Syntactical Parsing of Kannada Text Based on CYK Algorithm

Chapter · March 2022

DOI: 10.1007/978-981-16-7330-6_5

2 authors:

M. Rajani Shree
B. N. M. Institute of Technology
8 PUBLICATIONS **10** CITATIONS
SEE PROFILE

Ravi B Shambhavi
BMS College of Engineering
23 PUBLICATIONS **73** CITATIONS
SEE PROFILE

Syntactical Parsing of Kannada Text Based on CYK Algorithm

M. Rajani Shree and B. R. Shambhavi

Abstract Computational linguistics is one of the attractive research topics in Natural Language Processing (NLP) and Artificial Intelligence domains. This paper presents a particular syntactical parsing technique on Kannada texts which is one of the South Indian languages. Cocke–Younger–Kasami (CYK) parsing technique has been adopted to parse Kannada sentences and identify their grammatical structure. Currently, very less NLP tools are available to parse several Indian languages. Hence, an effort has been made by us to efficiently parse the structure of the complex sentences in Kannada text using CYK algorithm. It is a bottom-up dynamic programming approach which functions only with the grammar in Chomsky normal form (CNF). We are giving annotated Kannada sentences as input to the parser model and obtaining the grammatical correctness of each sentence as the output. The parser model generates NLTK parse tree only for grammatically correct sentences in the output. The error messages have been displayed for incomplete and incorrect Kannada sentences. This work has been implemented and tested on 1000 annotated Kannada sentences. The input dataset contains both simple and compound sentences in which many sentences are 20–22 words in length. We obtained considerable results from the proposed syntax parser model which is implemented on Kannada text.

Keywords Natural language processing · CYK algorithm · Syntactical parsing · Chomsky normal form · Production rules · Rule-based grammar

M. Rajani Shree (✉)
Visvesvaraya Technological University, Belagavi, India
e-mail: m.rajanishree@jainuniversity.ac.in

Department of CSE, Jain Deemed to be University, Bengaluru, India

B. R. Shambhavi
Department of ISE, BMSCE, Bengaluru, India
e-mail: shambhavibr.ise@bmsce.ac.in

A. P. Pandian et al. (eds.), *Proceedings of Third International Conference on Intelligent Computing, Information and Control Systems*, Advances in Intelligent Systems and Computing 1415, https://doi.org/10.1007/978-981-16-7330-6_5

1 Introduction

A very little attempt has been made to develop a syntactical parser on South Indian languages including Kannada. Kannada is a spoken as well as written language in Karnataka state and it is basically very agglutinative in nature. We made an effort to develop a systematic syntax analyzer to parse all types of Kannada texts, and the same has been presented in the paper. Python programming language has been used and NLTK tree module also been imported to surve the purpose. NLTK tree module has been adopted to display the parser tree for every valid Kannada sentence. Syntax analysis is the process of checking the grammatical correctness of a sentence and identifying the relationship between the words in an input sentence. It is very simple to write the production rules or grammar for fixed order spoken languages compared to other free order languages. Hence, this becomes a challenging task to write the grammar rules for any free order languages and Kannada is a free order language.

CYK algorithm follows bottom-up and dynamic programming approach in building the parser model. In the bottom-up parsing technique, the parser make an attempt to reduce the input words (considered as terminals) from a sentence to the start symbol S through the defined grammar G. This is accomplished by tracing out the RHS and its corresponding LHS of every production. Here, RHS means terminals (input words), and LHS means non-terminals (POS tags). This process will be continued repeatedly until it reaches to a start symbol S on the LHS. But the constraints in CYK parsing algorithm is that it will work only on context-free grammar (CFG) which is defined in Chomsky normal form (CNF). CNF is a restricted version of CFG and it allows only the following types of production rules:

$$S \rightarrow AB, \quad A \rightarrow a$$

where "S", "A", and "B" are non-terminals. "S" is a start symbol in the grammar, and "a" is a terminal symbol. A production can have either two non-terminals or can have a single terminal on the right hand side of the production.

