A Rule-based Dependency Parser for Telugu

A thesis submitted to the University of Hyderabad
for the award of the degree of

Doctor of Philosophy
in
Applied Linguistics

by

P. Sangeetha

Reg. No: 18HAPH01

Supervisor

Dr. K. Parameswari

Centre for Applied Linguistics and Translation Studies
School of Humanities, University of Hyderabad, Hyderabad, India
December, 2022

Center for Applied Linguistics and Translation Studies
School of Humanities
University of Hyderabad

CERTIFICATE

Dated - 30/12/2022

This is to certify that **P. Sangeetha** has carried out the research-work embodied in the present thesis entitled **"A Rule-based Dependency Parser for Telugu"** at the Centre for Applied Linguistics and Translation Studies, University of Hyderabad. The thesis represents her independent work and has not been submitted for any research degree of this university or any other university.
This thesis is free from plagiarism and has not been submitted in part or in full to this University or any other university or institution for the award of any degree or diploma.

The following papers were published during this period:

1. P. Sangeetha, K. Parameswari, Amba Kulkarni. 2021. *A Rule-Based Dependency Parser for Telugu - An Experiment with Simple Sentences.* In 'Translation Today'. Vol-15, Issue-1. Pg-[124-144]. ISSN-0972-8740, e-ISSN-0972-8090 [UGC-CARE].

2. P. Sangeetha. 2021. *yaMtrānuvādaMlō vākyaviślesana akkara* (Importance of parser in machine translation). In the journal '*ammanuḍi*'. Pg-[16-18]. ISSN NO: 2582-8738 (UGC-CARE)

3. P. Sangeetha, K. Parameswari, Amba Kulkarni.2021. *Parsing subordinate constructions in Telugu using rule- based dependency parser.* In Proceedings of ICON 2021. Association of Computational Linguistics Indexed.

Further, the student has passed the following courses towards the fulfilment of the coursework requirement for Ph.D.:

Course	**Course Name**	**Credits**	**Pass/Fail**
AL801	Research Methodology	4.00	Pass
AL821	Readings in Applied Linguistics(Research Oriented)	4.00	Pass
AL824	Advanced Topics in core Linguistics	4.00	Pass

Dr. K. Parameswari
Supervisor
Centre for Applied Linguistics and Translation Studies
School of Humanities

Head of the Department
CALTS

Dean
School of Humanities
University of Hyderabad

.

Center for Applied Linguistics and Translation Studies
School of Humanities
University of Hyderabad

DECLARATION

I hereby declare that the work embodied in this thesis entitled **"A Rule-based Dependency Parsing for Telugu"** is carried out by me under the supervision of Dr.K.Parameswari, Centre for Applied Linguistics and Translation Studies, University of Hyderabad, Hyderabad, and has not been submitted for any degree in part or in full to this university or any other university. I hereby agree that my thesis can be deposited in Shodhganga/INFILBNET.

A report of plagiarism statistics from the Indira Gandhi Memorial Library, University of Hyderabad is enclosed.

P.Sangeetha
18HAPH01
Dated:

Dr. K. Parameswari
Supervisor
Centre for Applied Linguistics and Translation Studies
School of Humanities

Acknowledgement

When I started out, I never thought Ph.D. is a big deal. I was full of enthusiasm and zeal to work. However, as time passed, it seemed quite daunting. If not for my supervisor, Dr. Parameswari Krishnamurthy, I would never be able to finish this. Over a span of 5 years, I realised, she is my biggest strength and my biggest cheerleader. I did not come across a person who re-innovates oneself every day and is always taking up new challenges. Beginning from choosing the topic of research to correcting my final draft, she taught me immensely. I will always be grateful for things (both professional and personal), she taught me on a daily basis.

Secondly, Prof. Amba Kulkarni has been one of the driving forces for me to finish this thesis. I thank her for her insights in choosing the topic of my research, for hosting multiple meetings to discuss my research progress, for readily lending a helping hand whenever in need. I will never fail to be in awe of Prof. Kulkarni's selfless nature, hard work and perseverance.

I express my deepest gratitude to Prof. Uma Maheshwara Rao for constanly checking on my research progress and encouraging me to give my best. I am thankful for his valuable feedback on the final draft of my thesis.

I thank Prof. K. Rajyarama for her insightful comments during my RAC meetings. I thank her for all the support she rendered as my RAC member. I thank Dr. Gracious Mary Temsen for always believing in me right from my IMA days. I am immensely thankful the department of CALTS and the faculty of CALTS, Prof. Bhimrao Bhosale, Prof. Arulmozi, Prof. Shivarama Padikkal, Prof. Panchanan Mohanty, Prof. Subramaniam, Dr. Deepak Morey Tryambak (who used to always address me as Dr.Sangeetha even during my MA days), Prof. Prabhakar Rao and Dr. Sriparna.

I thank the non-teaching staffs Mr. Murthy, Mr. Mallesh, Ms. Chandrakala, Ms. Anuradha and Ms.Swati for their cooperation and support all through my Ph.D. days.

I express my deepest gratitude to University Grants Commission for their financial support through UGC-JRF for 5yrs of my Ph.D. I thank the fellowship section, UoH for their timely disbursement of the fellowship.

I am indebted to my dear friends from the University, Dr. Nivea Thomas (for being my biggest moral support), Prakash *anna*(my confidant), Madhukar (for all the selfless support and discussions on my drafts), Sowjanya (for cheering me up during tough days), Dr. Anakha Ajith (for always encouraging me to stay positive during this journey), Bhavani (for being my biggest cheerleader), Albin Rico Xalxo, Sushmita, Prabhakar, Keerthana.B, Koyyada Mahender, Lathasri, Ramya, Jaita Venu, Naveen Gadicharla, Shreelaxmi, Pranav, Gouthama, Bapuji *anna*, Sreenu for directly or indirectly helping me. My special thanks to Bhagya for staying with me through thick and thin.

Special thanks to my friends from Sanskrit department, Saee, Amrutha, Shriram and Malay. I render my special thanks to Sanal Vikram for all his technical help.

I thankfully acknowledge LTRC, IIIT-H for their support directly and indirectly.

I thank the cutest twins *paṭṭu* and *ciṭṭu* for bearing with us while we had long discussions with their *ammā*. I extend my thanks to Mr. Sakthivel for his kindness and generosity during our untimely visits.

I thank my father for always believing in me. I thank my brother, *pratāp*, for being there always. And my sister, *surēkha* for having faith in everything I do. My special thanks and love to *bangāraM*, my little niece who showered immense love and lifted up my spirits during difficult times. I am indebted to my fiance, *Ashok* and his family who always pushed me to do better work.

This thesis is as much my *amma*'s as it is mine. I owe a lot to her for going through this journey along with me and kept me sane all this while. If there is one person I would like to dedicate this thesis to, it should be to her, for I know you will be the happiest to witness your daughter become Dr. Sangeetha.

Transliteration Schema

Roman	*a*	*ā*	*i*	*ī*	*u*	*ū*	*ṛ*	*ṝ*	*e*	*ē*	*ai*	*o*	*ō*	*au*	~	*M*	*H*
WX	a	A	i	I	u	U	q	Q	eV	e	E	oV	o	O	z	M	H
Telugu	అ	ఆ	ఇ	ఈ	ఉ	ఊ	ఋ	ౠ	ఎ	ఏ	ఐ	ఒ	ఓ	ఔ	ఀ	ం	ః

Roman	*ka*	*kha*	*ga*	*gha*	*ṅa*	*ca*	*cha*	*ja*	*jha*	*ña*	*ṭa*	*ṭha*	*ḍa*	*ḍha*	*ṇa*
WX	ka	Ka	ga	Ga	fa	ca	Ca	ja	Ja	Fa	ta	Ta	da	Da	Na
Telugu	క	ఖ	గ	ఘ	ఙ	చ	ఛ	జ	ఝ	ఞ	ట	ఠ	డ	ఢ	ణ

Roman	*ta*	*tha*	*da*	*dha*	*na*	*pa*	*pha*	*ba*	*bha*	*ma*	*ya*	*ra*	*ṟa*	*la*
WX	wa	Wa	Xa	Xa	na	pa	Pa	ba	Ba	ma	ya	ra	rYa	la
Telugu	త	థ	ద	ధ	న	ప	ఫ	బ	భ	మ	య	ర	ఱ	ల

Roman	*ḷa*	*va*	*śa*	*ṣa*	*sa*	*ha*
WX	lYa	va	Sa	Ra	sa	ha
Telugu	ళ	వ	శ	ష	స	హ

Abbreviations

ACC	Accusative
ADJ	Adjectivalizer
ADV	Adverbializer
AGR	Agreement
ASS	Associative
AUX	Auxiliary
CAT	Category
CAUS	Causative Marker
CCG	Combinatory Categorical Grammar
CM	Case Marker
COMP	Complementizer
CONC	Concessive
COND	Conditional
CONJ	Conjunction
COP	Copula
CP	Conjunctive Participial
CRF	Conditional Random Fields
DAT	Dative
DG	Dependency Grammar
DOM	Differential Object Marking
DUR	Durative
DUB	Dubitative
EMPH	Emphatic
EXCL	Exclusive
F	Feminine
FUT	Future Tense
GEN	Genitive
GERUN	Gerund
GNP	Gender-Number-Person
GPSG	Generalised Phrase Structure Grammar
GUI	Graphical User Interface
HAB	Habitual
HON	Honorific
HORT	Hortative
HPSG	Head-Driven Phrase Structure Grammar
HyDT	Hyderabad Dependency Treebank
IMP	Imperative
INCL	Inclusive
INF	Infinitive
INS	Instrumental
IIIT-H	International Institute of Information Technology-Hyderabad
LAS	Labelled Attachment Score
LDC	Linguistic Data Consortium

LFG	Lexical Functional Grammar
LA	Labelled Accuracy
LOC	Locative
LSP	Lexicalized and Statistical Parsing
M	Masculine
MALT	Models and Algorithms for Language Technology
MT	Machine Translation
MWE	Multi-Word Expressions
N	Neuter
NEG	Negative
NLP	Natural Language Processing
NLU	Natural Language Understanding
NOM	Nominative
NP	Noun Phrase
NST	Nouns of Space and Time
NUM	Number
NV	Noun-Verb
OBL	Oblique
PASS	Passive
PCFG	Probabilistic Context-Free Grammar
PL	Plural
POS	Parts Of Speech
POSS	Possessive
PS	Phrase Structure
PSP	Postposition
PST	Past Tense
PG	Pāṇinian Grammar
QUOT	Quotative
RBP	Rule-based Parser
REDUP	Reduplication
RP	Relative Participle
SG	Singular
SOV	Subject Object Verb
STAT	Stative
SVM	Support Vector Method
TAG	Tree Adjoining Grammar
TDG	Tesniere's Dependency Grammar
TEMP	Temporal
UD	Universal Dependencies
UAS	Unlabelled Attachment Score
VREC	Verbal Reciprocal
VREF	Verbal Reflexive

Abstract

Parsing natural languages has been gaining popularity in recent years and attracted the interest of Natural Language Processing (NLP) researchers around the world. It is challenging when the language under study is a free-word order language and morphologically rich like Telugu, the south-central Dravidian language. Parsing refers to the process of syntactic analysis of a specific language text. A parser is an automated tool that dissects sentences to provide syntactic/syntactico-semantic analysis of relations of words in a sentence. Parsing is useful in the downstream analysis and applications of NLP such as machine translation, document classification, dialogue modelling, etc..,

This study adopts a knowledge-driven approach, i.e. a rule-based technique for building parser for Telugu using linguistic cues as rules. The present research adopts the Indian grammatical tradition i.e. Pāṇini's Grammatical (PG) tradition as the dependency model to parse sentences. A detailed description of mapping semantic relations to vibhaktis (case suffixes and postpositions) using linguistic cues in Telugu is presented.

An enhanced annotation scheme for Telugu dependency relations is introduced. Challenges faced in parsing ambiguous structures are elaborated alongside providing enhanced tags to handle them. The study describes the parsing algorithm and the linguistic knowledge employed while developing the parser. The research further provides results, which suggest that enriching the current parser with linguistic inputs can increase the accuracy and tackle ambiguity better than existing data-driven methods. Results are encouraging and this parser proves to be efficient for languages like Telugu which can be later extended to other morphologically-rich languages.

Contents

List of Tables

Chapter 1

Introduction

1.1 What is Parsing?

Parsing is a process of syntactic analysis of a specific language text. A parser is an automated tool that dissects sentences to provide syntactic/syntactico-semantic/morpho-syntactic analysis of the relations of words in a sentence. Parsing is useful in the downstream analysis and applications of Natural Language Processing (NLP) such as Machine Translation (MT), Document Classification, and Dialogue Modelling. Also, parsing is a significant module in Natural Language Understanding (NLU) to capture the structure of a sentence, thereby, understanding the meaning of the text. Parsing is an interesting yet challenging task as it involves resolving language ambiguities like the attachment and the scope ambiguities [1]. The present research aims to build a high-quality parser for Telugu using a rule-based method adopting dependency grammar formalism.

Parsing, a word derived from Latin *pars orations* which means 'parts of speech' was used in elementary schools for grammatical explication of sentences (Nivre, 2006). Over the years, word parsing acquired a specific definition in NLP. Bunt et al. (2005) defines parsing as "the decomposition of complex structures into their constituent parts and parsing technology as the methods, the tools and the software to parse automatically". Alternatively, a sentence [S] is said to be parsed if one or all of its derivative trees are known (Aho and Ullman, 1972). For instance, if a grammar[G] and a sentence[S] are provided, the parsing problem provides the answers for: (1) whether or not a sentence(s) $\in$ L(G). (2) If s $\in$ L(G), the answer to this question may be either a parse tree or a derivation (Jeuring and Swierstra, 2001). Let us comprehend the phenomenon of parsing using examples from English and Telugu. Consider example (1.1) and its structural representation in the dependency framework. In figure (1.14), a linear structure of words is converted into a hierarchical structure that represents relations between words (Kulkarni, 2021a).

[1]The classic examples of attachment and scope ambiguities include 'I saw the elephant with binoculars' and 'The old men and women' respectively

(1.1) John saw Mary

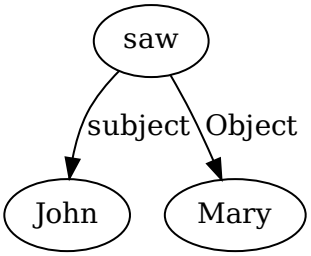


Figure 1.1: Structural Representation for (1.1)

In figure (1.14), the higher node 'saw' is the root or head node which dominates the other two nodes 'John', and 'Mary'. 'John' is related to the root node as the subject and 'Mary' as the object. This information about the subject and object in an English sentence is elicited using the position of words. However, this is not true of all languages, especially for free-order languages like Telugu. Other syntactic cues must be used to parse sentences for such languages. Consider (1.2)

(1.2) *nēnu* *sinimā* *cūs-ā-nu*
I.NOM cinema see-PST-1.SG.
'I saw a movie'

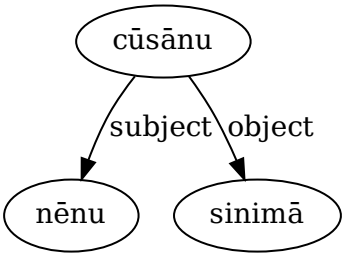


Figure 1.2: Structural Representation for (1.2)

In the example (1.2), the root node *cūs-ā-nu* 'saw' has *nēnu* 'I', *sinimā* 'cinema' as the subject and object respectively. But the position of words does not determine the syntactic role of words in free word-order languages like Telugu. This calls out for a different identification strategy to elicit such structural information. Here, we consider agreement and ontological properties of nouns (animacy in specific) to

parse the Telugu sentence(1.2) which is discussed in detail through the course of this thesis.

Each language requires different linguistic cues like position, agreement, case, ontological features etc. to parse sentences. In case of Telugu, major syntactic information is coded in words in the form of suffixes. This study exploits morpho-syntactic features of Telugu to show how simple linguistic cues can ease the parsing process and cut out the cumbersome task of annotating huge corpus for building treebanks. Treebank, a term coined by Geoffrey Leech (Brdar et al., 2003, cf.) refers to a repository of annotated corpus with syntactically/syntactico-semantically labelled trees based on grammatical analysis of languages. This study, instead of building treebanks, creates linguistic approximation of Telugu syntax to build a parser.

1.2 Aim and Scope of the Study

This study attempts to build an efficient parser for Telugu in the dependency framework using a rule-based method. In the process of building the aforementioned parser, this thesis envisages the following aims and scopes:

- To critically examine the existing grammar formalisms and choose an appropriate framework that rightly captures the language information under study
- To explore and understand various parsing techniques used for languages around the world, thereby, enabling the suitable technique selection for Telugu parsing
- To understand the syntactic properties of Telugu from a parsing perspective, in order to encode syntactic information in the form of rules to automate the parsing process
- To review and compare the existing tagsets available for labelling dependency relations subsequently enhancing or building new guidelines.
- To integrate linguistic cues as rules for each dependency relation and develop databases wherever necessary so as to build the rule-based parser
- To collect a corpus comprising of various sentences available in Telugu for the evaluation

- To evaluate and analyse the output of the parser to find efficient ways to improve the accuracy

Eventually, the Telugu parser can be plugged in NLP applications such as Machine Translation (MT,henceforth) to validate its contribution to NLU.

1.3 A Brief Note on Telugu

Telugu is a south-central Dravidian language (Krishnamurti, 2003a, p.19) spoken in the states of Telangana and Andhra Pradesh. According to the census of India 2011, Telugu is the fourth most spoken language of India with approximately 8,11,27,740 speakers which constitute 6.7% of total population (India, 2011). Telugu has four regional dialects [1], viz. i). Northern dialect of Telugu popularly called the Telangana dialect spoken by the people of Telangana state (formed as a separate state in 2014), ii). Southern dialect of Telugu popularly known as the Rayalaseema dialect spoken in 4 districts (Kurnool, Kadapa, Anantapur, Chitoor) including 2 coastal districts (Nellore and Prakasm), iii). Eastern dialect of Telugu is spoken by northeast districts (Vishakapatnam, Vijayanagaram, Srikakulam), iv). Central dialect of Telugu popularly known as the standard Telugu dialect spoken in central districts (Krishna, Guntur, East and West Godavari). In this study, we focus on building a parser for written variety of Telugu which is available widely in Central dialect of Telugu.

