_NN ಪ್ರೂನಿಂಗ್)

(VP (N_NN ಹಂತವನ್ನು) (VP&V_VM_VF ದಾಟಿರುವುದಿಲ್ಲ)))

(DT .)))

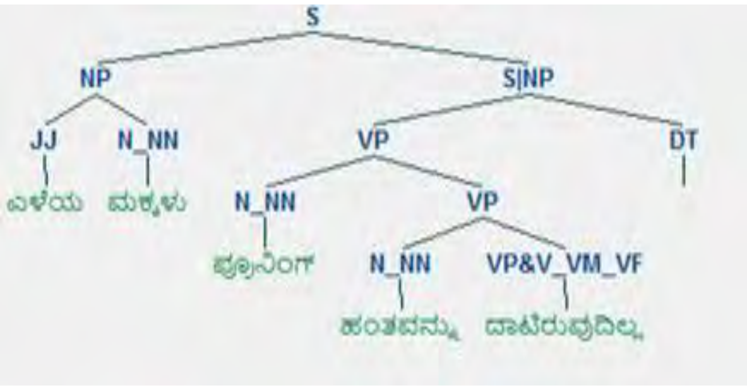

Figure 2. CNF tree structure

Similarly, all sub-trees from the Treebank are recursively converted to binary trees, a process known as grammar binarization. If you look at the first level of the tree in Figure 1, you'll notice that the production rule S → NP VP DT has three non-terminals in the RHS and is not in Chomsky Normal format. All these types of productions are turned into two non-terminals in the RHS and then it will be in CNF i.e after applying the binarization procedure. Figure 2 shows how S → NP VP DT production rule is transformed into S → NP S|NP, S|NP → VP DT. Figure 3 depicts the entire process of creating the Syntax parser model, training the model, generating parsed sentences from the model, and evaluating the model.

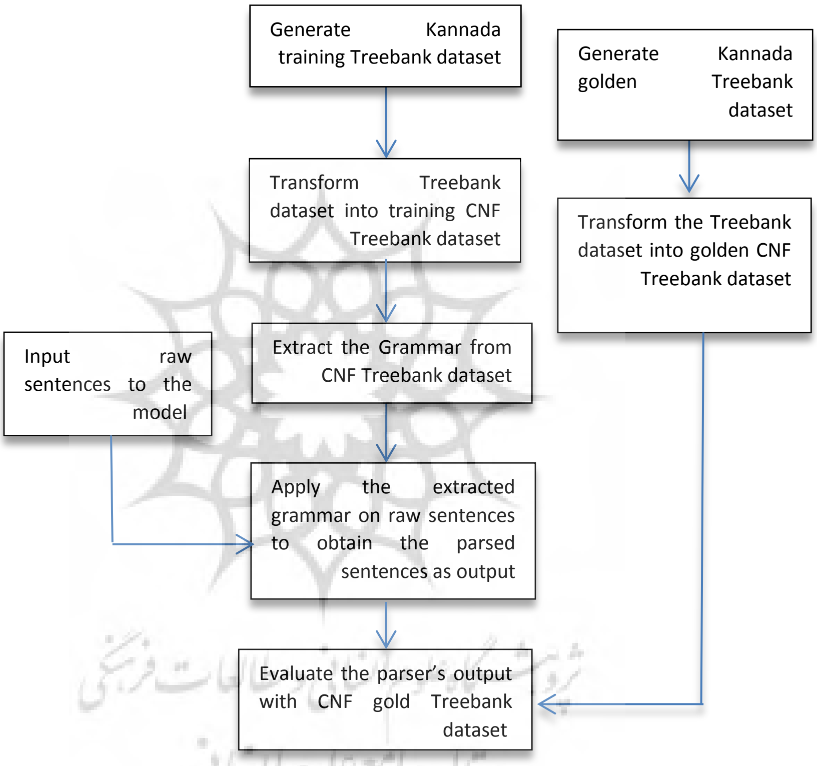
Generate Kannada training Treebank dataset
Generate golden Kannada Treebank dataset
Transform Treebank dataset into training CNF Treebank dataset
Transform the Treebank dataset into golden CNF Treebank dataset
Input raw sentences to the model
Extract the Grammar from CNF Treebank dataset
Apply the extracted grammar on raw sentences to obtain the parsed sentences as output
Evaluate the parser's output with CNF gold Treebank dataset

Figure 3. Block diagram of Syntax Parser model

Grammar is extracted from the CNF Treebank dataset to train the model in the following processing stage. The probabilities of each grammar rule are calculated and saved in a grammar file during the extraction of grammar from a CNF Treebank dataset. Each terminal, non-terminal, and the occurrences of two non-terminals combined are counted. The probability of the words (terminals) and non-terminals are calculated using the counts of their occurrences in the dataset.