We developed a rule-based grammar G which contains productions in the CNF format. If any productions are not meeting the strict CNF standard, then these are converted into flexible CNF automatically by the program itself. The tagged Kannada dataset is downloaded from Technology Development for Indian Languages (TDIL) Web site [1]. The dataset which has been selected for parsing in our proposed work contains all types of sentences like imperative, declarative, assertive, interrogative, and compound sentences with conjunctions. The words in every sentence are annotated with POS labels which followed BUREAU OF INDIAN STANDARD (BIS) tagset standard [2].

A sample of rule-based grammar which is framed to implement the proposed work is given in Table 1. The tags that are shown in table all comply with BIS Tagset [3]. It can be observed from Table 1 that few grammar rules which are not in CNF are converted into flexible CNF.

Table 1 Rule-based grammar

S → NP VP
NP → N_NNP N_NNP
NP → N_NN
NP → NP VP
NP → CC_CCS NP
NP → CC_CCD NP
NP → N_NNP
NP → RP_INTF NP
NP → JJ NP
NP → N_NN + N_NN VP
N_NN + N_NN → N_NN N_NN
NP → DM_DMD NP
NP → QT_QTF NP
NP → RP_RPD NP
NP → N_NST NP
VP → CC_CCD VP
NP → PR_PRP
NP → PSP NP
VP → NP VP
VP → RP_INTF + QT_QTF VP
RP_INTF + QT_QTF → RP_INTF QT_QTF
VP → QT_QTF + JJ NP
QT_QTF + JJ → QT_QTF JJ
VP → RB VP
VP → V_VM_VNG
VP → V_VM_VINF VP

For Example: VP → QT_QTF + JJ NP, QT_QTF + JJ → QT_QTF JJ are the two flexible CNF Grammar converted from original grammar like VP → QT_QTF JJ NP. i.e the production rules having 3 non-terminals in the RHS have been transformed into 2 non-terminals.

In Kannada sentences, the words may be in any one of the three orders like ***subject object verb***, ***object subject verb,*** and ***subject verb object***. All these are valid sentences unlike in English. In English, the words should always be in *subject verb object* (SVO) format.

For example

ರಾಜು ಶಾಲೆಗೆ ಹೋದನು (Raju to School went) (S1)

ಶಾಲೆಗೆ ರಾಜು ಹೋದನು (To School Raju went) (S2)

ರಾಜು ಹೋದನು ಶಾಲೆಗೆ (Raju went to School) (S3)

In this given example, S1, S2, S3 all are grammatically correct Kannada sentences. To parse these three different sentences which give the same meaning, we need to add the corresponding production rules in the grammar.

S → NP VP
VP → NP VP
VP → VP NP
NP → N_NN
NP → N_NNP
VP → V_VM_VF
N_NNP → “ರಾಜು”
N_NN → “ಶಾಲೆಗೆ”
V_VM_VF → “ಹೋದನು”

Where S is a start symbol, {NP, VP, N_NN, N_NNP, V_VM_VF} are all non-terminal symbols, and {“ರಾಜು”, "ಶಾಲೆಗೆ", “ಹೋದನು"}are all terminals in the productions rules.

The syntactic structure of S1 is

(S (NP (N_NNP “ ರಾಜು”)) (VP (NP (N_NN “ ಶಾಲೆಗೆ”)) (VP (V_VM_VF “ ಹೋದನು”))))
(S
(NP (N_NNP “ ರಾಜು”))
(VP (NP (N_NN “ ಶಾಲೆಗೆ”)) (VP (V_VM_VF “ ಹೋದನು”))))

The syntactic structure of S2 is

(S (NP (N_NN “ ಶಾಲೆಗೆ”)) (VP (NP (N_NNP “ ರಾಜು”)) (VP (V_VM_VF “ ಹೋದನು”))))
(S
(NP (N_NN “ ಶಾಲೆಗೆ”))
(VP (NP (N_NNP “ ರಾಜು”)) (VP (V_VM_VF “ ಹೋದನು”))))