Telugu is a head-final and left-branching language with Subject Object Verb (SOV) word order. Though the verb is a central component of sentences, verbless sentences are quite prevalent in Telugu. A sentence can be headed by a verb (verbal predicate) or a nominal (nominal predicate). Telugu is a nominative-accusative language. Simple sentences in Telugu consist of nominal arguments/adjuncts and a verbal or nominal predicate. Complex sentences include a subordinate clause (usually non-finite clauses) and a matrix clause. Compound or co-ordinate sentences are also constructed through morphological inflections in Telugu and a 'the part of function of compound sentences has been taken over by participial clauses' (Ramarao, 2017). Arguments of the verb are nominal entities wherein subject argument is expressed using nominative/ non-nominative case. Some structural features of Telugu are listed below:

[1]The classification of dialects and the districts discussed here is before 2014 when Telangana and Andhra Pradesh are part of a single state. However, Andhra Pradesh and Telangana expanded the number of districts now which are not included here

- Telugu is an agglutinative language which stores linguistic information in the form of affixes. Affixes in agglutinative languages are glued together to represent complex linguistic features. Eg. *pāḍ-iMcu-konn-ā-ḍu* 'sing-CAUS-REF-PST-3.SG.M 'he made(someone) to sing'

- Telugu is a pro-drop language that allows subject-less sentences. Pro-drop languages allow pro-drop to an extent that the φ - features (gender, number, person, etc) are reflected on the verb for the local recovery of the dropped arguments (Biberauer, 2008, p.331). In Telugu, the verbs often carry gender, number and person agreement with the subject. As the subject information is encoded on the verb, subject can often be dropped. Example - *annaM tiMṭunnāḍu* - 'He is eating food'.

- The copula verb is almost absent in equative affirmative constructions in Telugu. Equative constructions with the absence of copula is realised with either nominal or adjectival predicates. Eg - *nēnu pariśōdaka vidyartini* 'I am a research scholar'

- Case syncretism (Comrie, 1991), a case marker with multiple functions, is prominently available in the Telugu case system. A single case marker serving multiple case functions leads to several analysis in a parser. Eg. *ravi iMṭi**ki** veḷḷāḍu* 'Ravi went home', *ravi**ki** kōpaM vacciMdi* 'Ravi is angry', *amma ravi**ki** annaM peṭṭiMdi* 'Mother served food to Ravi'

- Non-nominative or quirky subjects in Telugu are common in certain domains (Bhaskararao and Subbarao, 2004), where the structure is different from regular nominative constructions. Dative subject constructions, in particular are quite prevalent. Eg. *nāku dabbulu kāvāli* 'I need money'

- Predicates consisting of Noun-Verb (NV) compounds are productive in Telugu. Example: *snānaM cēyu* 'to bath'

- The use of participles in complex construction is the unique feature of Telugu including other Dravidian languages. Participial constructions are used to express conditionality, cause/reason etc. Eg - *ravi hōMvark rāsi baḍiki veḷḷāḍu* 'Ravi went to school after writing his homework'

- Relative clauses occur before noun phrases which they modify, unlike English. Eg- *pāṭa pāḍina sunita vacciMdi* - 'Sunita who sang the song came'

- Multi-token words such as word forms with clitics expressing conjunction, disjunction, questions, dubitativeness and emphasis are part of morphology which require syntactic status in parsing. Eg. *vastāḍō lēdō* - 'whether he will come or not' clitic 'ō' expressing dubitativeness
- Conditional and concessive clauses manifest morphologically as inflection on verbs

As shown above, wordforms in Telugu are rich with grammatical information which can be used as cues while developing a rule-based parser.

1.4 Overview of Parsing

Parsing as a sub-field of NLP had emerged several decades ago. Over these decades, a repository of knowledge on automation of syntactic analysis has been compiled. Nevertheless, Parsing NL text is still a complex NLP task. Parsing open-ended unrestricted NL text is far more complicated when compared to parsing formal grammars. Building robust, efficient and accurate parsers for NLs are still considered a challenge. Parsing is considered a central part in several NLP tasks due to its utility in larger applications like information retrieval, Machine Translation (MT), text summarization, question answering system etc (Clark et al., 2013). Building a parser requires an appropriate grammar formalism, parsing strategy, implementation technique and customised annotation schema (tagset) for the language under study. These criteria are briefly discussed in the further subsections.

1.4.1 Grammar Formalisms

Grammatical understanding of the language and selecting a suitable formalism for syntactic representation are a pre-requisite for parsing. The most commonly used grammatical frameworks for parsing include Constituency and Dependency approaches. However, other grammatical formalisms are also in practice viz. Categorical Grammar (CG), Generalised Phrase Structure Grammar(GPSG), Lexical Functional Grammar(LFG), Tree-Adjoining grammar(TAG), Head-Driven Phrase structure Grammar(HPSG), and several others.

1.4.1.1 Phrase Structure Grammar (PS)

Constituency or phrase structure has arrived as an avalanche to the linguistic community with Noam Chomsky's work (1957, 1965, 1981 & 1995). Since then, constituency has been the prominent linguistic theory both in theoretical and computational linguistics (Nivre, 2006).
Constituents are the prime elements of grammar as assumed by Consituency grammar. 'Constituents are groups of words that function as units with respect grammatical processes' (Carnie, 2008). Osborne (2013) defined constituents as 'node plus all the nodes that that node dominates'. Constituency is a grammatical framework which describes the structure of a sentence in-terms of these constituents. Constituency parse trees provide richer linguistic structures. A constituency sentence tree has terminal and non-terminal nodes. Terminal nodes are the ones which do not have any branches underneath (usually the actual words of a sentence). Non-terminal nodes are all the other nodes other than the terminal nodes. Non-terminal nodes consist of root nodes and intermediate nodes/pre-terminals. These pre-terminal nodes provide the lexical category of words. Due to its constituent structure, it is well-suited for positional languages than free word-order languages. Consider the constituency parse tree of (1.3): [john]NP [went]VP [to the school]PP :

(1.3) 'John went to the school'

1.4.1.2 Dependency Grammar(DG)

Dependency Grammar is one of the theoretical frameworks, like phrase-structure, used to describe a natural language. This can be viewed as one of the methods of language analysis in general and syntactic analysis in particular. Dependency grammar is one of the earliest grammars which is often traced back to Pāṇini, who postulated the grammar of Sanskrit. The term *dependency* was first used by Hays (1964) as reported in Jurafsky (2000).DG is popularly in use in European linguistic tradition with the work of Mel'cuk et al. (1988). Though dependency framework was in use to describe languages, it is with the work of Tesnière (1959) that dependency as a grammatical framework was introduced to analyse languages. Tesniere who devoted major part of his life in teaching French, later, using his experiences in teaching, worked on developing a grammar to describe French and other Slavic languages that he taught. Tesniere's work 'Eléments de syntaxe

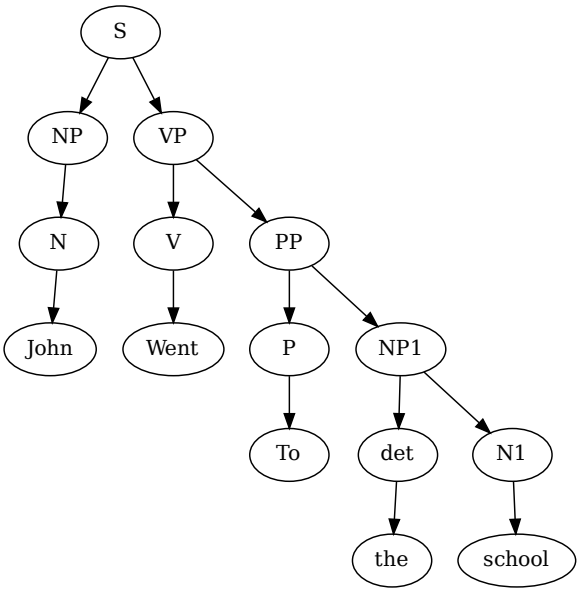


Figure 1.3: Constituent Structure for Ex-1.3

structurale' (1959) originally published in French is much later translated to English by Timothy Osborne & Sylvain Kahane in 2015. Tesniere's work stands as a pioneering work in the history of dependency grammar. Apart from Tesnière (1959), other theories of dependency grammar include Functional Generative Description, Meaning-Text Theory (Mel'cuk et al., 1988), Lexicase, Word Grammar, Constraint Dependency Grammar, Functional Dependency Grammar and several other theories.

Dependency formalism assigns relations to words in a sentence based on modifier(viśeṣaṇa) and modified(viśeṣya) relations. There exists one word which acts as a root of the sentence called the"chief qualificand" (mukhya-viśeṣya) (Kulkarni, 2021a). Dependency formalism is tested and proved to be suitable for free word-order languages. Dependency advocates verb-centrality and rejects the subject-verb distinction. In addition to this, dependency trees are not very complex, they have nodes equal to the number of words in a sentence. Consider the dependency tree of (1.3)

From the figure, it can be seen that dependency tree is a labelled directed acyclic graph. It has a single root node (go). All dependency nodes except the root node has one incoming node.

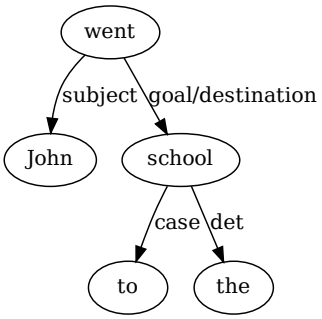


Figure 1.4: Dependency Structure for Ex-1.3

1.4.1.3 Combinatory Categorical Grammar(CCG)

One of the oldest lexicalized grammatical formalisms is said to be the Categorical Grammar(CG) (Ajdukiewicz, 1935) and the extended form of CG is the Combinatory Categorical Grammar (CCG). CCG is a framework developed by Steedman and Baldridge (2011). CCG is a kind of lexicalized grammar which frames syntactic rules solely on the category of the input text given. No syntactic rule is structure or derivation dependent. CCG is different from traditional notions of constituency in that it allows flexible surface structure. The potential constituents are the most continuous substrings of a well-formed sentence. The grammatical/syntactic rules are applied further with compositional semantic interpretation.

Several treebanks were developed using CCG namely, English CCGbank (Hockenmaier and Steedman, 2007), Chinese CCGbank (Tse and Curran, 2012). CCG is also used in improving the existing dependency treebanks using CCG categories for Telugu. (Ambati et al., 2013, 2014).

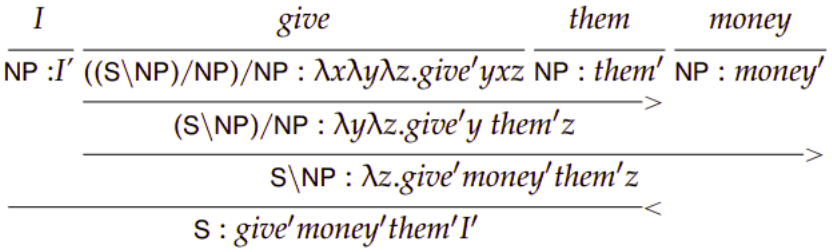


Figure 1.5: Sample CCG representation retrieved from (Hockenmaier and Steedman, 2007)

1.4.1.4 Lexical Functional Grammar(LFG)

LFG is a grammatical theory introduced by Bresnan (1978), a former student of Chomsky who disagreed with Chomsky's idea of transformations. Later, Bresnan collaborated with Kaplan (1972) and developed this grammatical framework called LFG (Kaplan and Bresnan, 1982). LFG is a generative, non-derivational, constraint-based grammar in which lexicon plays a major role (Falk, 2001, pg-9). LFG was applied in parsing the UPenn Wall Street Journal (WSJ) treebank (Riezler et al., 2002) reporting the f-score of 76.1%. Several other works related to parsing were made using LFG (Eisele and Dorre, 1986; Güngördü and Oflazer, 1995; Kim, 1993; Salloum et al., 2016). However, some earlier works (Joshi and Mathur, 2012) stated that they had great difficulty in associating features with constituent structures and its incomprehensibility in implementing it.

1.4.1.5 Generalised Phrase Structure Grammar(GPSG)

GPSG is a variant of context-free phrase structure grammar developed by Gazdar et al. (1985a). This framework was developed to show that natural languages are context-free and can be described using context-free grammatical frameworks.
GPSG is framed on three types of rules (Phillips, 1992), namely, (i) Immediate-Dominance (ID) rules, (ii) Linear Precedence(LP) rules and (iii) metarules. ID rules specify which categories(grammatical constituents) can combine in-order to produce other categories,for example, a Noun Phrase(NP) can have an adjective phrase. LP rules specify the linear order of the constituent elements of a category, for example, an NP precedes a Verb Phrase(VP). metarules state that if a language grammar contains rules that affirm to one specified pattern, it also contains rules that match some other derived pattern (Shieber et al., 1983). All these rules combined together define the syntax of a language. Every node of a tree should conform to these rules in order to form an acceptable syntactic tree.
GPSG is used to implement grammars for various languages like Persian (Bahrani et al., 2011), English (Gawron et al., 1982), French (Bès and Baschung, 1985), and several other languages. GPSG analysis for a sample sentence is provided below:

(1.4) She went to the park

GPSG output- ((NP-she (N)))((VP-went (V)))((PP- to (P))((NP-the (DET) park (N)))

1.4.1.6 Head-Driven Phrase Structure Grammar(HPSG)

Head Driven Phrase Structure Grammar(HSPG) is a work developed by Pollard and Sag (1987). HPSG has two main components namely, (1) representation of grammatical categories(words) using feature structures and (2) a description of constraints stating linguistic generalizations of a language. These constraints usually include a lexicon, lexical rules of derived words, ID of words in a sentence, LP. Any given sentence is grammatical if and only if it abides by the rules of these two components and the constraints (Levine and Meurers, 2006).
HPSG is proved to be useful in understanding various phenomena in languages like Slavic, Romance, German and several other languages (Levine and Meurers, 2006). Among Indian languages, HPSG is implemented for Bangla (Khan and Khan, 2006), Hindi (Goyal et al., 2003), etc.

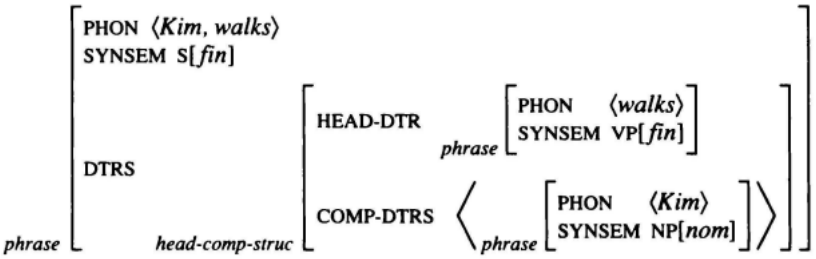


Figure 1.6: Sample HPSG representation retrieved from (Pollard and Sag, 1994b)

1.4.1.7 Tree Adjoining Grammar(TAG)

Tree-adjoining grammar was developed by Joshi et al. (1975) initially started as Tree adjunct grammar. This grammar is an attempt to prove that formal mathematical grammars can still be used to describe natural languages (Kroch and Joshi, 1985). TAG is defined as 'both weakly and strongly equivalent to Grammar'. Let G = (1,A) where I and A are finite sets of elementary trees. The trees in 'I' will be called the initial trees and the trees in 'A', the auxiliary trees" (Kroch and Joshi, 1985). TAG is used for English (Group et al., 1998), Hindi (Jain et al., 2018), Tamil (Menon et al., 2016) and several other languages.

1.4.2 Methods of Parsing

Based on an appropriate grammar formalism, an implementation technique is adopted to parse the given language text. There are various methods of parsing

(Nivre, 2006, p.20) that has been in practice for several decades, among four prominent methods (Nivre, 2006) are discussed in detail in this section :

1. Grammar-Driven Parsing
2. Data-Driven Parsing
3. Hybrid Parsing
4. Neural Network based Parsing

1.4.2.1 Grammar-Driven Parsing

Grammar-driven approach also known as rule-based parsing is one of the widely used parsing methods. In grammar-driven parsing, it is assumed that a natural language can be to an extent approximated to a formal language (Nivre, 2006). A given language's grammar is analysed to form generalisations which in-turn are converted to formal language rules. Based on these formal language rules, a language text is parsed. (Baud et al., 1999), (Haverinen et al., 2009), (Ramasamy and Žabokrtskỳ, 2011), (Anchiêta and Pardo, 2018), and several others adopted a grammar-driven technique for parsing and stressed its importance in parsing natural language texts.
However, it is a known fact that all of natural language content cannot be approximated to a formal language due to the novelty and richness of natural languages. This stands as the biggest drawback of grammar-driven approaches. Problems of robustness, coverage, over/under generation of parsing analysis can be few issues related to grammar-driven parsing approach.

1.4.2.2 Data-Driven Parsing

In data-driven parsing, parsers are build by employing treebank grammars. Treebanks are a collection of correctly/manually parsed sentences of a given language. Data-driven parsing is also called statistical parsing. Probabilistic Context-Free Grammar (PCFG) is the commonly used grammar formalism in building statistical parsers. Any sentence provided as an input is considered as a valid grammatical sentence and is attempted to be parsed. Data-driven dependency parsing is sub-classified into two types: transition-based dependency parsing and graph-based dependency parsing. Models and Algorithms for Language Technology (MALT) is considered the best example for statistical parser. Like other parsers, statistical parser also has some disadvantages like lack of lexical

conditioning and poor independence assumptions which can be improved by annotating larger data (Jurafsky, 2000).

1.4.2.3 Hybrid Parsing

Hybrid parsing is a combination of both grammar-driven and probabilistic methods. In this type of parsing, a treebank is required with correctly parsed sentences and also a set of grammatical rules for further disambiguation. This proved to be an effective kind of parsing as it provides multi-layered filtering and analysis of the text. Consider fig-1.7 that shows the convergence of grammar and data-driven methods and the approaches that adopt them.