An example of a unary count for calculating the probability is given below:

The non-terminal "V_VM_VNG" appeared in the dataset 4 times as a left hand side of the production rule (V_VM_VNG → "ಬೆಳೆಸುವುದು/to-grow") with a terminal "ಬೆಳೆಸುವುದು/to-grow" in the right hand side. While the overall count of non-terminal V_VM_VNG appeared in the dataset 384 times regardless of specific right hand terminals. As a result, the probability of a word or terminal "ಬೆಳೆಸುವುದು/to-grow" is 4/384 = 0.01, which will be stored in a CNF grammar file as ["Q1", "V_VM_VNG", "ಬೆಳೆಸುವುದು/to-grow", 0.01].

Similarly, in the right hand side of the production rule, the occurrences of two non-terminals together are also counted and utilized to determine the probability.

An example of the binary count for calculating the probability is shown below:

The two non-terminals DM_DMD N_NN appeared 3417 times in the right hand side of the production rule "NP → DM_DMD N_NN" along with NP in the left hand side. The total number of instances of NP on the left hand side of the production, regardless of any right hand side productions, is 33069. As a result, the probability of NP with regard to DM_DMD and N_NN is 3417/33069 = 0.10, which is saved as ["Q2", "NP", "DM_DMD", "N_NN", 0.10] in a CNF grammar file.

The original CFG grammar rule present at the top always is S → NP VP DT, this is transformed to Chomsky Normal format as specified in (1) and this process is termed as grammar binarization as mentioned earlier also.

S → NP S|NP (1)

S|NP → VP DT (2)

["Q2", "S|NP", "VP", "DT", 1.00] (3)

The probability of a grammar rule (2) is 1.0; it means the occurrences of this are more in a Treebank dataset. This has been added in a CNF grammar file as shown in (3) and it continues in this way during the development of syntax parser model. In order to parse every sentence successfully using CYK algorithm, it has to reduce ultimately to a grammar production rule (1) where S is the Start symbol. The probability of every non-terminal and terminal is computed by taking the natural logarithmic of the corresponding number stored in a grammar file. After calculating the probabilities of all terminals and non-terminals of grammar rules including the start symbol from a CNF Treebank dataset, these probabilities have been added in a CNF grammar file.

A CKY chart is created for each sentence during the parsing process conducted by the model. If a statement is eight words long, the chart will have a dimension of 64 (8*8). In general, if a sentence's length is n, the chart generated by the CYK algorithm will be n*n. To make a CYK chart, take each terminal or word from a sentence, determine its corresponding LHS or non-terminal, calculate their probability, and enter it into the chart. RHS1 is one non-terminal in a specific production rule. The next step is to extract all feasible combinations with another non-terminal known as RHS2 from the grammar file that match the pattern "LHS → RHS1 RHS2." To make an entry on the chart alongside LHS, add the probabilities of these three elements. The assigned probability for each rule is checked because we are using Probabilistic Context Free Grammar (PCFG). In a chart, if the probability of an existing item is smaller than the likelihood of a new item, the existing item is replaced by the new item.

For example: - The existing grammar rule “NP → N_NN DM_DMD” is assigned with the probability of -17.03 which is already stored in a chart. In the next processing steps, if another grammar rule with the same LHS (NP) and different RHSs like “NP → DM_DMD V_VM_VNF” is found and its probability is more than the existing one (-12.66), then the existing rule is replaced by the new rule in a chart. Here -17.03 is less than -12.66, accordingly the rule “NP → N_NN DM_DMD” is replaced by another rule “NP → DM_DMD V_VM_VNF”. In this manner, all entries are made, and then these entries from a CYK chart are traced back to get the full parsed tree of a sentence. The model generates parse trees with a CNF structure as its final output. In the final stage, compare the model's output to the golden Treebank dataset to assess the outcomes. Precision, recall, and F1-score measurements are used to present the findings.