The syntactic structure of S3 is

(S (NP (N_NNP “ ರಾಜು”)) (VP (VP (V_VM_VF “ ಹೋದನು”)) (NP (N_NN “ ಶಾಲೆಗೆ”)))).
(S
(NP (N_NNP “ ರಾಜು”))
(VP (VP (V_VM_VF “ ಹೋದನು”)) (NP (N_NN “ ಶಾಲೆಗೆ”))))

Hence, it will be a challenging task to develop a parser to parse such free order languages. This paper presents the technique of identifying the syntactic structure of input sentences using CYK algorithm, and we also mentioned its limitations (Figs. 1, 2, and 3).

2 Related Work

Nitin Hambir and Ambrish Srivastav in 2012 [4] have developed a Hindi parser using CYK algorithm. They have used database to fetch Hindi sentences and implemented with the help of user interface. The authors have obtained considerable results

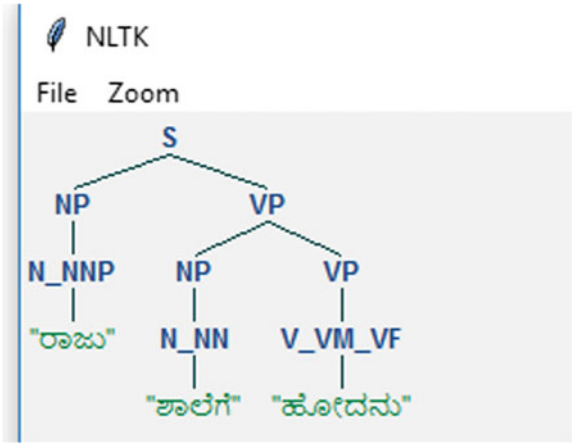


Fig. 1 Parse tree for S1

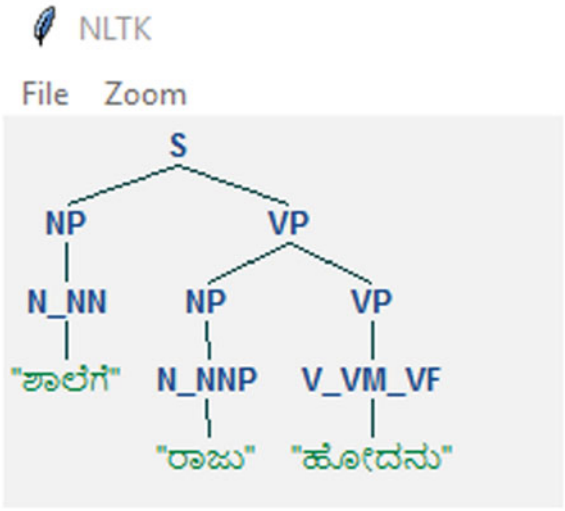


Fig. 2 Parse tree for S2

and able to parse Hindi sentences containing 4–5 words successfully. In the paper, Prathibha and Padma [5] have proposed a syntax analyzer on Kannada language using *Kaaraka* framework. They tested a parser on EMILLE corpus and gained an accuracy of 75%. The syntax analyzer proposed here did analysis on the given Kannada sentence and checked for its grammatical correctness by verifying the subject–verb compatibility. 500 Chhattisgarhi sentences have been parsed using CYK parser and obtained an accuracy of 82% in the paper, "Parsing Chhattisgarhi Languages Using CYK Algorithm and Grammar Rules". This was presented by Vikas Pandey et.al. [6]. Words in each sentence were labeled with POS tags and checked for syntactic parsing. The outputs were shown in the form of parse tree.