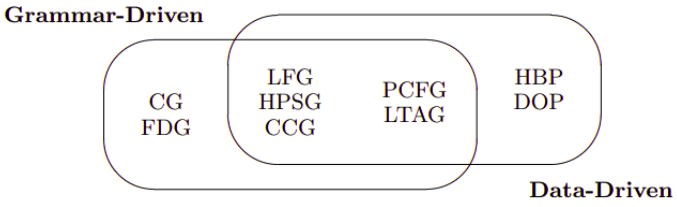


Figure 1.7: Convergence of Grammar and data-driven parsing proposed by (Nivre, 2006)

1.4.2.4 Neural Network based Parsing

In his seminal paper, McCulloch and Pitts (1943) introduced a simple mathematical model of a single neuron. Using the neuron analogy, they proved that neural networks can be used for universal computing. Neural network based parsers are a new development and is widely used currently. Neural network is a paradigm that processes information which consists of numerous neurons working towards solving a specific problem."An artificial neural network (or simply neural network) consists of an input layer of neurons (or nodes, units), one or two (or even three) hidden layers of neurons, and a final layer of output neurons" (Wang, 2003). Neural networks work in transition-based and graph-based methods for parsing purposes. Chen and Manning (2014) developed a neural network based schema for depending parser purposes. Consider the figure for the schema proposed by (Chen and Manning, 2014):

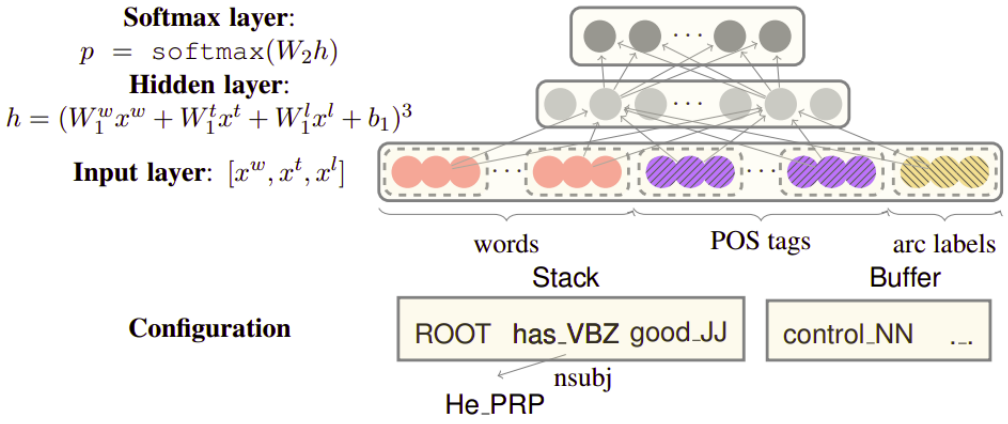


Figure 1.8: Architecture of neural network schema proposed by Chen and Manning (2014)

1.4.3 Parsing Strategies

Most parsers have underlying parsing strategies namely Top-down or goal-oriented search and Bottom up or data-directed search strategies (Jurafsky, 2000).

1.4.3.1 Top-Down or Goal-Oriented

Top-down or goal-oriented parsing strategy builds parse trees from the top root node or S node to leaf nodes. Starting from the root node, trees are build downward till they reach the parts of speech categories at the bottom. In this kind of parsing, all the trees lead to the root node, hence not wasting time in generating non-sentences. Nevertheless, analysing output which is inconsistent with the input can be considered as a disadvantage.

1.4.3.2 Bottom-up or Data-Directed

This parsing strategy builds trees starting from leaves and projecting to root node. Here, the words from the lexicon are first considered and looked for any ambiguous interpretations that affect the tree. Later, based on the grammar rules each step of the tree is built. This strategy ensures that the trees are in connection with the input. The disadvantage of this method is that it cannot ensure that all bottom-up trees lead to the root node.
The present study follows a bottom-up strategy to provide the syntactico-semantic relations among wordforms in a given input.

1.4.4 Review of Annotation Schema

Annotation schema refers to guidelines using which parsing relations are labelled. Guidelines ensure a uniform pattern across languages. Various annotation guidelines have been proposed by several researchers based on the syntactic features of particular languages. Here, we review the popular tagsets in use. Tagsets which we discuss include Penn tagset, Stanford tagset, Universal Dependencies tagset, Anncora tagset and *samsādhani* tagset.

1.4.4.1 Penn Treebank Syntactic Tagset

Penn tagset [1] is proposed as bracketing guidelines for treebanks under Penn Treebank Project by Linguistic Data Consortium(LDC), University of Pennsylvania (Bies et al., 1995). Since 1989, Penn treebank project produced a repository of around 7M words of POS tagged text, 3M words of parsed corpus, approximately 2 million words of text parsed for predicate-argument structure, and 1.6 million words of transcribed speech text annotated for speech disfluencies. The annotated material includes genres such as nursing notes, IBM computer manuals, transcribed telephone conversations, Wall Street Journal articles among others(Taylor et al., 2003). The penn treebank syntactic tagset is provided below:

ADJP	Adjective phrase
ADVP	Adverb phrase
NP	Noun phrase
PP	Prepositional phrase
S	Simple declarative clause
SBAR	Subordinate clause
SBARQ	Direct question introduced by *wh*-element
SINV	Declarative sentence with subject-aux inversion
SQ	Yes/no questions and subconstituent of SBARQ excluding *wh*-element
VP	Verb phrase
WHADVP	Wh-adverb phrase
WHNP	Wh-noun phrase
WHPP	Wh-prepositional phrase
X	Constituent of unknown or uncertain category
*	"Understood" subject of infinitive or imperative
0	Zero variant of *that* in subordinate clauses
T	Trace of wh-Constituent

Figure 1.9: Penn tagset retrieved from Taylor et al. (2003)

[1] https://www.ldc.upenn.edu/

1.4.4.2 Stanford Dependency Tagset

Stanford dependency tagset [1] originally developed for English began in 2005. A group of linguistic researchers attempted to develop a 'linguistically sound, surface-syntax oriented dependency representation' for English which progressed as Stanford dependencies till the development of Universal dependencies. Stanford dependency tagset contains around 50 grammatical relations which are binary relations that hold between a governor and the dependent (De Marneffe and Manning, 2008). Stanford dependencies is also available for Chinese, Italian, Bulgarian and several other languages.

root - root
dep - dependent
    aux - auxiliary
        auxpass - passive auxiliary
        cop - copula
    arg - argument
        agent - agent
        comp - complement
            acomp - adjectival complement
            ccomp - clausal complement with internal subject
            xcomp - clausal complement with external subject
            obj - object
                dobj - direct object
                iobj - indirect object
                pobj - object of preposition
        subj - subject
            nsubj - nominal subject
                nsubjpass - passive nominal subject
            csubj - clausal subject
                csubjpass - passive clausal subject
    cc - coordination
    conj - conjunct
    expl - expletive (expletive "there")
    mod - modifier
        amod - adjectival modifier
        appos - appositional modifier

Figure 1.10: sample of syntactic tags of Stanford tagset in hierarchy retrieved from (De Marneffe and Manning, 2008)

[1] https://nlp.stanford.edu/software/stanford-dependencies.shtml

1.4.4.3 Chinese Dependency Tagset

Chinese Dependency Treebank [1] 1.0 was proposed by the Harbin Institute of Technologies Research Center for Social Computing and Information Retrieval (HIT-SCIR) in 2012. It contains 49,996 (902,191 words) Chinese sentences annotated with syntactic dependency structures selected randomly from *Peoples Daily* news wire stories published from 1992 to 1996. Currently, Chinese Treebank 8.0 consists of approximately 1.5 million words of corpus annotated for parsing purposes. Liu and Huang (2006) states that Chinese dependency tagset contains around 34 dependency tags and 13 word classes.

1.4.4.4 Universal Dependencies Tagset

Universal Dependencies (UD) [2], is a platform initiated to facilitate cross-linguistic morphosyntactic treebank annotation. UD is originally stanford dependencies which evolved to accommodate world languages. UD annotation schema has 37 universal dependency tags and around 198 language-specific tags which can be used whenever necessary. UD is fast-growing in terms of number of treebanks. UD currently contains 200 treebanks contributed by 300 language specialists for around 100 languages. Consider fig-1.13 to see the classification of syntactic relations in UD.

	Nominals	Clauses	Modifier words	Function Words
Core arguments	nsubj obj iobj	csubj ccomp xcomp		
Non-core dependents	obl vocative expl dislocated	advcl	advmod* discourse	aux cop mark
Nominal dependents	nmod appos nummod	acl	amod	det clf case
Coordination	**MWE**	**Loose**	**Special**	**Other**
conj cc	fixed flat compound	list parataxis	orphan goeswith reparandum	punct root dep

Figure 1.11: Syntactic relations of Universal dependencies (retrieved from https://universaldependencies.org/u/dep/index.html)

[1] https://catalog.ldc.upenn.edu/LDC2012T05

[2] https://universaldependencies.org/guidelines.html

1.4.4.5 Anncora Tagset

Based on these syntactico-semantic relations, (Bharati et al., 2012) has developed a dependency tagset known as Anncora tagset which can be used for almost all major Indian languages. Anncora, an abbreviated form of 'annotated corpora' is an initiative taken up by IIIT-H[1] as part of 'Workshop on Lexical Resources for Natural Language Processing' to develop linguistic resources for Indian language in electronic form. As part of this initiative, they developed guidelines for tagging dependency relations initially for Hindi (Bharati et al., 2012) which was further used for other languages. Indian languages being free-order languages are quite efficiently modelled using Pāṇinian framework. The dependency relations formed henceforth are popularly called the *kāraka* relations. *kāraka* relations are not just the syntactic relations of words in a sentence but they correspond to semantic roles too.

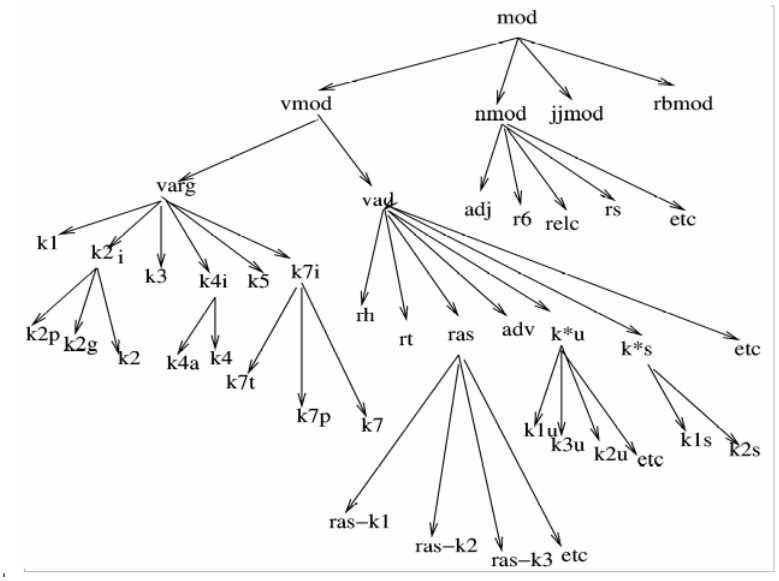


Figure 1.12: Syntactico-semantic relations anncora tagset (retrieved from (Bharati et al., 2012))

1.4.4.6 *saMsādhani* Tagset

saMsādhani[2] is another dependency based annotation framework based on Pāṇinian dependency framework. *saMsādhani* is the Sanskrit computational toolkit, exclusively used for Sanskrit Kulkarni (2016). K V Ramakrishnamacharyulu, , a renowned sanskrit scholar, complied around 89

[1]International Institute of Information Technology-Hyderabad

[2]https://sanskrit.uohyd.ac.in/scl/

dependency relations after a thorough research on Pāṇinian's work on Sanskrit grammar. However, all these 89 relations are not used in building a parser for Sanskrit. Among those 89 relations, 38 relations are identified as useful for parsing task. These 38 tags are fine-grained and involves semantics heavily. *saMsādhani* is considered as a separate annotation schema here because it differs from Anncora in certain aspects. Anncora has customised original Pāṇinian tags to be suitable for Indian languages in general. But *saMsādhani* retains the original tags.

kartā	prayojyakartā	prayojakakartā
karma	karaṇam	sampradānam
apādānam	adhikaraṇam	ṣaṣṭhīsambandhaḥ
kriyāviśeṣaṇam	sambodhanasūcakam	sambodhyaḥ
viśeṣaṇam	nirdhāraṇam	kartṛsamānādhikaraṇam
hetuḥ	prayojanam	karmasamānādhikaraṇa
tādarthya	upapadavibhaktiḥ	prawiṣedhaḥ
sambandhaḥ	pratiyogī	anuyogī
samuccitam	samuccayadyotaka	anyataraḥ
vīpsā	samānakālaḥ	anantarakālaḥ
pūrvakālaḥ	bhāvalakṣaṇasaptamī	sahāyakakriyā
nitya_sambandhaḥ	sahakāraka	vākyakarma
vākyakarmadyotaka	avadhāraṇa	

Figure 1.13: Syntactico-semantic relations (retrieved from Kulkarni (2021a))

1.4.5 Review of Parsers

Parsing as a process evolved when the pioneers in machine translation systems in early 90's found some lacunae in the output due to multiple meaning problems. Subsequently, it was identified that this multiple meaning problem is driven by multiple structural representation of the text. It is then that the need for syntactic intervention in machine translation was first identified in 1955 by Yngve (Hutchins, 2000). Since then, parser was considered as a crucial module in machine translation. Over the years, several works were carried out in the field of parsing and parsers for several languages are developed. In this section, a review of parsers is divided into three sections based on the work carried out in parsing in languages. Firstly, a review of parsers in foreign language (excluding Indian languages) is made. Secondly, a review of Indian languages (excluding Telugu) is attempted. Finally, a thorough review of work on parsing in Telugu is provided.

1.4.5.1 Review of Foreign Language Parsers

In this section, a review of seminal works of parsers that were developed for world languages is provided.

- Magerman and Marcus (1994) developed a probabilistic chart parser named Pearl which is one of earliest parsers based on stochastic models. Pearl is a bottom-up parser consistent with the then existing grammar formalisms and parsing designs developed for English. The system is 88% accurate in providing parse outputs and efficient in resolving Prepositional Phrase attachment ambiguity.

- Charniak (1997) proposed a parsing system based on then existing model for English to which probabilities are assigned. This is one of the earliest works in the field of parsing following constituency grammar formalism. This system proved to yield an Labelled Attachment Score(LAS) of 89.1%. Charniak (2000) also developed a maximum-entropy parser for English.

- Collins et al. (1999) investigates statistical parsing method for Czech which is a highly inflectional language with free order which is quite different from English. In this study, the parsing model of (collins 97) is adopted for Czech. The accuracy of 80% is reported for this system.

- The most popular parsers available using constituency-based approach include: Parser for English by Charniak (2000), the bikel parser developed for Chinese Bikel and Chiang (2000), Berkeley parser proposed by Petrov et al. (2006), Stanford parser for English based on the Probabilistic Context Free Grammars(PCFG) (Klein and Manning, 2002). Also, the most popular parsers using dependency framework include MALT parser by Nivre et al. (2006), Ensemble MALT (Hall et al., 2010), Maximum Spanning Tree (MST) parser by McDonald et al. (2005), Relation extration(RelEx) by Fundel et al. (2007), easy-first parser by Goldberg and Elhadad (2010).

- Currently, the most popular project for parsing is the Universal Dependencies (De Marneffe et al., 2021) project initiated by Joakim Nivre to provide a common parsing platform for all the languages of the world. At present, UD consists of around 100 languages, 200 treebanks.

1.4.5.2 Review of Indian Language Parsers

This section provides an elaborate review of existing parsers for Indian languages. This survey primarily focuses on significant projects involving parsing starting from early 20s.

- Nivre (2009) in his paper titled 'Parsing Indian languages using MALT parser' attempted to parse Bangla, Hindi and Telugu language data as part of the NLP Tools Contest at ICON, 2009. MALT is a transition-based system that maps sentences to dependency trees. MALT parser is optimized for the above mentioned languages and trained the system with a dataset of 1651 sentences for Hindi, 1130 sentences for Bangla and 1615 sentences for Telugu. For evaluation, a corpus of 150 sentences was considered. The Unlabelled Attachment Score (UAS) was reported as 89% for Bangla, 89.4% Hindi and 86.3% for Telugu . Labelled Attachment Score (LAS) for Bangla is 76.1%, for Hindi it is 78.2% and 62.4% for Telugu. These results are for coarse-grained dependency labels.

- For Tamil, Selvam et al. (2008) developed a parser using the hybrid approach that combine phrase structure grammar and dependency grammar with Lexicalized and Statistical Parsing(LSP). A corpus of 3261 sentences was used to train the model and the system was tested with 600 new data. The system was claimed to be 65% accurate. Secondly, Sureka et al. (2014) adopted a hybrid approach using rule-based and machine learning methods (Conditional Random Fields (CRF)). The system is trained with 150 sentences and tested with a new 150 sentence dataset. The system was repoted to have a precison of 82.78%, recall of 93.67% , and a f-score of 87.89%. Also, Muralidaran and Misra Sharma (2016) proposed a construction grammar based dependency approach to parse Tamil data. A corpus of 935 sentences is trained and trained using the MALT parser. For testing, a corpus of 354 sentences is used and an accuracy of 82.21% is claimed

- Bharati et al. (2008a) present experiments on Hindi dependency treebank using two features which enchances the accuracy of the parser greatly. Two features namely: Gender, Number,Person (GNP) and minimal semantics have been incorporated to the Hindi treebank which was claimed to provide an LAS of 89.03% and UAS of 70.93% respectively

- Bharati et al. (2009a) developed a simple dependency parser for Indian languages using a grammar-driven approach. Though it claims to be functional for all the Indian languages, the system is tested only for Hindi. The parser was tested for 5 different intra-chunk relations namely, modified word, modified constraints, modifier word, modifier constraints, dependency relation. Precision is reported as 96.2% and recall as 82.6%

- Ghosh et al. (2009) reports the work on dependency parser for Bengali which was developed as a part of NLP tools Contest at ICON, 2009. A statistical CRF based model coupled with rule-based post-processing technique is adopted for this study. This system is trained with a dataset of 980 sentences which reported the LAS of 53.90%, UAS of 74.09%. It was reported that the accuracy of the system after applying rule-based post-processing increased by 5%.