Results

On Kannada text, we have presented a typical approach for identifying the grammatical structure of any sentence that uses both probabilistic and CYK techniques. After assessing the Kannada parser's performance, favorable results were found. Figure 4 depicts the general procedure of any Syntax parser model that may be constructed for any natural language.

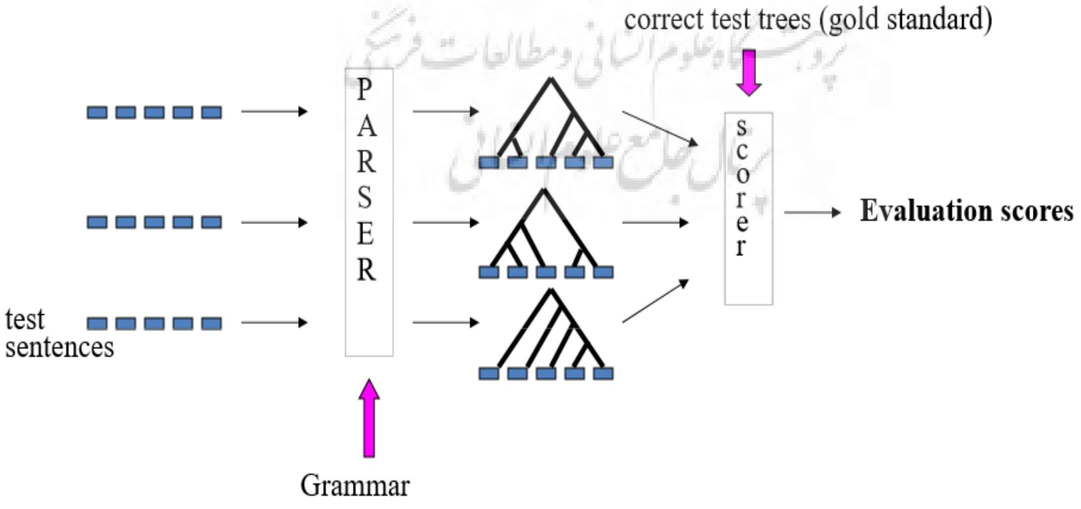

Figure 4. Evaluation of Parser model performance

Figures 5 and 6 demonstrate the parse tree of a sentence from a golden data set and the parse tree of a sentence obtained as an output from the syntax parser model, respectively.

The two parse trees shown in Figure 5 and Figure 6 are having the same sentence *“ಆ ಕಾರಣ ಈ ವಯೋಮಾನದಲ್ಲಿ ಗರಿಷ್ಠ ಕಲಿಕೆ ಸಾಧ್ಯವಾಗುತ್ತದೆ .”*. The meaning of this text in English is “*for this reason, it is possible to learn at the maximum extent”*. This is the continuation of a previous sentence “*Young kids have not yet crossed the pruning stage.”* which is shown in Figure 1 and Figure 2.