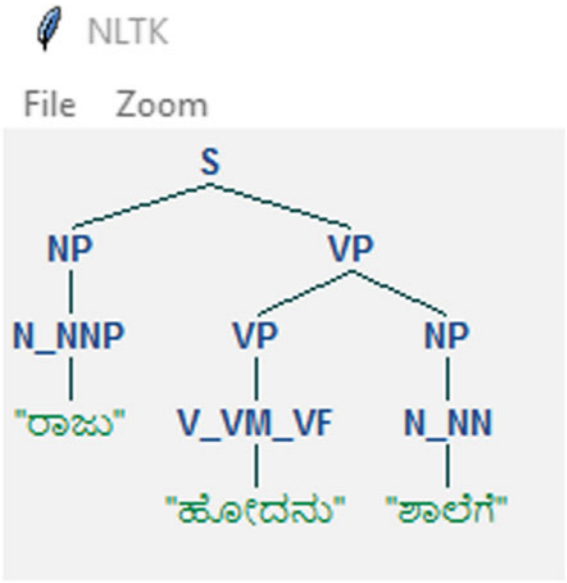


Fig. 3 Parse tree for S3

The authors [7] made a survey of many research papers on syntactic analysis for Indian languages. Rule-based parser and statistical-based parser were detailed in general in this paper. Also, they have compared different parser tools developed by different researchers implemented on Indian languages like Hindi, Kannada, Telugu, Bangla, and Bengali. The methods and techniques used in these parsers along with their performances were discussed. A simple CFG grammar has been built only for simple Kannada sentences having words not more than 3, and this was proposed by Sagar et al. in 2010 [8]. They have used shift reduce parser technique in their work and generated parse trees on simple Kannada sentences.

Kulkarni [9] in 2014 has presented a syntactic parser on Marathi sentences. She has expressed the difficulty in parsing Indian languages because of different gender and number forms. She made an effort to construct simple CFG for simple Marathi texts. Another effort has been made to parse the English sentences [10] using rule-based approach. They tested on all kinds of English sentences like simple, active, passive, interrogative, and compound sentences, and an accuracy of 81% has been achieved. They have shown the output whether a sentence is grammatically correct or not without generating the parse tree for the same.

Prajapati and Yajnik [11] surveyed on different parsing techniques which have been adopted on many Indian languages. Top-down parsing, bottom-up parsing, rule-based approach, and statistical-based techniques were discussed in this paper along with their pros and cons. The authors [12] proposed a syntax analysis on simple Kazakh sentences. In this paper, they have defined the syntactic rules of sentences in terms of CFG as a formal grammar. From this, the parsing analysis has been made, built parsing trees and ontological models. A novel approach has been adopted to

parse English sentences by the researchers [13]. They extracted the parse knowledge from already parsed sentences. The knowledge has been extracted in two stages like link information and phrase templates from constituent words of parsed text. Using these two, the input sentences were parsed successfully and achieved more than 80% accuracy. In the recent paper (2020) [14], the authors have performed syntactical analysis on Odia language, and this has been done based on Panini method. They have used context-free grammar along with top-down parsing approach. Another survey has been made on Indian languages [3] that they expressed the challenges of parsing and understanding Indian languages because of differences exist in grammar and structure. They have reviewed on various research work carried on several Indian languages and discussed the results of those techniques.

3 Methodology

An input sentence is processed by a parser according to the productions of a grammar and builds one or more constituent structures that satisfy the grammar. A parser is a procedural interpretation of the grammar. It permits a grammar to be evaluated against a collection of test sentences, helping linguists to find mistakes in their grammatical analysis.

The CYK parser model proposed in this paper reads CFG grammar and converts into CNF format if required. The Kannada dataset which contains different kinds like imperative, declarative, assertive, compound sentences, and interrogative sentences has been given as input to the parser model. Each word is treated as a terminal symbol. These terminals are matched with the production rules in the CNF grammar and try to fetch the corresponding productions by RHS. This continues to identify the respective LHS of the productions recursively through the bottom-up approach until it gets the start symbol (S). Finally, if the start symbol of the grammar is not obtained, then the input sentence given is not accepted by the grammar. If the start symbol is obtained, it outputs with the NLTK parse tree. For some sentences, it may create more than one parse tree when parsing technique gets the same start symbol through different production rules.