- Yeleti and Deepak (2009) describe a two stage constraint based approach to dependency parsing for Hindi as part of ICON, 2009. This parser processes syntactic information in 2 levels: firstly, in stage-1 the sentence is parsed for its intra-chunk relations and in stage-2, more complex sentences are considered for parsing. Stage-1 and stage-2 uses hard and soft constraints which correspond to grammatical information and weights from annotated treebanks respectively. UAS is reported to be 85.55%, LAS as 62.20% and LA as 65.88%

- Antony et al. (2010) developed a Penn treebank based syntactic parser for Kannada and Malayalam. For this study, a corpus of 1000 diverse Kannada and Malayalam sentences was construed which were annotated using the Penn treebank guidelines which follows the constituency framework. The treebank thus built was trained using the Support Vector Method (SVM) algorithm. The system was tested using 100 sentences from the same training corpus. The results are encouraging.

- Universal dependencies (https://universaldependencies.org/) currently consists of 8 Indian language treebanks viz., Bhojpuri treebank (Ojha and Zeman, 2020) consisting of 357 sentences, Hindi contains two tree banks: Parallel Universal Dependencies(PUD) (Zeman et al., 2017) hindi treebank and Hindi Dependency Treebank (HDTB) consisting of 1000 and 16,647 sentences respectively, Marathi treebank (Ravishankar, 2017) consisting of 466 sentences, Sanskrit has two treebanks one consists of 233 sentences from Pañcatantra and the other consisting of 4000 sentences from vedic sanskrit (Biagetti et al., 2020), Tamil also contains two treeebanks: Modern Written Tamil Treebank(MWTT) (Krishnamurthy and Sarveswaran, 2021) consisting of 534 sentences from 'A Grammar of Modern Tamil by Thomas Lehmann (1993)' and Tamil Dependency Treebank (TDT) consisting of 600 sentences (Ramasamy, 2012) and Telugu treebank (Rama and Vajjala, 2018) consisting of 1151 sentences. The other languages that are to be part of the UD

repository include Assamese, Kannada, Magahi (Raj et al., 2022), Mandyali, Odia (Parida et al., 2022), Prakrit (Farris and Arora, 2021), Punjabi and Pnar.

1.4.5.3 Review of Telugu Parsers

Few attempts were made in developing Telugu dependency parser using data-driven approaches which are briefed below:

- Vempaty et al. (2010a) in the article titled 'Issues in Analyzing Telugu Sentences towards Building a Telugu Treebank' describes the then ongoing effort in developing a Telugu dependency treebank which is annotated using the Hyderabad Dependency Treebank (HyDT). In addition to this, the authors discussed issues in parsing various linguistic constructions like copula, genitive, implicit and explicit conjunct and complementizer constructions. A corpus of 1487 sentences is annotated as part of this study

- Garapati et al. (2012a) in the work on 'Dative Case in Telugu: A Parsing Perspective' analysed the dative case (*-ki*) elaborately and exhaustively with several examples from a parser perspective. This study show that by providing a semantic environment of verbs and nouns, the dative case markers disambiguated for computational purposes

- Kesidi et al. (2013) implemented a 2 stage constraint-based dependency parser following hybrid approach for Telugu which was earlier used for languages like Hindi. This parser deals with relations in two different stages wherein stage-1 handles intra-clausal relations and stage-2 handles inter-clausal relations. Both stage-2 and stage-2 further goes through H-constraints (structural and lexical knowledge of Telugu)and S-constraints (weights taken from treebanks). The training data consists of 1300 sentences and the testing data consists of 150 sentences. LAS of the system is repoted to be 65.33% , UAS to be 84.14% and LS to be 66.60%

- Kumari and Rao (2015) had developed combinatory categorial grammar supertags for Telugu using which they claim the enhancement of identification of verbal arguments. Using maximum entropy features, a supertagger has be developed. With the intergration of supertagger, the results show an improvement of 1.8% in the UAS and 2.2% in the LAS. This study claims that an MST parser's output can be enchanced using CCG supertags.

- Nagaraju et al. (2016), Kumari and Rao (2017), Kanneganti et al. (2016) worked on various statistical approaches of parsers like MaltParser, ZPar TurboParser, MSTParser, and Easy-First Parser respectively. These studies provide results using small datasets.

- Rama and Vajjala (2018) developed a Telugu treebank using Universal Dependency (UD) tagset with an addition of language-specific tags like nsubj:nc, compound:svc/lvc, nmod:cmp/poss/tmod, obl:tmod/cau, etc for Telugu. A treebank consisting of 1328 sentences is developed and a LAS , UAS of 78.50% , 89.74% is reported

- Gatla (2019) developed a manually annotated treebank for Telugu consisting of 2424 sentences consisting various sentences using Paninian grammatical framework. The training data is used to experiment using data-driven parsers, namely, MST and MALT. MST provides a LAS of 73.62%, UAS of 91.44% and Labelled Accuracy(LA) of 76.30%.

- Nallani et al. (2020b) expanded the existing Telugu treebank of 1600 sentences that is part of ICON 2009 dataset by adding language-specific intra-chunk tags. To the existing annotation guidelines based on the Paninian framework, intra-chunk tags like *nmod_wq* for question words modifying noun phrases, intensifiers, *pof_cv* etc have been added. In addition to improving the existing tagset, annotated corpus of 2000 sentences has been added to the existing treebank. The enchanced treebank is tested with a new dataset of 106 sentences resulting in a LAS of 93.7% and UAS of 95.8%.

- Nallani et al. (2020c) in the article named 'A Simple and Effective Dependency parser for Telugu' attempted to train a BERT model, a contextual vector representation using Telugu wikipedia data. The system is trained of 2400 sentences and tested on 240 sentences which yielded in a UAS of 90.89%, LS of 72.11% and LAS of 70.60%.

1.5 Methodology

Inthis section, methodology adopted for this study is discussed briefly. The following three aspects of methodology are discussed.

1. Theoretical Framework

2. Implementation Technique

3. Corpus used for the study

1.5.1 Theoretical Framework

This study adopts the dependency grammatical framework in general and the Pāṇinian grammatical tradition in particular. Pāṇinian dependency focuses primarily on how information is coded and how it can be retrieved. The parser is built following the Indian theories of verbal cognition where three factors viz. *ākāṅkṣā* (expectancy), *yogyatā* (meaning compatibility) and *sannidhi* (proximity) are used. This framework is discussed in detail in chapter-2 of the thesis.

1.5.2 Implementation Technique

To implement the parser, a grammar-driven or rule-based approach is chosen. It is observed that for languages like Telugu rule-based algorithm yields the desired results on par with the other statistical/neural network parser due to the agglutinative morphology of Telugu. The following advantages stands as a rationale to choose a grammar-driven approach:

1.5.2.1 Why a Rule-based Parser?

1. One of the important reasons to choose rule-based parser is the agglutinative nature of Telugu which encodes prominent syntactic information in the form of suffixes. Syntactic cues are evident on the words of a sentence.

2. Rule-based parser allows wide-coverage of language structures

3. Rule-based does not require huge-corpus annotation.

4. Any error in the output can be easily rectified with some manipulation in the rules

5. Accuracy of the system can be further improved using the inclusion of lexical database

6. Analysis of the sentences are not inconsistent in rule-based parser like in data-driven approach as the sentences are based on the grammatical information of the language. Whereas in data-driven approach analysis purely depends on the annotators. Inter-annotator agreement is not an issue for rule-based parser

7. All the ambiguous structures are retrieved unlike data-driven approaches where one input-one output is possible

1.5.2.2 Rule-Based Parser for Telugu

The parser takes input from sentences which are morphologically analysed, POS tagged and processed through pick-one morph. Telugu shallow parsing tools [1] Uma Maheshwara et al. (2011b) are used as pre-processing tools.
We model the parser as a tree where the nodes of a tree correspond to a word and the edges between nodes correspond to a relation between the corresponding words. For instance, the parsed tree of example-(1.1) is provided as below:

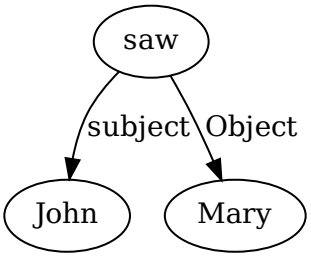


Figure 1.14: Structural Representation for (1.1)

1.5.2.3 Architecture of RBP

The parser under study adopts the dependency grammatical framework to parse the Telugu text. Guidelines to mark dependency relations are build in accordance with the structure of Telugu and the dependency framework that best represents it. The input sentence is parsed using a rule-based implementation technique wherein language-specific rules are provided to the parser. Each stage of the parser is described in detail below. The architecture of the parser can be seen in fig:(1.15):

(i) Morphological Analysis: The morphological configuration of Telugu encodes information like number, gender, case, agreement, tense, aspect, modality, emphasis, dubitativeness, etc. Hence, morphology stands as the prominent component in building a parser. RBP solely depends on morphological understanding of words. Morphemes on words are used to mark relations between

[1] http://calts.uohyd.ac.in/calts

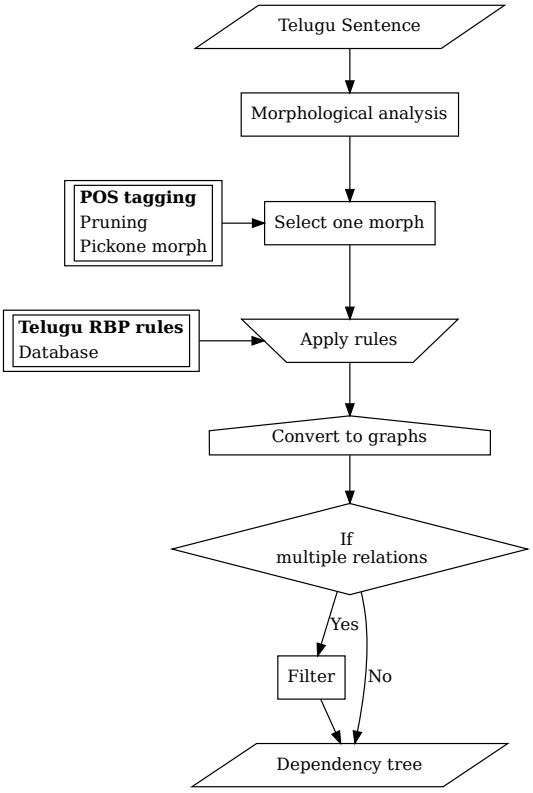


Figure 1.15: Architecture of the Rule-based Dependency Parser

words. Open category of word like nouns, adjectives and verbs and closed class of words such as pronouns, number words and nouns of space and time are studied to thoroughly understand their internal structure and the functions of morphemes. Other categories like indeclinables, compounds, clitics etc are also explored in-detail for marking relations between words.

(ii) Pick one Morph: At this stage, disambiguation happens at the word-level. When a word contains multiple morphological analysis, one candidate analysis has to be picked for further processing. This module automatically picks the morphological analysis of a candidate that best suits the context

(iii) Application of Rules: Compiling rules for each dependency relation is the primary task of this model. Rules are written based on the morphological analysis and the pick one morph of the input sentence. Once the input sentence is separated as token and their respective morph analysis is provided, the complied rules are applied. After the rules are applied, each words gets related to other word

in a sentence using the relations of the annotation guidelines.

(iv) Disambiguation: The next stage in the parser is relational disambiguation. After the application of rules, dependency relations are assigned to each word with respect to other word in a sentence. However, sometimes multiple relations can be assigned to a pair of words. In such cases, a filter is applied to disambiguate the relation. This module makes sure that a pair of words is always assigned a single relation and is free from ambiguity.

(v) Dependency Trees Once the dependency relations are assigned and disambiguated, every sentence from a linear structure is converted to a hierarchical structure. A GUI depiction of each tree is provided using this parser.

1.5.3 Corpus Used for the Study

In this study, corpus is used at two different levels. Firstly, corpus used to build parser rules which acts as a theoretical base to the parser. Secondly, corpus for testing the accuracy of the parser.

1.5.3.1 Corpus to Build the Rules

- This study being grammar-driven, requires a model corpus generalising the properties of language under study. For this purpose, corpus from Telugu grammar books has been collected. The following books were used as reference to frame rules:

1. *telugu vākyaM* (Ramarao, 1975)
2. *bālavyākaraṇaM* (Chinnaya Suri, 1855)
3. A grammar of Modern Telugu (Krishnamurti and Gwynn, 1985)
4. Non-Nominative Subjects (Bhaskararao and Subbarao, 2004)
5. The Dravidian Languages (Krishnamurti, 2003b)
6. Experiencer Subject in South Asian Languages (Verma and Mohanan, 1990)
7. A Reference Grammar of Modern Telugu (Ramarao, 2017)

1.5.3.2 Corpus for Testing

For testing the parser, 1000 sentences from Telugu corpus (3 million words CALTS)) are considered. Finally, the evaluation is based on the following parameters:

1. **Labelled Attachment Score(LAS)**
2. **Unlabelled Attachment Score(UAS)**
3. **Labelled Accuracy**
4. **Relation-based Performance Index**
5. **Confusion Matrix**

1.6 Organization of the Thesis

Chapter 1 outlines the introduction of the thesis with objectives, reviews, methodology and scope of the current study. It explores the parsing trends, theoretical frameworks, treebanks, tagsets available for parsing.

Chapter 2 provides a detailed description of dependency structures, a critical review of dependency grammar , types of dependency frameworks and comparison of dependency with other grammatical frameworks. In addition to this, popular tagsets used for Indian languages are compared for their relevance in parsing Telugu sentences.

Chapter 3 presents the tagset used to label dependency relations in Telugu for Rule-based parser. Every relation as part of a sentence is discussed in detail.

Chapter 4 provides the architecture/implementation of the Rule-based parser including pre-processing tools, components of the parser, algorithm and parser rules.

Chapter 5 titled 'Evaluation and error analysis' evaluates the appropriateness of the parser providing an elaborate error analysis for further improvements

Chapter 6 concludes the thesis, states the limitations and discusses the future perspectives of the study.

Chapter 2

Dependency Framework - A Review

2.1 Introduction

Dependency framework is the widely employed framework in building parsers. In this chapter, the discussion on dependency grammar, its use in computational analysis of language and reviews on the most used dependency frameworks are attempted. Certain crucial questions like what makes a structure a dependency structure?, what are the types of dependency relations? Are all relations in a sentence dependency? are explored in this chapter. In addition to the grammatical frameworks, a comparison of various existing tagsets has also been made. As the current study aims at developing a rule-based parser, the existing frameworks and the tagsets are reviewed and the most suitable one is adopted with required revision for Telugu.

2.2 Dependency Grammar

Though dependency framework was in use to describe languages, it is with the work of Tesnière (1959) that dependency as a grammatical framework to analyse languages is introduced. Tesniere who devoted major part of his life in teaching French, later, using his experiences in teaching, worked on developing a grammar to describe French and other Slavic languages that he taught. Tesniere's work '*Eléments de syntaxe structurale*' (1959) originally published in French is much later translated to English by Timothy Osborne & Sylvain Kahane in 2015.

Tesniere's work stands as a pioneering work in the history of dependency grammar.Though Tesniere pioneered the study of dependency framework, several varieties of dependency emerged from all over the world. DG is also popularly in use in European linguistic tradition with work of Mel'cuk et al. (1988). Functional Generative Grammar by Sgall and Hajičová (1971), Word grammar by Hudson (1984), Meaning-Text theory by Mel'cuk et al. (1988) are some of the varieties of

dependency grammars. In spite of several dependency frameworks, *the core of dependency grammar advocates that a* ***dependency structure is an asymmetric acyclic*** *graph build based on dependency relations between words in a sentence.*

Dependency grammar, currently, is the most popular and widely used framework in natural language processing (Nivre, 2006). This is due to the wide range of advantages that this framework offers.

2.3 What is a Dependency Structure?

A sentence is a collection of words that combine together to convey a specified idea. This grouping of words is not a random process, but is aided by several linguistic mechanisms. Each word in a sentence should be related to another word to convey this collective idea. For example, in a simple sentence 'Radha walks', the words 'Radha' and 'walks' are two individual morphological entities when combined together provides a collective idea. The abstract bond that joins these two words in a sentence is the syntactic bond. Syntactic bond between words is defined in multiple ways by various grammarians over decades. However, the common conception is that there exists a hierarchy between words in a sentence which is driven by syntax. Dependency grammatical tradition describes this bond between words as the bond between a head and dependent or modified and modifier/governor[1] and subordinate bond.

The dependency grammatical model represents the relation between head and its dependents through directed arcs and arc labels. These are called the dependencies. Dependencies are defined as the **binary asymmetric relations** between words in a sentence (Nivre, 2006, pp-10). These relations between content words are marked by dependency relations; functional words attach to the content words they modify. The parse thus generated is a tree, where the nodes of the parse tree stand for words in an utterance and the link between words represent the relation between pair of words.

As the dependency structure is a hierarchical representation, system of ranking between words exists. The word/group of words that constitutes the core of the sentence, often a verb, occupies the top-most rank in the structure. All the other dependencies are attached to the root, that occupies the top-rank in the tree.

All the dependencies in a sentence can either be argument dependencies (subject,

[1]The relations are expressed in various terms in dependency such as the head-dependent, modified-modifier, governor-subordinate

object, indirect object, etc.) or modifier dependencies (determiner, noun modifier, verb modifier, etc.). Consider the example-(2.1):

(2.1) Albert adopted a child

In fig-(2.1), the top-most rank is occupied by the verb 'adopted' which is the head of the sentence and dominates every node under it. The arguments 'Albert' and 'child' are at the same level in the tree occupying the same rank. Nextly, the determiner 'a' modifying the noun 'child' is placed lower in the hierarchy than the child as it is dominated by it.

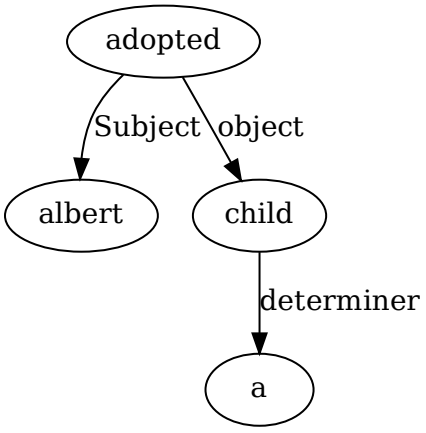


Figure 2.1: Tree for (2.1)

Using the structure of the dependency tree in figure-(2.1), some criteria to identify a dependency structure are discussed below (de Marneffe and Nivre, 2019):

1. 'Head determines the syntactic category of structure which can often replace that structure. For example, in the figure-2.1, 'Child' is the head of the phrase 'a child'. 'child' determines that the phrase is a noun phrase and stands as an obligatory element

2. Head determines the semantic category of sentence; dependent provides semantic specification.

3. Head is obligatory; dependent may be optional.

4. Head selects dependent and determines whether dependent is obligatory or optional.

5. The form of dependent depends on head.

6. The linear position of dependent in a sentence is specified with respect to head'.

Another important criteria of dependency is the constraint of projectivity.