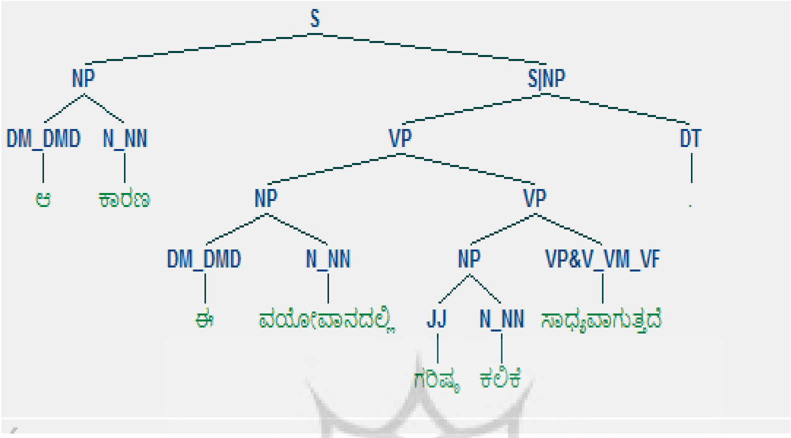

Figure 5. syntax tree of a sentence from golden dataset

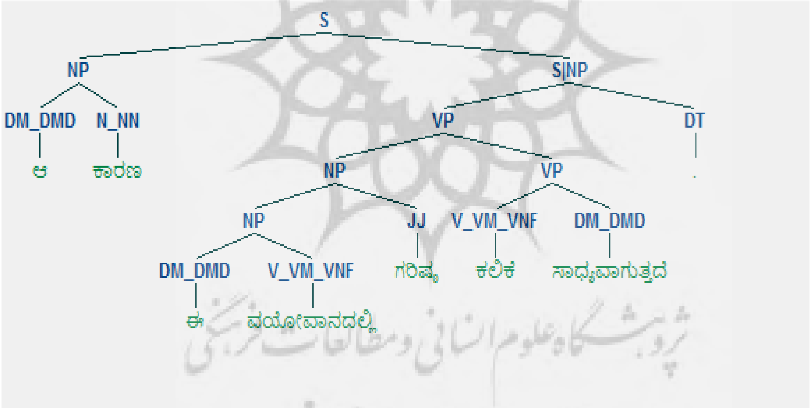

Figure 6. syntax tree of a sentence obtained from a parser model

{('S', 1, 8), ('VP', 3, 7), (**'NP', 5, 6**), ('S|NP', 3, 8), ('NP', 1, 2), ('NP', 3, 4)} (4)

{('S', 1, 8), ('VP', 3, 7), (**'NP', 3, 5**), ('S|NP', 3, 8), ('NP', 1, 2), ('NP', 3, 4)} (5)

The first set (4) comes from a golden Treebank dataset, and the parse tree for it can be shown in Figure 5. The second set (5) is a parser's output Treebank, with its associated parse tree depicted in Figure 6. These two sets have the same length, which are 6. Each item in a set is nothing more than a branch of a parse tree. Figure 6 and Figure 7 both are same, but Figure 7 highlights the sub-branch which is nothing but the item "'NP', 3, 5" from set (5). It signifies that 'NP' is the parent node in the highlighted sub-branch from Figure 7, and it covers the terminals or words from word 3 to word 5 in the sentence “ಆ ಕಾರಣ ಈ ವಯೋವಾನದಲ್ಲಿ ಗರಿಷ್ಠ ಕಲಿಕೆ ಸಾಧ್ಯವಾಗುತ್ತದೆ”.

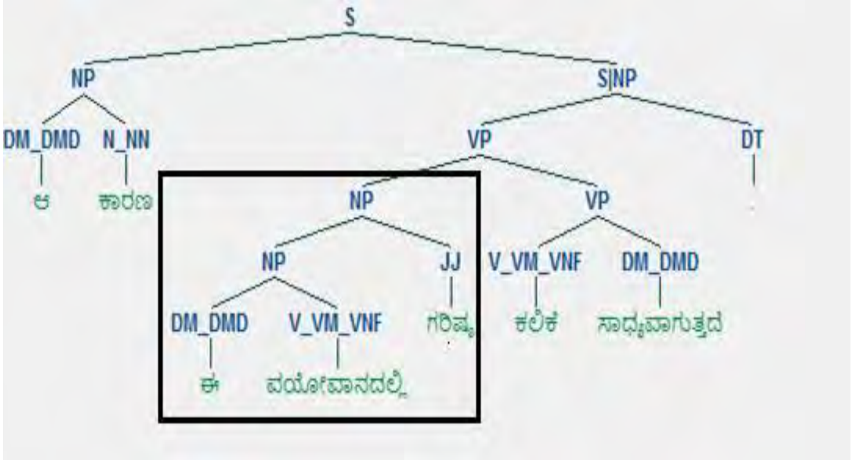

Figure 7. Parsed Tree obtained from Parser model

We can see that an item ('NP', 5, 6) in the first set (4) does not match with an item ('NP', 3, 5) in the second set (5). The remaining pieces from the two sets are otherwise identical. It indicates that the two parse trees from Figures 5 and 6 differ slightly. This is because the Kannada words in the golden Treebank dataset were manually tagged with suitable POS labels before the Treebank set was generated. However, the Syntax Parser model automatically assigns POS tags to the words in the generated Treebank set, which may or may not be totally appropriate. For example the word “*ವಯೋವಾನದಲ್ಲಿ*” is rightly tagged as *N_NN* in the golden dataset and it is depicted in Figure 5, but the same word is incorrectly tagged by the parser as *V_VM_VNF* in Figure 6. The words in the test dataset if not found in training Treebank dataset are interpreted as ‘*RARE’* words and automatically tagged by the model based on the patterns or sequence of the grammar rules learnt. Hence this may lead to the selection of different production rules which may not be the same as in the golden Treebank dataset.