The simple parsers suffer from limitations in both completeness and efficiency. In order to overcome these, dynamic programming technique has been applied here to resolve the parsing problem. Dynamic parsing algorithm accumulates the intermediate results and reuses them when relevant, achieving efficiency gains. This technique can be applied to syntactical parsing, allowing us to store partial solutions during the parsing task and then look them up as necessary in order to efficiently arrive at a complete solution. This approach to parsing is known as chart parsing.

A typical sentence has been taken as an example to present the proposed technique. The sentence (S4) is in subject object verb (SOV) format which is the standard form in Kannada language. Of course, any other format is also valid and gives the same meaning.

Table 2 Chart of CYK algorithm

ಹೋದನು (2,0)	(2,1)	(2,2)	V_VM_VF, VP (2,3)
ಶಾಲೆಗೆ (1,0)	(1,1)	N_NN, NP (1,2)	S, NP, VP (1,3)
ರಾಜು (0,0)	N_NNP, NP (0,1)	(0,2)	S, NP, VP (0,3)

Where the tags of these words are ರಾಜು–N_NNP, ರಾಜು–N_NNP, ಹೋದನು–V_VM_VF

ರಾಜು ಶಾಲೆಗೆ ಹೋದನು (Raju to School went) (S4)

The related production rules are S → NP VP**,** NP → NP VP**,** VP → NP VP**,** NP → N_NNP**,** NP → N_NN**,** VP → V_VM_VF**,** N_NNP → “ ರಾಜು”**,** N_NN → “ ಶಾಲೆಗೆ”**,** V_VM_VF → “ ಹೋದನು”.

S is a start symbol as well as a non-terminal. {NP, VP, N_NNP, N_NN, V_VM_VF} are all other non-terminals. {“ ರಾಜು”, “ ಶಾಲೆಗೆ”, and “ ಹೋದನು”} are all terminals.

The CYK parser finds the LHS of the respective terminal symbols in the production rules recursively adds it to the table, and the same is shown in Table 2. The final result of the parser for every sentence is depending on the table’s 0th index and the last index, i.e., table [0][3] here, because the last word index in this sentence (s4) is 3 which is the length of the sentence (3 words in a sentence in this example). The last column and first row in the chart entry must contain the start symbol **S**, to declare the given input sentence is grammatically correct or else it outputs that the sentence is not accepted by the grammar. Hence, CYK is a type of bottom-up parsing algorithm implemented on rule-based grammars.

4 Limitations

It will become very difficult to increase the size of grammar in order to cover large corpora of natural languages and very hard to keep control of the complex interactions between the many productions required to cover the vital constructions of a language. Another difficulty is that as the grammar extends to cover a wider and wider range

of constructions, there is a corresponding growth in the number of analyses that are admitted for any one sentence. In other words, ambiguity increases with the coverage.

CYK algorithm is an efficient parsing and bottom-up dynamic programming technique. But it leads to an increase in space and time complexity if the grammar is having more number of matching productions for one particular sentence. We have shown in this paper few lengthy and compound sentences of up to 15 words with different POS tags. In some cases, these kinds of sentences generate multiple parse trees and go up to 20 different parse trees for one sentence. The reason is CYK algorithm considers every possible production rule recursively which generates the same start symbol. Ambiguity arises in these types of situations.

5 Test and Results

The total number of Kannada input sentences given is 1000 which includes different types of sentences and different domains like agriculture, health, and education system. It also includes many complex sentences. All grammatically correct sentences are represented by the NLTK parse trees as the output, and incomplete or invalid sentences are outputted as "not accepted" without generating parse tree.

For example:

The sentence, " ಪುದಿನಾವನ್ನು ವರ್ಷದ ಎಲ್ಲ ದಿನಗಳಲ್ಲಿ " (Mint all days in a year), is not accepted by the given grammar and finally! Since this is an incomplete one.