2.3.1 The Constraint of Projectivity

Projectivity is a concept dealing with dependency tree and the word order. Dependency grammars consider 'projectivity' as a norm. A dependency tree is projective if every relation from a head to dependent and a node(n), 'n' occurs between head and dependent in the linear order if 'n' is dominated by head. Alternatively, "a dependency tree is projective if the yield of every subtree forms a contiguous substring of the linear order" (de Marneffe and Nivre, 2019). A simple way to identify non-projectivity in a dependency tree is to check for arcs that cross one over other. Though dependency grammar advocates projectivity, there are structures in a sentence which are non-projective. These non-projective trees occur due to long-distance dependencies where-in words are not in linear order. Telugu, having free-word order, may tend to have non-projective trees. However, Bhat and Sharma (2012), in their paper titled 'Non-projective structures in Indian language treebanks' states that non-projective trees are almost absent in Telugu. It is also observed as part of our study that non-projective trees are a rare phenomenon in Telugu. For example, a phrase *pramukha naṭi* 'famous actress' is possible but *naṭi pramukha* is not possible. This information of grammaticality is coded in Telugu morphologically. This study exploits all such morphological information to prune-out ungrammatical and non-projective constructions.

2.4 Dependency vs Non-dependency Relation

A dependency relation is between two words that are in modified-modifier relation. It entails that the modifier requires a modified which has an expectancy and cannot exist without it. Each such pair of words get into a dependency relation. The modified word is higher in the hierarchy than the modifier. Words like verb-noun, noun-adjective, verb-adverb etc. are examples of dependency relations. In a verb-noun relation, verb is the modified and noun is the modifier. Verb has an expectancy of the noun and a noun does not exist without a verb. Hence, there is a dependency of a noun on the verb, thus a dependency relation is established.

It is not always the case that every word in a sentence is related to other only based on a modifier-modified relation. There are also words which do not modify any other word but are still part of the sentence. Words like address terms, words that are conjoined using a conjunct, parts of words that are written separately in orthography are all such words that are not part of a hierarchical structure. Hence, they occupy the same rank in the dependency tree. These structures fall under the category of non-dependency structures. These are marked horizontally in a dependency structure. Consider (2.2)

(2.2) *nēnū* *rāmū* *vacc-ā-mu*
I.NOM.CONJ Ram.NOM.CONJ come-PST.3.PL
'I and Ram came'

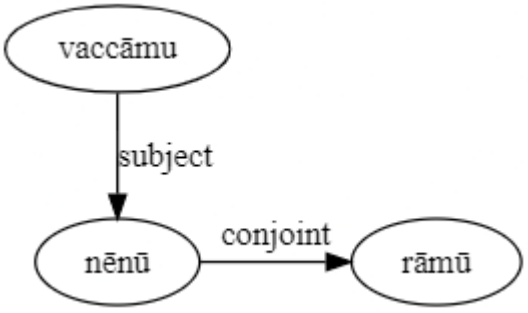


Figure 2.2: Dependency tree for the sentence-(2.2)

In the example-(2.2) and the dependency tree in figure-(2.2), 'nēnu' and 'rāmu' are conjoints which is expressed with lengthening of the final vowel in Telugu i.e. *nēnu* and *rām=u*. In this case, they serve the same function i.e. conjoined subject to the verb 'came'. Hence, they are placed at the same rank in the tree. These conjoints are not dependent on one another, leading to a non-dependent arc position in the dependency structure. Other such relations include light verb constructions, negative particles, and so on.

2.5 Grammatical frameworks - A Comparison

In this section, we attempt to compare the widely used grammatical frameworks of dependency and constituency. Constituency or phrase structure grammar has taken over the field of linguistics since 1957 (Chomsky, 1957). However, it could not attract attention in the field of natural language processing. Both dependency and constituency frameworks have their own advantages and disadvantages. Hence, we compare phrase structure grammar to dependency framework to gauge their viability

for parsing purposes. Initially we compare dependency and constituency and later, compare two frameworks within dependency grammatical framework.

2.5.1 Differences Between Phrase Structure and Dependency Grammar

1. Structural representations in Phrase Structure(henceforth PS) can be viewed as indirect links between units of sentence structures as intermediary groupings of nodes called **constituents** are formed. But structural representations in DG can be viewed as a direct connect between **words**. (Osborne, 2019, pp-33).

2. **Word to node ratio**
   In PS, number of words in a sentence is not equal to number of nodes in the tree. Because of the phrasal nodes, the number of nodes are higher in PS. Hence, it is one-one or more mapping between words and nodes. However, in DG, number of words in a sentence is equal to number of nodes[1].

3. **Dependency Structure(DS) does not have too many layers of linguistic analysis like in phrase-structure**
   A dependency structure is a tree spanning all the words in a sentence. It represents relation between words in a sentence. It does not add any other entities in-order to analyse a sentence. But for most constructions, DS solely analyses words in a sentence and nothing beyond. Contrary to this, phrase-structure grammar or other grammars based on phrase-structure grammar contains multiple linguistic layers in a tree which are not of interest from computational perspective, especially for building parsers in NLP applications.

4. **Subject-Predicate division**
   PS advocates the binary subject-predicate division and places the noun phrase and the verb phrase at the same level. Whereas DG rejects the subject-predicate division and adopts verb-centrality.

5. **Linear Contiguity**
   PS grammar dictates its words to be part of a constituents which ensures the linear contiguity of words in a sentence. However, in DG, words need not

[1]As mentioned in the earlier sections, often so in PDG, in verbless equative constructions, a null-verb is introduced which leads to the increase in number of nodes. Some such constructions remain as exception to this statement

necessarily attach to their parent node de-emphasizing its focus on word-order and shifting focus on hierarchy. This is one of the reasons for DG to be best-suited for free-word order languages (Osborne, 2019, pp-69).

6. **Linearisation**
   A dependency graph converts the linear order of words into structural order. Sub-trees in DS mostly yield in linear order of words in a sentence. In such cases, the order of heads and their dependents is important. Tesnière (1959) states two important points
   1). A dependent usually follows or precedes the head depending on the word-order of particular language.
   2). If they are far from each other morphological devices like agreement provides the connection between them.
   Dependency grammar focuses on projectivity. But it is not always true that trees are projective. For example, coordinate structures are non-projective.

7. **Linguistic generalizations - word order**
   "Dependency trees are not sensitive to the order of the words in a sentence, in contrast to phrase-structure trees" (de Marneffe and Nivre, 2019). Dependency tree does not change according to the word-order of the language or order of words in a sentence. For example, sentences like 'When I am tired, I take a nap' and 'I take a nap, when I am tired' have different structural representation in PS whereas in DS both of them have the same structure. This makes dependency more adaptable for languages with flexible word order yielding in a single analysis.

8. **Transparency of trees**
   Dependency trees are transparent in the sense that it is easier to comprehend the hierarchies in a sentence. For example, in a dependency tree, the modified-modifier or head-dependent relation can be easily identified. However, in phrase structure tree, it is quite difficult as the trees are intricate and has several extra elements apart from words in a sentence.

From all the above discussion on theoretical frameworks, dependency grammatical frameworks have many advantages when compared to the popular phrase-structure. Telugu, having a flexible-word order can be best modelled using dependency grammatical framework.

2.5.2 Dependency frameworks - A Comparison

As dependency frameworks are popular in both Indian and western traditions, we provide a review of Pāṇinian and Tesniere's dependency and draw a comparison.

2.5.3 Pāṇinian Dependency framework - Indian Grammatical Tradition

Pāṇini's '*Ashtadhyayi*' which translates to 'The eight chapters' deals with the grammar of Sanskrit. Eventually, this grammar is popularly established as the Pāṇinian Grammatical (PG) Model which was used by major Indian languages. Many Indian language grammars were framed based on Sanskrit terminology. Pāṇini's grammar is described using meta-rules, unlike other descriptive grammars. For instance, grammatical rules to decide any semantic role of a noun is given in a more general manner than in terms of grammar. Using these generic instances, Sanskrit grammar is interpreted. Special terminology or metalanguage is coined for ease of explanation. Several sections on word and sentence formation from roots and rules on structural transformations are part of this grammar. Pāṇini's concept of recursion to repeat elements of earlier rules in later rules is a novel idea which later is considered a norm in computer science. A finite set of rules is sufficient for Pāṇini's system to generate an infinite number of sentences. The algebraic character of Pāṇini's rule was not appreciated in the west until recently when similar generative structures were discussed by Noam Chomsky (Chomsky, 2013) and other proponents of Chomsky's idea. Previously, in the 19th century, Pānini's analysis of roots and suffixes in Sanskrit, later, is considered a prototype for computational analysis of other Indian languages. Especially in parsing, Pāṇinian understanding of sentence formation, constraints imposed on words in a sentence, analysis of roots and suffixes etc is an invaluable resource. This grammar is useful for both analysis and generation alike. In this section, we describe some key concepts in PG framework that are relevant to parsing.

Firstly, PG model is the oldest dependency traditions available that discards the idea of constituents. In this model, dependency structure is considered to be a relation which is expressed as *viśeṣya-viśeṣana* 'modified-modifier' relation. For example, consider a phrase 'happy child' in which the adjective 'happy' requires a host to attribute its quality of 'happiness'. And the word 'child' acts as a host and gets modified by the adjective. The host that gets modified is the head and the modifier that attributes some value to the host is the dependent. Hence, the word 'happy' is the modifier or *viśeṣana* and 'child' is the modified or *viśeṣya*.

The entity that occupies the top-most rank in the dependency tree is called the '*mukhya viśeṣya*'(primary modified) or primary head of the sentence. All the other elements are directly/indirectly dependent on the primary head. It should be noted that a dependency structure can have multiple heads but only a single primary head.

A sentence is formed when a group of meaningfully compatible words combine to provide a collective sense. The point of curiosity lies in understanding how each word in a sentence is related to each other. The theories of *śābdabodha*[1] of Indian grammatical tradition is a knowledge-resource that define rules in parsing. *śābdabodha* has three essential factors for words to form a structurally and semantically meaning sentence, these factors are to be fulfilled (Kulkarni, 2021a).:

(i) *ākaMkṣā* 'Expectancy'

(ii) *yōgyata* 'Compatibility'

(iii) *sannidhi* 'Proximity'

2.5.3.1 *ākaMkṣā* 'Expectancy'

According to the PG framework, every word in a sentence has a desire or expectancy for some other word in the same sentence, which is termed as *ākaMkṣā*. This expectancy among words is one of the three primary features in formation of sentences. This is due to the fact that there is an expectancy/desire among words to form a meaningful and a complete sentence. Expectancy, in western grammatical traditions, is often noted to be observed between verb and a noun which is termed as sub-categorization frames (in Chomskian tradition) of verbs or valency of verbs (in Tesniere dependency). However, expectancy can be between any category of words in a sentence. An expectancy of verb for its participants (often nouns) is the most common and dominant expectancy in a sentence. Nevertheless, expectancy between noun and a noun, modified noun and modifier adjectives, modified verbs and modifier adverbs, expectancy between indeclinables etc are also quite prevalent in languages. Consider the example (2.3):

(2.3) *pilla-lu* *vacc-ā-ru*
kid-PL come-PST-3.PL

[1]The term *śābdaboda* translates to śabda - 'sentence/language string' and bodha - cognition. The theory of *śābdaboda* deals with the cognition derived from a language string and the processes involving in forming a meaningful & complete sentence

'Kids came'

In the above phrase, the verb *vaccāru* 'come' has an expectancy of a noun which is fulfilled by the noun *pillalu* 'kids'.

Though this appears simple on the outset, it gets complicated when this information has to be programmed to build a parser. The information of expectancy is encoded in a sentence in various forms like suffixes, agreement on words, properties of words like transitivity for verbs, position, semantic properties of nouns etc. For instance, the oblique form of *illu* 'house', '*iMṭi* has an expectancy of either a case marker/a nominal entity. Similarly, when a verb like *ivvu* 'give' occurs in a sentence, it has an expectancy for a subject, object and an indirect object. It should be observed that in morphologically rich languages, morphemes carry numerous information about the syntax and semantics of the sentence. All possible linguistic cues which are overtly present on surface are to be utilized in building parsers. However, certain expectancies might also require extra-linguistic information which might not always be possible to provide it in the form of rules.

PG framework extensively discusses expectancies in various extra-linguistic (world knowledge) and linguistic environments which proved to be useful for many Indian languages including Telugu.

2.5.3.2 *yōgyata* 'Compatibility'

The concept of *yōgyata* refers to mutual compatibility between words in a sentence. Each word in a sentence should be mutually compatible to each other. This idea of compatibility is related to the Noam Chomsky's (Chomsky, 1957) famous construction '*colorless green ideas sleep furiously*'. It shows that though the mutual expectancy feature is fulfilled, words are not compatible with each other giving rise to a meaningless construction. Hence, compatibility or *yōgyata* among words is a requisite for a meaningful sentence.

The concept of *yōgyata* is synonymous to semantic restriction/constraints in western linguistics. Semantic restriction refers to the selection of contextually appropriate words in the specified slots in a sentence. As the term suggests, semantic features including ontological properties of words are considered in building semantic frames of words. For example, the verb *tinu* 'eat' requires a noun[+edible] feature to form a meaningful entity. *yōgyata* ensures that every word's semantic restriction is fulfilled by its modifier.

2.5.3.3 *sannidhi* 'Proximity'

Another important factor of sentential analysis is the *sannidhi* 'proximity' of each word to another word in a sentence. There must not be an intervention or distance among the words which obstructs the comprehensibility of a sentence. *Proximity* is closely related to the priniciple of projectivity, that we discussed earlier. If the nodes of a dependency tree reflects the linear order of words that represents the surface structure of the sentence, then the tree is projective. If there is any cross-over between words in a sentence that might lead to non-projective trees. The constraint of proximity is important to avoid ambiguous interpretations of the sentences.

Pāṇinian grammatical framework provides a wholesome understanding of sentence analysis especially for free-word languages that function heavily on morpheme information. The morpheme information on words carry a specific syntactic and semantic identity in the sentence, that is utilized in the theories of *śābdabōda*. These insights can be used for Telugu which is rich in the use of morphemes.

2.5.4 Tesniere's Dependency Framework

Lucien Tesniere, a French linguist, is considered the father of the dependency tradition. Lucien Tesniere's '*Eléments de syntaxe structurale*' (1959) marks as the starting point of DG which is most prevalent in modern computational tasks. His work on dependency has been overlooked due to the immense influence of phrase structure grammar then. But Tesniere's work released just two years after Noam Chomsky's seminal work, *Syntactic Structures* (1957).

Tesniere's Dependency Grammar (henceforth TDG) considers syntax as a relation between the units of syntactic structures, words. TDG considers words as the minimal units of sentence structures. TDG provides analysis for converting a linear order of words to a structural order which is multidimensional. This grammar is designed to be suitable for most of the natural languages as it contains examples from over 50 languages unlike constituency framework which is majorly described from English language perspective. TDG considers connection, junction and transfer as the main components of syntactic analysis. It discards the idea of subject and predicate which is borrowed to linguistics from logic which does not serve any purpose in syntactic analysis.

Here, key concepts introduced by Tesniere as part of his dependency grammar is discussed:

- **Connection**: Connection is defined as the link between words in a sentence. Tesniere states that a sentence is not merely a collection of morphological units(words) but also a link that joins these words together. Connection establishes a **dependency** between words. For example, in a sentence 'Sita sings', apart from the two morphological words, 'sita' and 'sing', there is an abstract link that holds them together which Tesniere calls a connection which can be seen in the figure-2.3

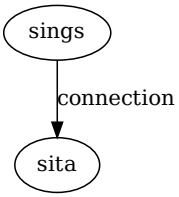


Figure 2.3: Connection

- **Governor and subordinate**: When a connection between two words is established, one word can be superior to the other. Sometimes, connection can be established between words of similar rank. However, when a connection happens between a superior and subordinate words, the superior word is called the governor and the word that is dependent on the governor is called the subordinate. In the sentence, 'Sita sings', 'sings' is the governor and 'sita' is the subordinate that can be seen in fig-2.3.

- **Stemma**: The graphical representation of a link between words and the hierarchy is what Tesniere calls a Stemma. Stemma contains words and the links driven by their syntactic relations. Stemmas are directed as there exists a non-uniform relation between words. A relation between a governor and a subordinate is depicted using a vertical line. Stemmas are equivalent to modern dependency trees consisting of various relations.

- **Junction**: Conjunction, which Tesniere calls junction is the concept that most dependency grammarians differ in. The structural representation of coordination has been controvertial across dependency grammars and Tesniere provides an insightful discussion in this matter. According to TDG, junction unlike other phenomenons like subordination is not a vertical relation but is a horizontal relation where the elements joined by

junctive(conjunction) are placed on the same level. TDG differentiates between partial and full junction. Tesniere's understanding of coordination is one of the best representations employed.

- **Transfer** : Transfer is another important concept in syntax that Tesniere discussed extensively. Transfer refers to the syntactic phenomenon of using the same word in another syntactic position. This is one of the most productive tools of language. Tesniere classified words into content and function words. Function words are referred in TDG as translatives. Translatives are considered as the tools to transfer as they convert content words into other syntactic categories.

Tesniere postulated dependency grammar based on the above key concepts. However, TDG was questioned for its capability to analyze coordinating structures. This is when some grammarians expressed the need for constituency analysis and included constituency. However, (Hudson, 1980) argues and proves that dependency in itself is sufficient to address the issues in syntactic analysis of natural languages using illustrations. He further argues as to why constituency analysis is not an appropriate framework for syntactic analysis. Later in this chapter, we attempt a comparison of analysis of coordination in TDG and other related frameworks.

It is interesting to note that, though Chomsky and Tesniere described similar topics, the terminology used by both of them is quite different with subtle conceptual differences. Basic concepts like valency, subcategorization frames, trees, stemma etc are defined differently by both Tesniere and Pāṇini.
Likewise, both TDG and PG deal with dependency yet there are certain differences. Here, we discuss the differences and similarities between them briefly.