Table 2 depicts the overall results obtained after comparing the parse trees from the golden Treebank dataset with the Treebank set created by the designed model. Table 2 shows that the row with "S" as a parent node or root node has precision, recall, and F1-score values of 1.00. It signifies that both Treebank sets have a 100 percent similar match. However, in another row, the non-terminal or parent node *"VP|RB"* has precision, recall, and F1-score of 0.00. It implies that, despite the fact that this non-terminal appeared as a parent node three times in the tree structure, no similar branch pattern was detected in any of the phrases, resulting in a score of 0.00. Table 2 contains only the parent nodes of trees or sub-trees; production rules such as “JJ → ಗರಿಷ್ಠ/Maximum”, “DM_DMD → ಆ/that” etc. (Means leaf nodes) and are not included.

Table 2. Average results gained from the built parser model

Parent nodes	Total	Precision	Recall	F1-Score
CC_CCD	10	0.364	0.444	0.400
NP	590	0.814	0.780	0.797
NP&NP2	2	0.000	0.000	0.000
NP|N_NN	5	0.800	0.800	0.800
NP|RP_INTF	0	0.000	0.000	0.000
N_NST	1	1.000	1.000	1.000
QT	7	0.500	0.500	0.610
S	150	1.000	1.000	1.000
S|CC_CCS	35	0.346	0.245	0.344
S|CC_CCS|NP	2	1.000	1.000	1.000
S|NP	125	0.938	0.922	0.930
S|V_VM_VNF	39	0.500	0.667	0.571
S|V_VM_VNF|NP	2	1.000	1.000	1.000
VP	190	0.740	0.763	0.751
VP|RB	3	0.000	0.000	0.000
VP|RP_INTF	6	0.133	0.333	0.190
VP|RP_RPD	1	0.000	0.000	0.000
V_VM_VNF	17	0.667	0.588	0.625
Total	1185	0.732	0.754	0.713

Conclusion

We used a novel strategy to combine CYK and probabilistic CFG approaches to create an effective statistical Syntax parser model for Kannada, which is discussed in this paper. The Kannada Treebank dataset, which contains 1000 syntactically organized phrases, was used to train the Parser. The prepared syntax parser model was put to the test using 150 unlabeled/raw Kannada phrases and the results were promising. The model's average precision, recall, and F1-score are respectively 74.2 percent, 79.4 percent, and 75.3 percent. If the sizes of both the train and test Treebank datasets are increased, the model will become more efficient, and this is a future goal. Given the paucity of research on many Indian languages, we set out to create an efficient tool for parsing Kannada phrases and producing a grammatically linked structured tree.

Conflict of interest

The authors declare no potential conflict of interest regarding the publication of this work. In addition, the ethical issues including plagiarism, informed consent, misconduct, data fabrication and, or falsification, double publication and, or submission, and redundancy have been completely witnessed by the authors.

Funding

The author(s) received no financial support for the research, authorship, and/or publication of this article

References

A. Khatun and M. M. Hoque. (2017). Statistical parsing of Bangla sentences by CYK algorithm, International Conference on Electrical, Computer and Communication Engineering (ECCE), pp. 655-661, doi: 10.1109/ECACE.2017.7912986.

A. Khatun and M. M. Hoque. (2018). Probabilistic Approach of Parsing Bengali Sentences, IEEE International WIE Conference on Electrical and Computer Engineering (WIECON-ECE), pp. 55-58, doi: 10.1109/WIECON-ECE.2018.8783000.

Akshar Bharati and Rajeev Sangal. (1990), A karaka based approach to parsing of Indian languages, COLING 90: Proceedings of the 13th conference on Computational linguistics, Volume 3, pages 25-29, https://doi.org/10.3115/991146.991151

Altynbek Sharipbay, Bibigul, Assel, Banu and Gaziza. (2019). Syntax parsing model of Kazakh simple sentences, DATA '19: Proceedings of the Second International Conference on Data Science, E-Learning and Information Systems, Article No. 54, Pages 1–5. DOI: 10.1145/3368691.3368745.