A sentence "ಅಶ್ವಗಂಧಕ್ಕೆ ಬೇಡಿಕೆ ಹೆಚ್ಚುತ್ತಿರುವುದರಿಂದ ವಾಣಿಜ್ಯ ಬೆಳೆಯಾಗಿ ಬೆಳೆದು ಲಾಭ ಗಳಿಸಬಹುದು " (Since the demand is increasing for Ashwagandha, it may be grown as commercial crop and get the profit) is having two valid parse trees as shown in Figs. 4 and 5. This is because of the two different production rules VP → VP NP and VP → NP VP and can be observed in the respective figures.

ಅಶ್ವಗಂಧಕ್ಕೆ ಬೇಡಿಕೆ ಹೆಚ್ಚುತ್ತಿರುವುದರಿಂದ ವಾಣಿಜ್ಯ ಬೆಳೆಯಾಗಿ ಬೆಳೆದು ಲಾಭ ಗಳಿಸಬಹುದು (S5)

Kannada is such a rich language in which we can frame a valid sentence which does not have a verb at all. For example:

The sentence " ಮೈಸೂರು ದಸರಾ ಒಂದು ಸುಂದರವಾದ ಹಬ್ಬ" (Mysore Dasara is a beautiful festival) is *not having any action word*. But still this is a meaningful sentence in Kannada. The parser tree of this is shown in Fig. 6.

ಮೈಸೂರು ದಸರಾ ಒಂದು ಸುಂದರವಾದ ಹಬ್ಬ (S6)

As shown in Fig. 6, the parse tree of sentence S6, the VP production has three non-terminals on the RHS.

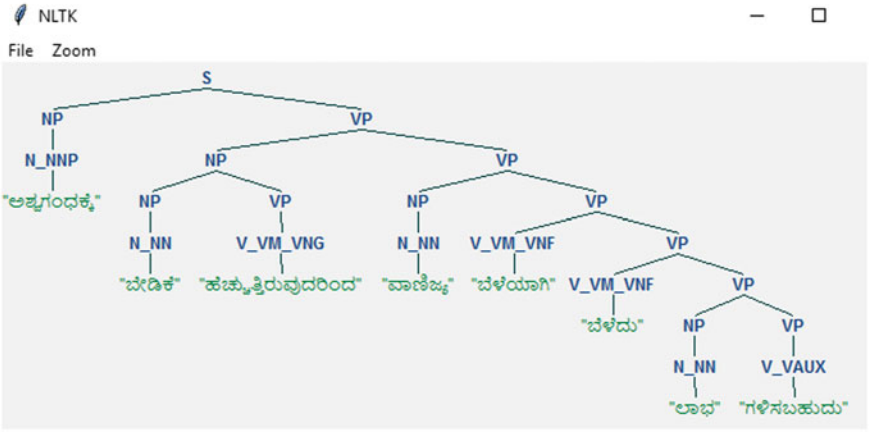


Fig. 4 Parse tree 1 for S5

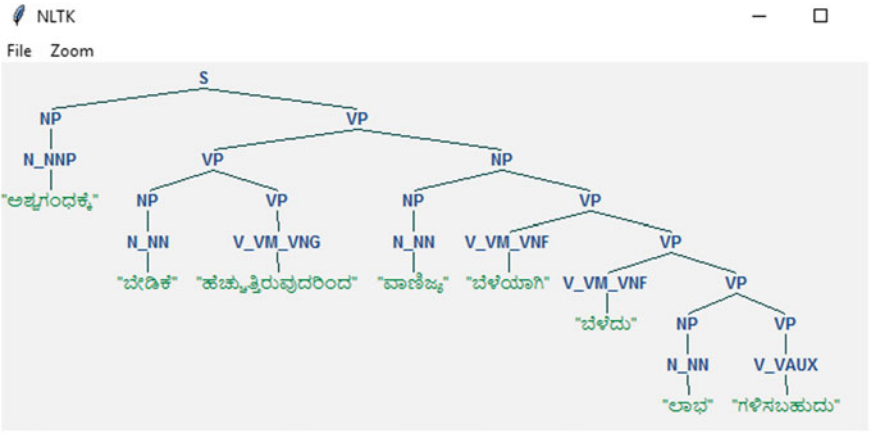
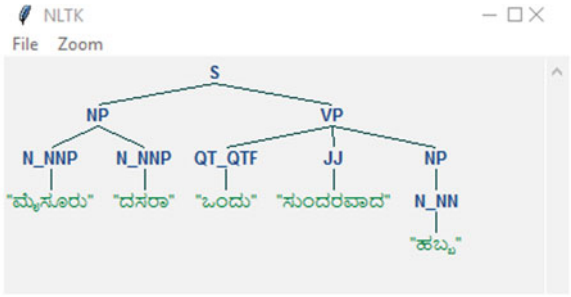