1. Subject-predicate distinction is rejected by both the frameworks. Both PDG and TDG did not agree on placing the subject on the equal ground as the predicate in the graphical representation. They acknowledge verb centrality and consider the verb as the head by placing the actants (arguments) below the verb. This brings in symmetry.

2. Both frameworks advocate verb-centrality. PG & TDG positions verb as the root of all sentence structures. In both the frameworks, main verb is usually placed as the root of the dependency tree. However, in PG "verb" is always considered as the central node even in the absence of a verb in the sentence i.e. a null/dummy verb is inserted in case the verb is absent. But in TDG,

if the verb is not overtly present, adjectival/nominal/adverbial nodes are still considered the central nodes. "In a simple sentence, the central node is not necessarily a verb, but when there is a verb, this verb is always the central node of the sentence" (Tesnière, 1959, pp-98). Hence, an introduction of a null-verb is not necessarily made in TDG.

3. Function words along with content words are considered as heads in TDG (translatives (auxiliary verbs, prepositions, subordinators)), auxiliary verbs are enclosed in a nucleus circle , no syntactic autonomy for function words, they are equi-level to the content words (Tesnière, 1959). Whereas PDG treats auxiliary and the main verb together. In PDG, only content words are the considered as heads.

4. Dependency representations of coordination is different in both the frameworks. But both the frameworks agrees on the idea that conjoints should be in a horizontal relation and the conjunction is not the head.

It is observed that both PDG and TDG aligns well in majority of the concepts and keeps the core of dependency in-tact. The major difference lies in the coinage of concepts. Except the minute differences, both frameworks advocates the key concepts of dependencies alike. Osborne (2019) expresses that Tesniere's coinage of new and complicated terminology is one of the reasons for not having wide attention of dependency grammar.

2.6 Is Dependency Grammar Adequate for Computational Purposes?

Descriptive grammars are composed by linguists several years ago without considering the massive technological applications of NLP. But now, NLP has taken a big leap forward. This required a concrete grammar which describes natural language grammar and also computationally parsed the language. In this section, we examine if dependency grammar adequately describes and parses language computationally.

The utmost important feature of a grammar as described by Jarvinen and Tapanainen (1998) is 'descriptive adequacy'. If a grammar is described adequately then it is empirically suitable for other NLP tasks. Tesnière (1959) describes his dependency framework from the perspective of major world languages and typologically different languages. Osborne (2019) in his book 'Dependency

grammar for English' makes an attempt to describe grammar of English using dependency grammar which is fairly a fruitful attempt. Osborne (2019) has comprehensively stated the advantages of describing a grammar in dependency framework. Hence, it is proved that a language can be adequately described using dependency grammar. Also, advantages of using Pāṇinian dependency grammar for computational purposes has also been proven by Sangal et al. (1995), Kiparsky (2007), Kulkarni (2021b) and many others. The main advantages of dependency pertaining to parsing task are the following as stated by Covington (2001):

- Dependency relations are equivalent to the semantic relationships that are needed for the next stage of interpretation; 'it is not necessary to "read off" head-modifier or head-complement relations from a tree that does not show them directly'.
- The dependency tree contains one node per word. Because the parser's purpose is only to connect existing nodes, not to postulate new ones, the task of parsing is in some sense more straightforward.
- Dependency parsing lends itself to word-at-a-time operation, i.e., parsing by accepting and attaching words one at a time rather than by waiting for complete phrases.

2.7 Existing Tagsets: A Discussion

Over the period of time, with increasing demand for language technology, dependency grammar has been applied in building various language tools. For parsing purposes, tagsets are a pre-requisite. In this section, we discuss the most popular tagsets available for marking dependency relations viz., 1) Universal Dependencies and 2) AnnCora (Hyderabad Dependency Treebank tagset (HyDT)). We have considered the above two tagsets as they are widely used for Indian languages. After drawing a comparison, we choose the most appropriate tagset for marking dependency relations in Telugu.

2.8 Universal Dependencies

Universal Dependencies (UD) [1], is a platform initiated to facilitate cross-linguistic morphosyntactic treebank annotation. The rationale behind this initiative is to

[1] https://universaldependencies.org/guidelines.html

provide a consistent annotation schema for all the world languages in order to support higher NLP applications. UD currently consists of 200 treebanks contributed by 300 language specialists for around 100 languages. UD is originally stanford dependencies [1] which evolved to accommodate world languages.

UD annotation schema has 37 universal dependency tags and around 198 language-specific tags which are added whenever necessary. Universal tags are shallow whereas language-specific tags often provide fine-grained information about the relations. UD is fast-growing in terms of number of treebanks, it currently consists of around 180 treebanks for various world languages. In addition to this, around 65 new language treebanks are also part of future extension.

UD concentrates more on syntactic parsing in order to be suitable for downstream applications like question-answering, relation extraction, text summarization etc. However, language-specific tags often add semantic information wherever necessary. The dependency relations in UD are simple and mostly taken from traditional grammar labels like subject, object, indirect object and the like. This familiarity with tags makes it easier for both linguists and non-linguists (computer scientists, data engineers etc) who are working on language processing to work with treebanks.
UD is a syntactic parser that provides syntactic analysis of the sentence. UD utilizes Google universal part-of-speech tags (Petrov and McDonald, 2012) for the morphological analysis with a revised version of morphological features from the Inter-set interlingua for morphosyntactic tag sets (Zeman, 2008).

2.9 AnnCorra

This tagset follows the Pāṇini's dependency tradition. The peculiar feature of Pāṇinian dependency is to provide syntactico-semantic relations. Based on these syntactico-semantic relations, (Bharati et al., 2012) has developed a dependency tagset known as AnnCorra tagset which can be used for almost all major Indian languages. AnnCorra, an abbreviated form of 'annotated corpora' is an initiative taken up by IIIT-H[2] as part of 'Workshop on Lexical Resources for Natural Language Processing' to develop linguistic resources for Indian language in electronic form. As part of this initiative, they developed guidelines for tagging dependency relations initially for Hindi (Bharati et al., 2012) which was further used for other languages. Indian languages being free-order languages are quite

[1] https://nlp.stanford.edu/software/stanford-dependencies.html
[2] International Institute of Information Technology-Hyderabad

efficiently modelled using Pāṇinian framework. The dependency relations formed henceforth are popularly called the *kāraka* relations. *kāraka* relations are not just the syntactic relations of words in a sentence but they correspond to semantic roles too.

This tagset originally consists of around 19 fine-grained tags for *kāraka* (K) relations and 25 fine-grained tags for non-*kāraka* /(r) relations. *kāraka* relations are used to express the dependencies between noun-verb and non-*kāraka* relations are used to capture noun-noun and other dependencies/non-dependencies. Nallani et al. (2020c) added inter-chunk tags for the existing AnnCora tagset. In total, 68 inter-chunk dependency relations are identified for Hindi as part of this scheme. The present study aims to expand the current AnnCora tagset for Telugu.

2.10 Comparison of Tagsets

In this section, two dependency based annotation schemata namely, Universal Dependencies(UD) version-(2.7) and *AnnCorra* version-(2.6) are compared for its suitability and an enhanced tagset is proposed for tagging relations in a rule-based parsers (RBP) for Telugu. Treebanks are automatically generated using the RBP. If necessary, output can be converted into any tagset. The aforementioned tagsets are dependency-based, however they differ in representation of certain constructions. This section focuses exclusively on major structural differences in certain special constructions like non-nominative subject constructions, verb-less constructions, constructions with coordination. After the comparison, the most appropriate tagset and representation of such special constructions in Telugu are described. If necessary, a special tagset for Telugu parsing is framed.

We focus the comparison on two major aspects - head projection and subject representation in both the frameworks.

2.10.1 Head Projection

The head of the sentence in dependency relation is generally called as 'root' and other dependent nodes are projected by connecting with the root. The handling of 'root' in Anncorra & UD show considerable differences in the following instances:

2.10.1.1 Nominal Predicate

The occurrence of copula in nominal predicates is covert in Telugu in affirmative constructions. In UD, the predicate nominal is projected as the `root` i.e. the head

and copula as a dependent. Whereas, in AnnCorra, the copula is the head and in case of absence of copula, a NULL node is established and conceived as the head/root and the nominal predicate is tagged as k1s (*kartā samānadhikaran* as seen in the following example[1].

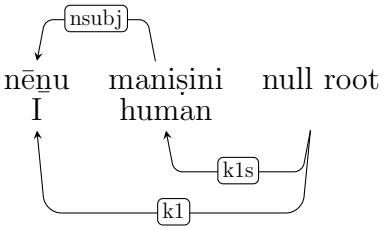


Figure 2.4: 'I am a human'

2.10.1.2 Coordinating Conjuncts

While tagging coordinating conjuncts in UD, the head-first approach is followed, and the first noun is projected as the head, whereas in Anncorra the conjunction is given the status of head.

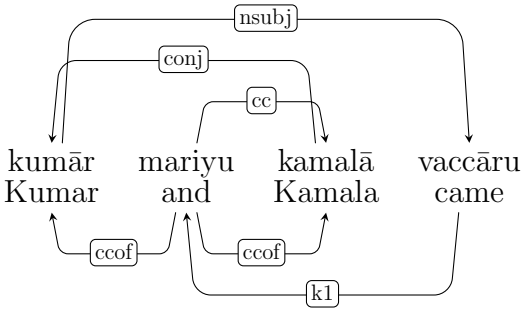


Figure 2.5: 'Kumar and Kamala came'

In Telugu, the coordinating conjunctions are canonically expressed by lengthening the final vowel of the coordinated phrases or by the conjunction *mariyu* (Krishnamurti and Gwynn, 1985).

Section (3.6.1) explains how these sentences are handled in the Rule-based Parsing.

[1]In the following dependency trees, examples with edge above relations are UD and edge below are marked with AnnCorra relations

2.10.1.3 Complement Clause

In complement clause, the complementizer is tagged as 'mark' in UD as dependent to the subordinate clause. However, the complementizer is projected as head and given the appropriate dependency relation in *AnnCorra*.

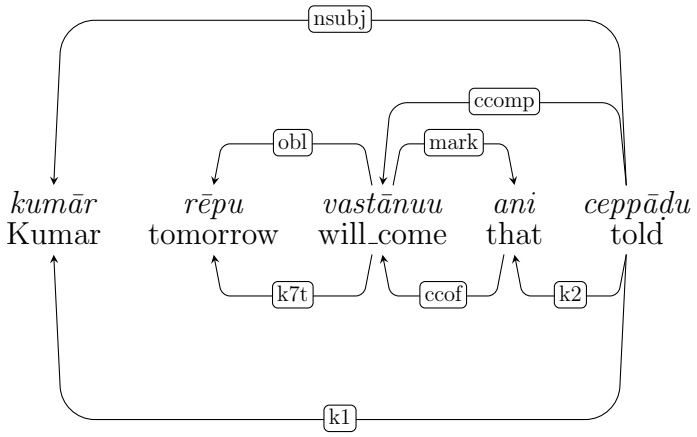


Figure 2.6: 'Kumar told that he will come tomorrow'

2.10.2 Subject

Subjects are marked for different relations in UD and Anncorra. This section explains the agentive subjects, experiencer subjects, possessive subjects.

2.10.2.1 Agentive Subject

A noun phrase that serves as a proto-agent or as a subject of a sentence is tagged as '`nsubj`' in UD (De Marneffe et al., 2014). Further, in passive construction, the noun phrase which is the proto-agent is tagged as '`obl:agent`' and the proto-patient as '`nsubj:pass`'. In annCorra, *kartā*, roughly an agent of an action is tagged as '`k1`' either in the active or passive constructions. The corresponding '`nsubj:pass`' of UD is tagged as '`k2`' i.e. *karmā*, roughly the patient.

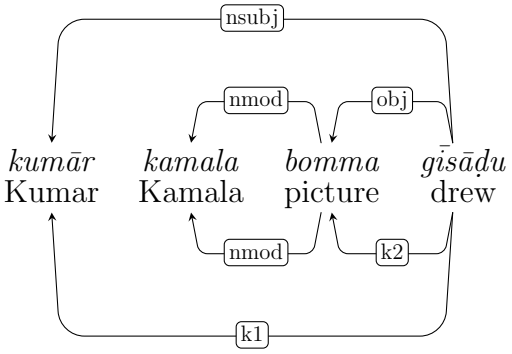


Figure 2.7: 'Kumar drew Kamala's Picture'

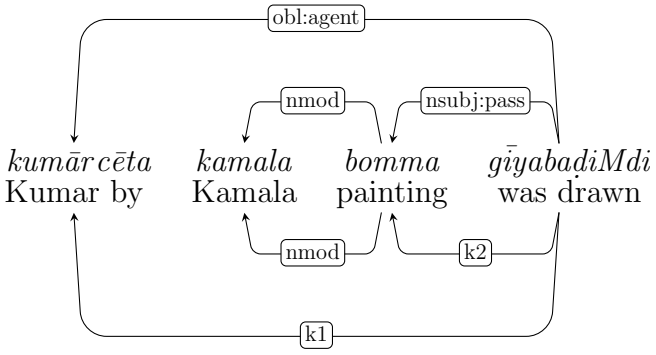


Figure 2.8: 'Kamala's Picture was drawn by Kumar'

2.10.2.2 Experiencer Subjects

An experiencer subject is usually marked by the dative case marker in Telugu. Verma and Mohanan (1990) describes that "In the so-called experiencer subject constructions in South Asian languages, the thematically prominent argument, which we expect to be a grammatical subject, is quite often an experiencer, and is marked with the case otherwise associated with indirect objects" . The dative marked subject acting as an experiencer subject is the most widespread in Dravidian languages (Subbārāo, 2012).

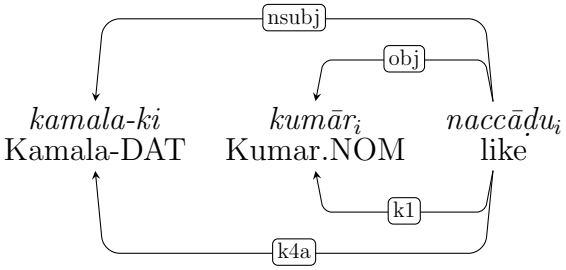


Figure 2.9: 'Kamala likes Kumar'

2.10.2.3 Possessive Subjects

While expressing possessive subjects with the verb 'have', the dative and locative case markers are used in Telugu to express inalienable and alienable possessions respectively. Possession is not marked for any case marker, thus realized in the nominative case. The verb to show the possession is expressed by 'be' form as Telugu do not have any form that corresponds to the verb 'have'. The tagging of possessive construction is as follows:

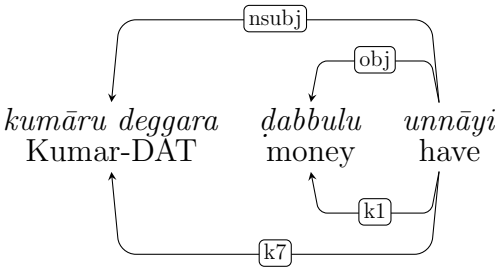


Figure 2.10: 'Kumar has money (i.e. Kumar is wealthy)'

In this construction, the marking of '`obj`' in UD does not conform to the selection of intransitive verb 'to be'. However, in annCorra, the possession is marked using `k1` and the locative case marked subject is marked using `k7`.

2.10.3 Causative Agent

Causatives in Telugu is realized as a morphological process. In Telugu, the pheriphrastic causative marker *-iMcu* is attached with the verb. In AnnCorra, the causer is marked as pk1 (prayojaka karta 'causer') and the causee as jk1 (prayojya karta 'causee'). On the other hand, in UD pk1 is marked as as *nsubj* and jk1 as *obl* or with the language specific tag *obl:agent*. Here, the information on 'causer' and 'causee' are clearly marked in Anncorra.

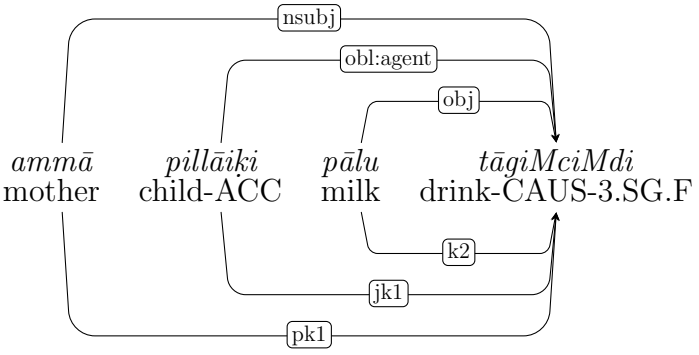


Figure 2.11: 'Mother makes the child drink milk'

2.10.4 Secondary Patient

Tagging the secondary patients in ditransitive verbs differ in Anncorra and UD. In Telugu, the objects in ditransitive communicative verb such as 'ask' are marked with the accusative case marker. However, the entity from which the information has to be elicited is marked as k2g as it functions as the secondary object (Bharati et al., 2009b), where as in UD, the secondary patient is given the tag 'iobj'.

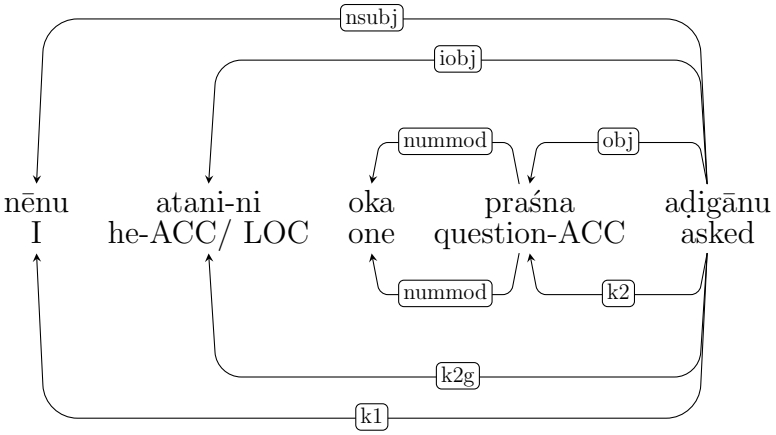


Figure 2.12: 'I asked him a question'

After a thorough comparison, it was observed that certain relations like causative subjects, passive subjects, experiencer subjects are syntactico-semantically well-expressed in *annCorra* than UD. Further, complement noun/predicative adjective in verbless constructions being the root of the construction is also not a convincing representation in UD. But certain tags in UD like the `obl` tag is a generic tag for adjuncts given when the exact relation cannot be identified. However, in annCorra, every relation has a specific tag and is quite fine-grained when compared to UD. In addition to this, *annCora* having tested and proved to be suitable for free-word order Indian languages,is chosen to mark relations in the present study.