B M Sagar, G Shobha and P Ramakanth Kumar. (2010). Context Free Grammar Analysis for simple Kannada sentences, Proceedings of the International Conference [ACCTA-2010] on Special Issue of International Journal of Computer and Communication Technology (IJCCT), Vol. 1, Issue 2 , Article 6. DOI: 10.47893/IJCCT.2010.1033.

Basheer, S., Anbarasi, M., Sakshi, D.G. & Vinoth Kumar V., Efficient text summarization method for blind people using text mining techniques. Int J Speech Technol 23, 713–725 (2020). https://doi.org/10.1007/s10772-020-09712-z,

D. E. Cahyani, L. Gumilar and A. Pangestu. (2020), Indonesian Parsing using Probabilistic Context-Free Grammar (PCFG) and Viterbi-Cocke Younger Kasami (Viterbi-CYK), 3rd International Seminar on Research of Information Technology and Intelligent Systems (ISRITI), pp. 56-61, doi: 10.1109/ISRITI51436.2020.9315395.

Elliot Glaysher and Dan Moldovan. (2006). Speeding Up Full Syntactic Parsing by Leveraging Partial Parsing Decisions, Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions, pages 295–300, Sydney, Association for Computational Linguistics.

Karthikeyan, T., Sekaran, K., Ranjith, D., Vinoth kumar, V., Balajee, J.M. (2019) "Personalized Content Extraction and Text Classification Using Effective Web Scraping Techniques", International Journal of Web Portals (IJWP), 11(2), pp.41-52

Kumar, D., P., S., Jahangir, A., Sah, N. K., & V., V. (2021). Intelligent Speech Processing Technique for Suspicious Voice Call Identification Using Adaptive Machine Learning Approach. Advances in Computational Intelligence and Robotics, 372–380. doi:10.4018/978-1-7998-6870-5.ch025

Madhuri A Tayal, M.M. Raghuwanshi and Latesh Malik. (2014). Syntax Parsing: Implementation using Grammar-Rules for English Language, International Conference on Electronic Systems, Signal Processing and Computing Technologies (ICESC), DOI: 10.1109/ICESC.2014.71

Manisha Prajapati and Archit Yajnik. (2018). Parsing for Indian Languages: A Literature Survey, International Journal of Computer Sciences and Engineering, Vol.-6, Issue-8, E-ISSN: 2347-2693, https://doi.org/10.26438/ijcse/v6i8.10091018.

N. Chatterjee and S. Goyal. (2007). An Example Based Approach for Parsing Natural Language Sentences," 2007 International Conference on Computing: Theory and Applications (ICCTA'07), pp. 451-457, doi: 10.1109/ICCTA.2007.28.

Prathibha R J and Padma M C. (2019). Design of Syntax Analyzer for Kannada Sentences Using Rule-Based Approach, Emerging Research in Electronics, Computer Science and Technology, Lecture Notes in Electrical Engineering, Springer, DOI: 10.1007/978-981-13-5802-9_26.

Praveen Sundar, P.V., Ranjith, D., Vinoth Kumar, V. et al. Low power area efficient adaptive FIR filter for hearing aids using distributed arithmetic architecture. Int J Speech Technol 23, 287–296 (2020). https://doi.org/10.1007/s10772-020-09686-y

Shalini, A., Jayasuruthi, L., & VinothKumar, V. (2018). Voice Recognition Robot Control Using Android Device. Journal of Computational and Theoretical Nanoscience, 15(6), 2197–2201. doi:10.1166/jctn.2018.7436

Swati Ramteke, Komal Ramteke and Rajesh Dongare. (2014). Lexicon Parser for syntactic and semantic analysis of Devanagari sentence using Hindi wordnet, International Journal of Advanced Research in Computer and Communication Engineering, Volume 3, Issue 4, ISSN (Online) : 2278-1021 ISSN (Print) : 2319-5940.

Umamaheswaran, S., Lakshmanan, R., Vinothkumar, V. (2020). New and robust composite micro structure descriptor (CMSD) for CBIR. International Journal of Speech Technology (2019), Vol. 23, Issue 2, pp. 243-249

Bibliographic information of this paper for citing:

Shree, M. Rajani & B. R., Shambhavi (2023). Generation of Syntax Parser on South Indian Language using Bottom-Up Parsing Technique and PCFG. *Journal of Information Technology Management*, 15 (Special Issue), 19-33. https://doi.org/ 10.22059/jitm.2022.89408