Fig. 5 Parse tree 2 for S5

Fig. 6 Parser tree for S6



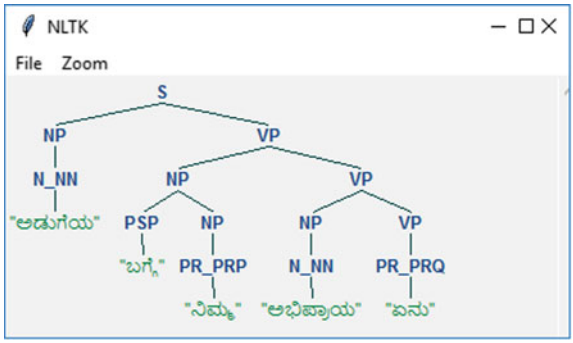


Fig. 7 Parser tree for S7

VP to QT_ QTF JJ NP

This is not in CNF form. But it will be converted into flexible grammar in CNF

$$\text{VP} \rightarrow \text{QT\_ QTF} + \text{JJ NP}$$
$$\text{QT} + \text{QTF} + \text{JJ} \rightarrow \text{QT\_ QTF JJ}$$

Sentence having question words (interrogative sentence): Our parser model is able to parse a text which is having question, and it is shown in Fig. 7.

For example: ಅಡುಗೆಯ ಬಗ್ಗೆ ನಿಮ್ಮ ಅಭಿಪ್ರಾಯ ಏನು? (What is your opinion about cooking?).

ಅಡುಗೆಯ ಬಗ್ಗೆ ನಿಮ್ಮ ಅಭಿಪ್ರಾಯ ಏನು ? (S7)

The NLTK parse tree for the sentence S7 is shown in Fig. 7.

Compound sentence having conjunctions is given in S8:

ಇಂಥವುಗಳಿಂದ ಮಕ್ಕಳ ಬಾಯಿಗೆ ರುಚಿ ಸಿಗಬಹುದು *ಆದರೆ* ಆರೋಗ್ಯದ ಬಗ್ಗೆಯೂ ಕಾಳಜಿ ವಹಿಸುವುದು ನಮ್ಮ ಕರ್ತವ್ಯ (S8)

The meaning of a sentence S8 in English is "Children might like these foods tasty but we need to take care of their health also".

The NLTK parse tree for the sentence S8 is shown in Fig. 8.

The appropriate results of our implemented work and typical examples have been presented in the paper to depict the output of a parser model.