Nevertheless, annCora tagset is not adopted as it is. It is modified for certain constructions and also new fine-grain tags are added wherever necessary. The tagset used for this study is enhanced for Telugu and henceforth called as enhanced AnnCorra.

2.11 Enhanced Anncorra for Rule-based Parsing

The Anncorra-v-2.7 guidelines are enhanced to account for Telugu when we built the Rule-based parser (RBP) with 51 relations for Telugu. RBP guidelines differ

from the original anncorra in following ways:

- In RBP enchanced anncorra guidelines, no functional element/indeclinable is marked as the head. Hence, conjunction in coordination construction or complementizer in complement clauses is not the head.
- `vmod` is an under specified tag which is used for wide variety of relations in anncorra guidelines. Therefore, `vmod` tag is further divided into several sub-tags.
- A `null verb` is introduced in case of verbless equative constructions in Telugu.

2.11.1 Representation of Coordination in Enhanced AnnCorra

Representation of coordinate structures in languages has been the topic of discussion since Tesniere's dependency grammar was introduced. Many scholars differ in their analysis of coordinate constructions. Here, we present a brief review of various analysis and examine which analysis best suits Telugu coordinate constructions. we present the work of Tesniere (Tesnière, 1959), Melcuk (Mel'cuk et al., 1988), Hudson (Hudson, 1984), Rosta (Rosta et al., 2005) and Timothy Osborne (Osborne, 2019)

2.11.1.1 Tesniere - *elements de structurale syntax*

Tesniere used the term 'junction' to refer to conjunction. Tesniere states that junction is a horizontal relationship unlike other dependency relations. In junction, there is no hierarchy between conjoints. Consider the example (2.4) and Tesniere's analysis in fig-(2.13):

(2.4)

nēnū	*mariyu*	*rāmū*	*vacc-ā-mu*
I.NOM.CONJ	and	Ram.NOM.CONJ	come-PST.3.PL

'I and Ram came'

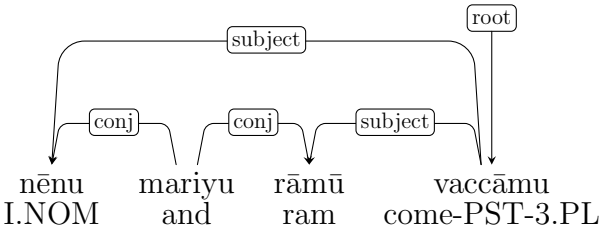


Figure 2.13: Tesniere's representation of coordination

2.11.1.2 Melcuk - Meaning-text Theory

Melcuk in his Meaning-Text Theory(MTT) (Mel'cuk et al., 1988), provides a representation of coordinate structures. He do not show any symmetry in the dependency tree for conjuncts. He argues that coordination symmetry is only at the semantic level not at syntactic level. Hence, the MTT's representation of coordination looks like any other dependency tree without any symmetry. He further argues that the first conjunct is governed by the root and the second or all the other conjuncts are dependent on the first conjunct. Consider the following fig-2.14:

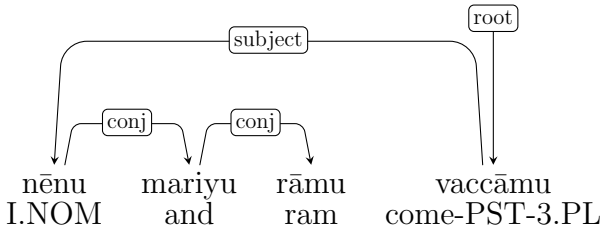


Figure 2.14: 'Melcuk's representation of coordination'

2.11.1.3 Hudson - Word Grammar

Word Grammar's analysis of coordination does not connect directly to the conjuncts but the proxy conjunction is introduced and is considered the head of the coordinate construction. Though both the conjuncts are placed on the same level, it is facilitated through the introduction of a proxy conjunction. (Hudson, 1984, pg-178).

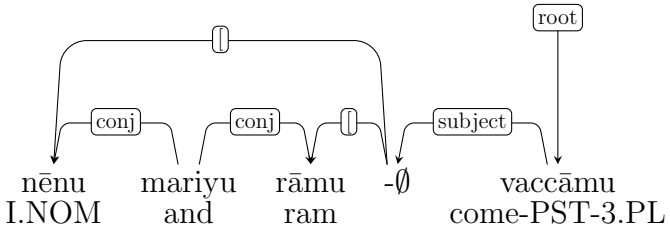


Figure 2.15: 'Hudson's representation of cocordination'

2.11.1.4 Timothy Osborne

In a paper titled 'Major constituents and two dependency grammar constraints on sharing in coordination' (2008), Timothy Osborne argues that unlike Tesniere's representation, both conjunts cannot be the head. He argues that only one conjunct (the left -most) is the head which is governed by the root. Hence, he connects only

the left-most element to the root and the other conjunct is connected through the conjunction as in the fig-2.16.

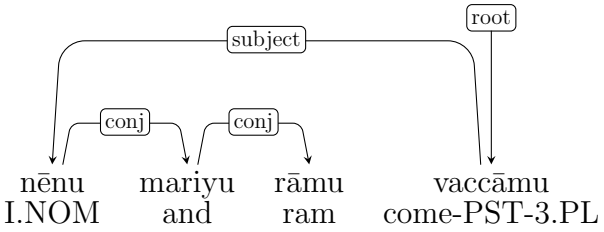


Figure 2.16: 'Timothy Osborne's representation'

2.11.1.5 Hyderabad Dependency Treebank for Hindi (HyDT)

HyDT (Bharati et al., 2012) makes the conjunction the head that is governed by the root node. Unlike all the above representations, conjunction is directly connected to the root. There is a subtle difference between HyDT and Hudson's representation in that Word Grammar introduces a proxy element whereas HyDT does not. This representation is similar to the one presented by Rosta et al. (2005). Consider the figure-(2.17):

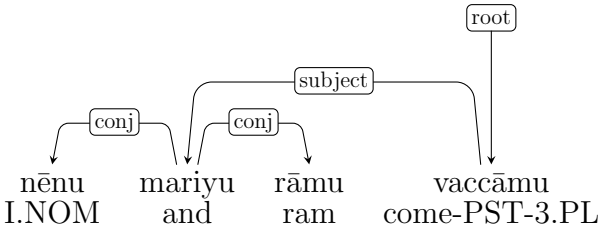


Figure 2.17: HyDT's representation of coordination'

2.11.1.6 Representation of Coordination in RBP

In Telugu, coordination is expressed through the conjunction 'mariyu' and an elongation of the last vowels of the conjuncts is often observed. But the use of conjunction is optional in Telugu. Sometimes, special characters like ',' is used but is still optional. The elongation of vowel after the conjuncts is often omitted in modern Telugu. In addition to this, when a coordinate construction has two conjuncts, the case suffixes are often observed only on the right-most element closer to the verb. Case markers are often omitted on the first conjunct.

Panchal and Kulkarni (2019)'s work on dependency analysis of coordination in Sanskrit provides insights for Telugu coordinate structures too. Taking insights from

the theories of Indian grammatical tradition the authors provide three features of coordination:

- There is no mutual expectancy between the conjuncts in the coordinate structure
- Any conjunct in the coordinate structure can be the head or both conjuncts headed by the conjunction can also be considered as the head of the coordinate structure
- conjunctive even if it functions as the head does not govern or governed by any element in the sentence

Based on the above observations, some observations about coordination in Telugu are made: (1). In Telugu, conjunction is almost always absent. Conjunction is expressed morphologically through elongation. (2). It is also observed that when nouns occur in coordinate construction, the right-most element which is the closest to the verb often take the case-suffix. For instance, *vāḍu hēmā lalitalatō bajāruki veḷḷāḍu* 'He went to the market with hema and lalita'. From these observations, it is decided to mark the right-most conjunct with overt morphological markers as the head of the coordinate constructions. In RBP, coordination is represented like in fig-(**??**)

(2.5)	*nēnū(mariyu)*	*nā*	*tammuḍu*	*sinimā-ki*	*veḷḷ-ā-mu*
	I.NOM	I-POSS	brother	cinema-DAT	go-PST-1.PL

'I and my brother went to a movie'

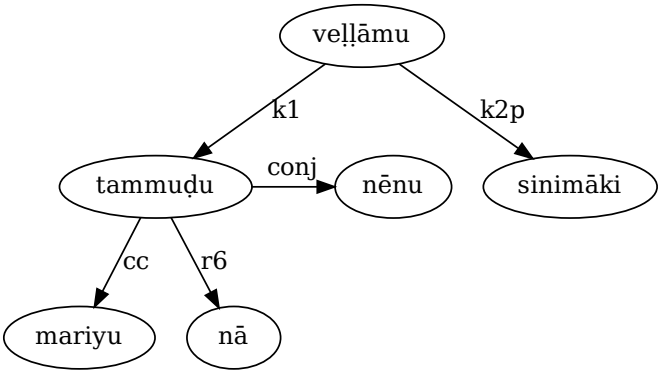


Figure 2.18: Dependency trees for (2.5)

2.11.2 Complement Clauses in Enhanced AnnCorra

Like in coordinate constructions, complementizer is not the head of complement clauses. *ani* in Telugu is the complementizer. In anncora, complementizer is considered as the head. However, as mentioned above, in RBP no functional elements can be the heads. So, in complement clauses, complementizer is just marked with the tag 'mark' to the finite verb similar to UD representation (De Marneffe et al., 2021). Consider the example (2.6):

(2.6)

rājeśwari	*vacc-iM-di*	*ani*	*vimala*
rajeshwari.NOM	come-PST-3.SG.F	QUO	vimala.NOM
grahiMc-iM-di			
realise-PST-3.SG.F			

'Vimala realised that Rajeshwari came'

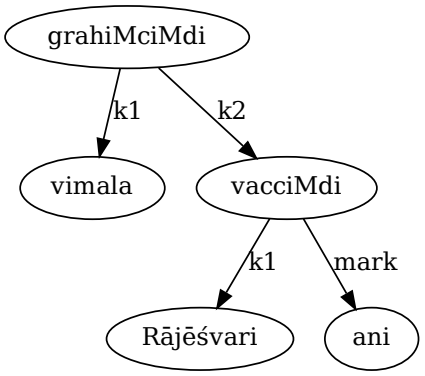


Figure 2.19: Dependency trees for (2.6)

2.12 Conclusion

This study adopts AnnCorra tags majorly but taking insights from Indian grammatical tradition and Universal Dependencies, changes in the tagset are made. For certain relations like coordination, verb-less sentences and complement constructions, the most suitable representation has been taken. We needed a tagset which represented major tags syntactico-semantically and do not heavily rely on semantics. Hence, we developed a tagset which is an amalgamation of the above mentioned tagsets. The detailed guidelines of dependency relations in Telugu are discussed in-detail in the next chapter.

Universal Dependencies	AnnCorra
nsubj	k1, pk1
nsubj:nc	k4a, r6v
nsubj:pass	k2
obj	k2
iobj	k4
csubj	k1
ccomp	k2
xcomp	vmod
obl	mk1,k2s,k2g,k2p,k3,k5,k5prk,k7,k7p,k7a,rd,rh,rt ras-k*,ras-neg,rsp
obl:caus	jk1
obl:tmod	k7t
obl:agent	k1
vocative	rad
dislocated	fragof
advcl	vmod, rh, rt
advl:cond	vmod
advmod	adv, rd, rsp, lwg__intf, vmod__adv,jjmod_intf, jjmod
discourse	sent_adv
aux	lwg__vaux
cop	root
mark	ccof,lwg_particle
nmod	nmod
nmod:poss	r6
nmod:cmp	k*u
nummod	enm
acl	nmod_relc, rs
amod	nmod_adj
det	nmod, nmod_wq
case	lwg_psp
conj	ccof
compound:svc	pof_cv
compound:lvc	pof
compound	pof_nn
compound:redup	pof_redup
list	-
parataxis	
orphan/ellipses	mrel
goeswith	-
punct	rsym

Chapter 3

Dependency Relations in Telugu

3.1 Introduction

In this chapter, we discuss the dependency relations that exist in a sentence in Telugu. This study frames guidelines that are best suited for marking dependency relations based on the syntax of Telugu. *AnnCorra* guidelines (Bharati et al., 2012), originally, framed for Hindi are taken as a base and Telugu-specific guidelines are compiled. This chapter discusses dependency relations in Telugu with explanations and their respective dependency trees. An attempt to explain and explicate dependency relations for parsing Telugu sentences. Though most of the relations and the hierarchy in dependency trees provided here are in accordance with *AnnCorra* guidelines, we have modified these guidelines for certain constructions like coordination, subordination, verbal modifiers, etc based on Telugu syntax. We have taken insights from the existing descriptive grammars of Telugu (Krishnamurti and Gwynn, 1985; Ramarao, 1975; Subbārāo, 2012).

3.2 Types of Dependency Relations

It is obvious that words in a sentence are related to other words in it. The relation between words are called dependencies in dependency grammar. Various kinds of dependency relations exist between different categories of words. These key relations are primarily divided into *kāraka* and Non-*kāraka* relations in Pāṇinian dependency tradition.

The word *kāraka* literally translates to 'a thing that gets or makes an action done or accomplished' (Kulkarni and Sharma, 2019). This can be referred to as participants of an action. The *Kāraka* theory was developed by Pāṇini in his seminal work, Aṣhṭādhyāyī which is considered the oldest among the Indian grammatical theories. Case markers i.e. *vibhaktis* serve the purpose of marking relationships between a noun and a verb or another noun in a sentence. The relations hence marked can be purely syntactic or morpho-syntactic or syntactico-semantic. Pāṇini's theory can be considered as a connect between

syntax and semantics based on the structural constraints imposed by a language. In a dependency structure, the 'verb' is considered as the vital element. The concept of verb centrality is a key component in dependency grammar(DG) wherein every dependency structure is solely dependent on the verb of a sentence[1]. The verb in a sentence represents an action carried out by various participants. These participants are of two types. **Type-1** denotes participants which are directly involved in the action. Direct participants are necessary for the sentence to be meaningful and act as a complete entity. **Type-2** includes participants which are indirectly involved in the action and provide extra, often more specific details. Type-1 relations are called as *kāraka* relations and type-2 are non-*kāraka* relations.

In addition to *kāraka* and non-*kāraka* relations, there are other dependency and non-dependency relations in a sentence. Relations between noun-adjective, verb-adverb, verb-verb etc are the other dependency relations and relations between conjunction and conjoined element, noun and verb in light-verb constructions etc are some of the non-dependency relations. In addition to this, some miscellaneous relations between indeclinables are also part of the guidelines. In this chapter, relations pertaining to the structure of Telugu namely, *kāraka*, non-*kāraka* relations, other dependency, non-dependency relations and miscellaneous relations are discussed in detail.

3.3 *kāraka* Relations

The type-1 relations or direct participants are classified as *kāraka* relations in Paṇinian dependency framework. *kāraka* relations are syntactico-semantic relations expressed through *vibhaktis* 'case-suffixes/post-positions' to capture dependencies between nouns and their corresponding verbs. The verb has an expectancy for its arguments which are typically nouns[2]. The Pāṇinian treatment of *kāraka* relations consider a system of default *vibhakti* for each relation. kāraka relations in Telugu include *kartā* 'roughly subject', *karma* 'roughly object', *karaṇa* 'instrument', *sampradāna* 'receipient/beneficiary', *apādāna* 'source' and *adhikaraṇa* 'locus'. These relations do have default *vibhakti*s in Telugu and its dependency relations are provided in table-(3.1).

[1]It should be noted that there are copula sentences which occur without overt verb in Telugu

[2]Here, nouns refer to any nominals including the lexical category of nouns, pronouns and number words

S.No	***kāraka* relations**	***default vibhaktis***
1	*kartā* 'roughly subject'	∅
2	*karma* 'roughly object'	*-ni/-nu*
3	*karaṇa* 'instrument'	*-tō*
4	*sampradāna* recipient/beneficiary'	*-ki/-ku*
5	*apādāna* 'source'	*nuMḍi/niMci/nuMci*
6	*adhikaraṇa* 'locus'	*-lō*

Table 3.1: *kāraka* relations and default *vibhaktis* in Telugu.

However, apart from these 6 prominent *kāraka* relations, other *kāraka*-related sub-relations are part of k-relations. The sub-relation to express comparison and similarity can occur with the existing *kāraka* relations and is called as *sādriśya*. It expresses with the tag `k*u`, here * can be replaced by *kāraka* relations as `k1u, k2u` etc. The list of *kāraka* relations and their sub-relations are provided in Table-3.2:

Sl.No.	***kāraka* Relations**	Tag	English Equivalent
1.	***kartā***	`k1`	Roughly equivalent to subject
2.	*prayōjaka kartā*	`pk1`	Causer
3.	*prayojya kartā*	`jk1`	Causee
4.	*madhyasta kartā*	`mk1`	Mediator causer
5.	*kartā samānādhikaraṇa*	`k1s`	Subject complement
6.	***karma***	`k2`	Roughly equivalent to object
7.	*gauṇa karma*	`k2g`	Secondary object
8.	Place as *karma*	`k2p`	Goal/Destination
9.	*karma samānādhikaraṇa*	`k2s`	Object complement
10.	***karaṇa***	`k3`	Instrument
11.	***sampradāna***	`k4`	Indirect object/Beneficiary
12.	*anubhava kartā*	`k4a`	Experiencer subject
13.	***apādāna***	`k5`	Source of separation
14.	*prakruti apādāna*	`k5prk`	Source material
15.	***adhikaraṇa***	`k7`	Location elsewhere
16.	*kālādhikaraṇa*	`k7t`	Location in time
17.	*deṣādhikaraṇa*	`k7p`	Location in space
18.	*sādriśya*	`k*u`	Similarity/comparison

Table 3.2: *kārakā* relations and their tags

3.3.1 *kartā* (k1) - 'Roughly subject'

The *kartā* is a relation which roughly corresponds to the subject that expresses an agent/doer/subject/experiencer etc., but cannot be concretely given any one of these tags. Kulkarni and Sharma (2019) provided illustrates as to why *kartā* is not just a 'subject' as popularly quoted in modern linguistics. It is argued that *kartā* performs various other functions like a doer, agent, instrument, experiencer depending on the verb of the sentence. An action is performed with the help of multiple participants and multiple sub-events. Among the participants, Pāṇini describes *kartā* as the most important and independent participant irrespective of the function it serves in a sentence (Kulkarni and Sharma, 2019).