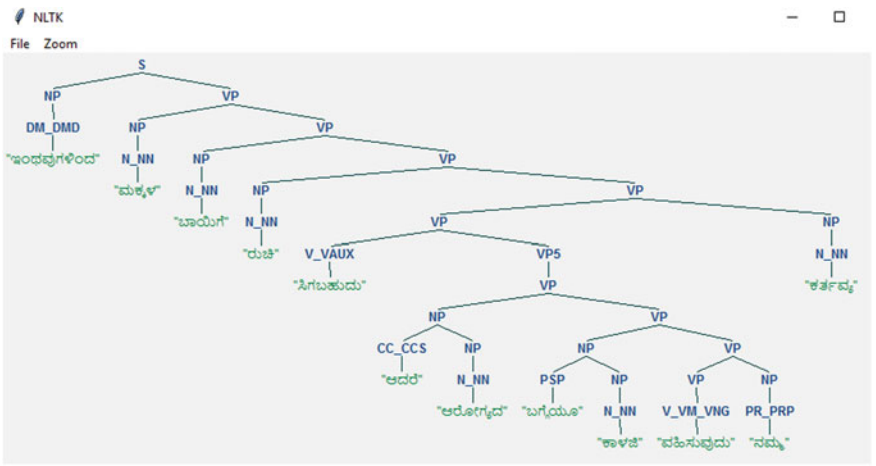


Fig. 8 Parse tree for S8

6 Conclusion

The syntactic parser model on Kannada language using CYK algorithm has been developed and presented in this paper. It is an efficient dynamic programming technique and suitable for analyzing complex Kannada sentences. In the proposed work, 1000 sentences have been given as input to our model. CYK technique follows bottom-up approach, and it is less prone to errors. Very little work has been carried out so far on developing syntax parser model for South Indian languages. We have made an attempt to parse all possible types of Kannada sentences efficiently using the proposed methodology.

References

1. https://tdil.meity.gov.in/
2. https://www.digitalxplore.org/up_proc/pdf/55-139590032413-17.pdf
3. Harish BS, Rangan RK (2020) A comprehensive survey on Indian regional language processing. SN Appl Sci 2:1204. https://doi.org/10.1007/s42452-020-2983-x
4. Hambir N, Srivastav A (2012) Hindi Parser - based on CYK algorithm. Int J Comput Technol Appl 3(2):851–853
5. Prathibha RJ, Padma MC (2019) Design of syntax analyzer for Kannada sentences using rule-based approach, pp 227–283. https://doi.org/10.1007/978-981-13-5802-9_26
6. Pandey V, Padmavati MV, Kumar R (2018) Parsing Chhattisgarhi languages using CYK Algorithm and grammar rules. Int J Pure Appl Math 119(7):77–84. 1311-8080 (printed version); ISSN: 1314-3395 (on-line version)
7. Monika T, Deepak C (2015) Survey: natural language parsing for Indian Languages. arXiv: 1501.07005

8. Sagar BM, Shobha G, Ramakanth Kumar P (2010) Context free grammar analysis for simple Kannada sentences. In: Proceedings of the ınternational conference [ACCTA-2010] on Special Issue of IJCCT
9. Kulkarni D (2014) Specifying context free grammar for Marathi sentences. Int J Comput Appl 99(14):38–41. https://doi.org/10.5120/17444-8285
10. Tayal MA, Raghuwanshi MM, Malik L (2014) Syntax parsing: ımplementation using grammar-rules for English language. In: Proceedings of the 2014 international conference on electronic systems, signal processing and computing technologies, Jan 2014. https://doi.org/10.1109/ICESC.2014.71
11. Prajapati M, Yajnik A (2018) Parsing for Indian languages: a literature survey. Int J Comput Sci Eng 6(8). E-ISSN:2347-2693. https://doi.org/10.26438/ijcse/v6i8.10091018
12. Sharipbay A et al (2019) Syntax parsing model of Kazakh simple sentences. In: DATA '19: Proceedings of the second ınternational conference on data science, E-learning and ınformation systems, Dec 2019. Article No. 54, pp 1–5. https://doi.org/10.1145/3368691.3368745
13. Chatterjee N, Goyal S (2007) An example based approach for parsing natural language sentences. In: International conference on computing: theory and applications (ICCTA'07), pp 451–457. https://doi.org/10.1109/ICCTA.2007.28
14. Das BR, Singh D, Bhoi PC, Priyadarshini P (2020) Parsing for Natural Language in Odia: a novel study. In: 2020 International conference on computer science, engineering and applications (ICCSEA), pp 1–4. https://doi.org/10.1109/ICCSEA49143.2020.9132964