Telugu is a nominative-accusative language which marks its subject in the nominative case. Typically, the nominative case is not marked explicitly on nouns and is null-marked -∅ in Telugu. However, in passive constructions, subject takes the post-position *cēta*. *kartā* agrees with the finite verb in Gender, Number and Person(GNP) in active construction[1].
kartā, is not just a syntactic relation, it is a syntactico-semantic relation which is determined by the *mukhya viṣeṣya* i.e. the 'root' of the sentence. Though by default, *kartā* is shown with -∅ marker(as shown in table-3.1), it occurs with non-∅ marker in certain constructions. Consider for instance, the following pair of sentences in Telugu,

(3.1) a. *rāmuḍu* *rāvanuḍi-ni* *caMp-ā-ḍu*
ram.NOM ravan-ACC kill-PST.3.SG.M
'Ram killed Ravan'

b. *rāmuḍi* *cēta* *rāvanuḍu* *campa-baḍ-āḍu*
ram by ravan-NOM kill-PASS-3.SG.M
'Ravan was killed by Ram.

The sentence-(3.1-a) is in active voice with the null-marked subject whereas in sentence(3.1-b) which is in passive voice, the subject is marked with the *vibhakti* '*cēta*'. In sentence((3.1-b)), the agent, *kartā* is identified taking into consideration the semantics of the verb. Similarly, there are other instances where *kartā* occurs in Telugu with different *vibhaktis*. *kartā* in various constructions like copula, transitive, ditransitive, passive and causative constructions are discussed in this section.

[1]In passive constructions, the semantic object agrees with the finite verb

3.3.1.1 *kartā* in Copula Constructions

Telugu is productive in copula construction. The verb *uM* 'to be/to exist' constitutes the copula verb in Telugu. The negation of copula is *lē* 'not to be'. In the copula constructions expressing equative sense, the verb is often dropped. The elements of a simple equational sentence are two noun phrases (one in subject and the other in predicative positions) or a noun phrase in the subject and an adjective in the predicative position. when the noun phrase in the subject position is in either first/second person, the agreement markers *-ni/nu, -vi/vu, -mu, -lu/ru* for the first person singular, second person singular, first person plural, second person plural are used respectively on the noun in the predicative position. These agreement markers are used to identify the equational relation in the parser. Example (3.2) consists of two noun phrases with subject in the third person and a noun phrase occurring in the predicative position. When an adjective occurs in the predicative position, third person agreement marker is reflected on the adjective.

As shown in examples-(3.2), (3.3), (3.4), the verb is absent. In such verbless constructions, a null-verb is introduced. The introduction of null-verb is the key to Pāṇinian dependency framework as dependency, in general, is verb-centric. Also, because negative equative constructions consist of a verb, we choose to introduce a null-root in case of verb-less affirmative equative constructions. The noun phrase with -∅ *vibhakti* in equative constructions in the subject position is marked as `k1` to the null verb, the predicate noun as `k1s`. See the example (3.2) and its respective dependency tree[1] in (3.1):

(3.2) *jayalalita* *pramukha* *naṭi*
jayalalitha.NOM popular actress
'Jayalalitha is a popular actress'

Example (3.3) is an equative construction with a noun phrase and an adjective phrase in the predicative position but in Telugu even the adjective phrase also requires an agreement. Predicative adjective takes the agreement of the noun phrase it describes. Here, it is *-di* suffix which denotes 'third person, singular, non-masculine' marker. Consider (3.3) and its dependency tree in figure (3.2):

(3.3) *jamuna* *teliv-aina-di*
jamuna.NOM smart-ADJ-3.SG.F
'Jamuna is smart'

Consider (3.4) for an example of equative construction describing the quality of the subject (Ramarao, 2017) and its dependency tree in figure 3.3:

[1]In dependency trees, a fixed word order is not followed, however, a strict modifier-modified order is not violated

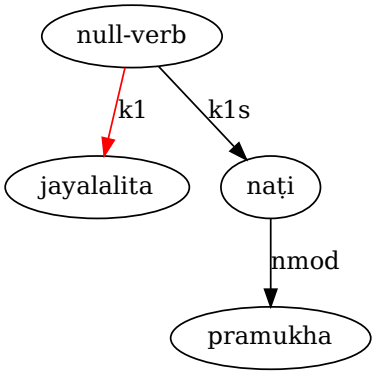


Figure 3.1: Dependency tree for the example-(3.2)

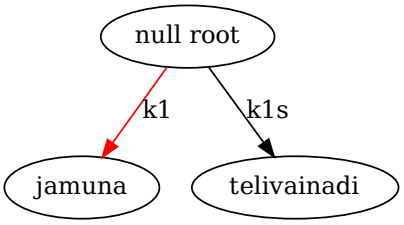


Figure 3.2: Dependency tree for 3.3

(3.4) *ā* *vajraM* *viluva* *padi* *kōṭ-lu*
that diamond value ten crore-PL
'The value of that diamond is 10 crores'

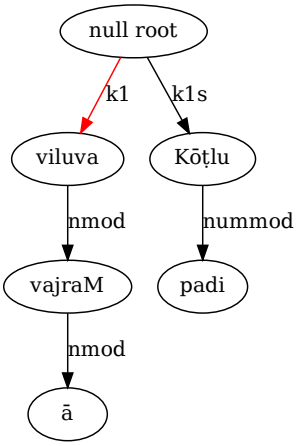


Figure 3.3: Dependency structure for the sentence- 3.4

Copula in Telugu also denotes three meanings, namely, existential, stative and possessive. In existential constructions, a noun and a noun represents 'the state of being'. In constructions denoting possession, a noun of possession and possessor are present. Noun denoting possession is in the nominative case.
In all these constructions, the noun with nominative case marker that agrees with the verb in GNP is considered the *kartā*.

Existence

(3.5) *vāḍu* *haidarābādu-lō* *unn-ā-ḍu*
I-NOM hyderabad be-PST-3.SG.M
'He is in Hyderabad'

Possession

(3.6) *nā-ku* *illu-∅* *uM-di*
I-DAT house be-3.SG.M
'I have a house'

Stative

(3.7) *pratāp* *saMtōṣaM-gā* *unn-ā-ḍu*
pratap-NOM happy-ADV be-PST-3.SG.M

'Pratap is happy'

In above illustrations, *nēnu*, *illu* and *pratāp* in examples-3.5, 3.6, 3.7 respectively, with -∅ suffix exhibiting GNP agreement with the copula verb are marked as *kartā*. Copula is not always overt in Telugu. Copula verb can be dropped leading to verbless sentences. Verbless constructions are quite prevalent in Telugu. Copula verb is optionally dropped in existential sentences when the 'existence' is known (Krishnamurti and Gwynn, 1985). Verb is also dropped in equational sentences (Ramarao, 2017).

3.3.1.2 *kartā* in Intransitive and Transitive Constructions

Intransitive verbs which are also called [1]*akarmaka* verbs contain a single obligatory *kārakā*. The noun phrase that occurs in intransitive constructions with -∅ suffix[2] which agrees with the verb in GNP is the typical case of *kartā*. Consider the following example-(3.8) for a typical instance of *kartā*

(3.8)	*iMdira*	*nēla*	*mīda*	*paḍuku-M-di*
	iMdira.-∅	floor-OBL	on	sleep-PST-3.SG.F
	'Indira slept on the floor'			

'iMdira' in the example-3.8 has a -∅ suffix and a GNP agreement with the verb 'sleep'. Hence, it is given a tag '*kartā*(k1)'.

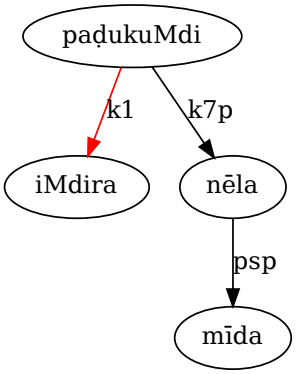


Figure 3.4: Dependency structure for the sentence- 3.8

Transitive verbs in active voice require an obligatory object in a sentence. In such constructions, typically, two *kārakās* can occur, a *kartā* and a *karma*. The

[1]A Sanskrit term for intransitive verbs

[2]Note: There may be other noun phrases in the sentence with -∅ suffix in the sentence, which may not be the *kartā*. Then, GNP agreement must be considered as the primary cue.

karma kārakā occurs with default *vibhakti* '-ni/nu', however, with certain nouns, they occur with null-marking. This phenomenon called 'differential object marking' (for more discussion, see (3.3.2). In cases where a noun phrase with -∅ *vibhakti* and *-ni/nu vibhakti* occur in an active voice construction, a noun phrase with -∅ null-marker and GNP agreement with the verb is considered as the *kartā*. In case, both noun-phrases have null suffix, the noun phrase higher in animacy scale which agrees with the verb in GNP is given the *kartā* relation . Consider the example-(3.9)

(3.9) *sādhana* *pillā-ḍi-ni* *tiṭṭ-iM-di*
sadhana.NOM child-M-ACC scold-PST-3.SG.F
'Sadhana scolded the child'

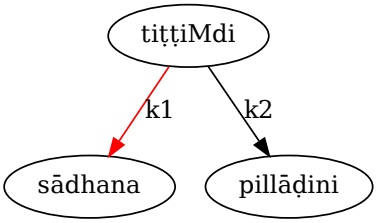


Figure 3.5: Dependency structure for the sentence- (3.9)

In (3.9), *sādhana* is a null-marked noun phrase that agrees with the verb *tiṭṭu*'scold'. Hence, the tag 'k1' is marked in the figure-(3.5).

Miscellaneous cases:

(3.10) *nuvvu* *pāṭa* *pāḍ-āli*
I.NOM song sing-HORT
'You have to sing a song'

(3.11) *vāḍu* *īrōju* *rā-vacc-ēmo*
He.NOM today come--DUB
'He may come today'

(3.12) *nuvvu* *mā* *ūri-ki* *rā*
you our village-DAT come-IMP
'You come to our village'

(3.13) *nā-ku* *ā* *viṣayaM* *telusu*
I-DAT that matter know-STAT
'I know that fact'

(3.14) *vāḍu* *pani-ki* *veḷḷavalasi-vacciMdi*
He.NOM work-DAT go-OBL-PST-3.SG.N
'He had to go to work'

In the above mentioned examples, agreement with the *kartā* kāraka is absent. Hence, the position of the noun phrase and the semantics of nouns(hierarchy) (See chapter-4 for a detailed explanation) is considered as the *kartā* . Based on the canonical word order of Telugu, noun phrase with -∅ which is not the closest to the verb is marked as the *kartā*.

3.3.1.3 *kartā* in Passive Constructions

In positional languages, subject and object interchange occurs for passive constructions. In free-order languages like Telugu, the active form of the verb form changes to passive by adding the suffix *baḍu*. However, subject and object are not always interchanged for position but the case marker *cēta/cē* is added to the subject and the accusative case-marker on object is dropped. This noun phrase with post-position '*cēta*' is considered the *kartā* in passive constructions. In passive constructions, *kartā* does not agree with the finite verb in GNP. Observe the set of examples in (3.15) and the dependency tree in figure (3.6):

(3.15) a. *cēkūri* *telugu* *vākyaM* *rās-ā-ru*
cekuri.HON telugu sentence-ACC write-PST-3.SG.HON
'Mr.Cekuri wrote the Telugu *vākyaM*'

b. *telugu* *vākyaM* *cēkuri* *cēta* *rāya-baḍ-iM-di*
telugu sentence cekuri by write-PASS-PST-3.SG.N
'Telugu *vākyaM* was written by Cekuri'

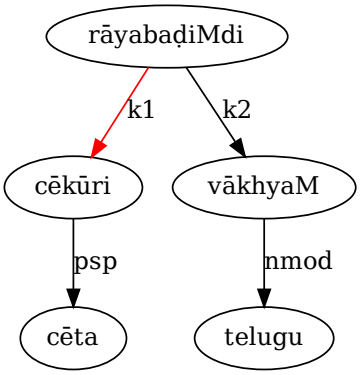


Figure 3.6: Dependency structure for the sentence- (3.15-b)

3.3.1.4 *kartā* in Causative Constructions

In Telugu, causative verb stem is formed by a adding the suffix *-iMcu* to a transitive verb. Causative constructions requires three arguments, namely, the causer agent, the actor agent (causee) and an object (Krishnamurti and Gwynn, 1985). In addition to this, mediator agent also occurs occasionally. Hence, the causer , the causee and the mediator causer are all marked with three different tags viz., *prayojaya kartā* (pk1), *prayojya kartā* (jk1), *madhyasta kartā* (mk1) respectively.

1. ***prayojaka karta***(pk1) -'Causer' & ***prayojya kartā*** (jk1) - 'Causee'
   The causer in the causative constructions is marked with *prayojaka kartā*. The agent in causative construction that causes the action to be done is pk1. Typical features of *kartā* like the agreement with the main verb, -∅ case marking can be observed. Consider the set of examples in (3.16). In (3.16-a), *lalita* is the *kartā*. However, in (3.16-b), *lalita* is the causer and is given the pk1 relation to the verb & *kamala* is the causee.

(3.16) a. *lalita* *hōMvark* *cēs-iM-di*
lalita.NOM homework.ACC do-PST-3.SG.F
'Lalita did her homework'

b. *lalita* *kamala* *cēta* *hōMwark*
lalita.NOM kamala by homework.ACC
cē-iMc-iM-di
do-CAU-PST-3.SG.F
'Lalitha made Kamala do her homework'

The causee in causative constructions is marked as the *prayojya kartā*. The actual doer of the activity or the action agent is jk1. In (3.16-b), *kamala* is the actual doer of the activity *cēyu* 'do'. Causee is marked with *cēta* post-position in Telugu.
Ramarao (1975) illustrates instances of *cēta* replaced by *-ni/nu* suffix. Consider (3.17)

(3.17) *nā* *snehitu-rālu* *nan-nu* *cadiv-iMc-iM-di*
I.POSS friend-F I-ACC read-CAU-PST-3.SG.F
'My friend made me study'

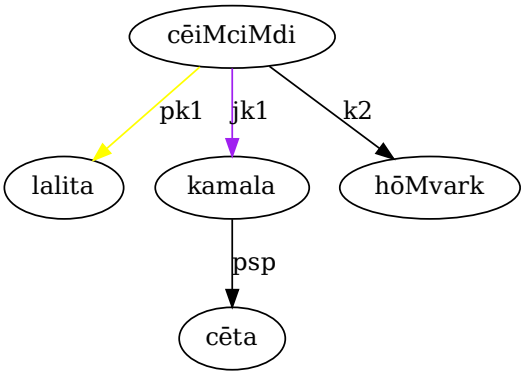


Figure 3.7: Dependency structure for the sentence- (3.16-b)

As shown in (3.17), the actual action is performed by *nēnu* 'I' and the activity is caused by *snehiturālu* 'friend'. It is observed that semantic ambiguity occurs in this sentence, this sentence can either mean 'my friend helped me study' or 'my friend help me financially for my education' (Ramarao, 1975). However, this is not the concern of this study. The *-ni/nu* marked nominal is given the tag *k2* considering the marker.

2. ***madhyasta kartā*(mk1)- 'Mediator causer'**
   In some cases, along with the primary causer, a mediator causer '*madhyasta kartā*' (mk1) also occurs.

   (3.18) *ramēś* *pillāḍi* *dvārā* *rāṇi-cēta*
   ramesh.NOM child-OBL through rani.INS
   uttaraM rāyiMcāḍu
   letter write-CAUS-PST-3.SG.M
   'Ramesh made the child to make Rani write the letter'

Hence, *rameś* is pk1, *pillāḍiki* is mk1 and rāṇi is jk1.

3.3.1.5 Clausal *kartā*- 'Clausal Subject'

The *kartā* in Telugu is sometimes realised as a clause and is also marked as 'k1'. Consider for the example of clausal *kartā* (3.19):

(3.19) *nēnu* *caduvu-kōv-aḍaM* *vāḍi-ki* *iṣṭaM-lēdu*
I.NOM study-REF-GERUN he-DAT like-NEG
'He does not like me studying'

As shown in figure (3.9) , *caduvukāvaḍaM* is tagged as k1 to the main verb.

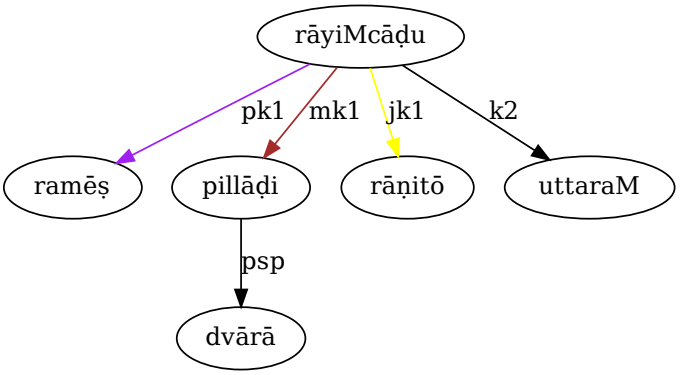


Figure 3.8: Dependency structure for the sentence- (3.18)

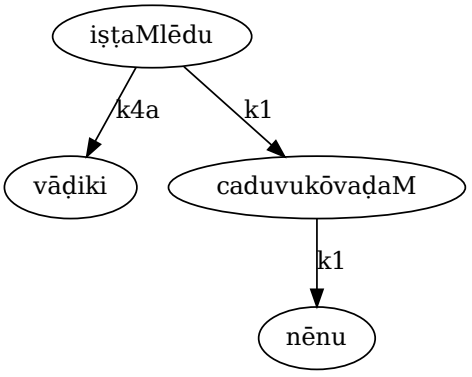


Figure 3.9: Dependency structure for the sentence- (3.19)

3.3.1.6 *katṛu samānādhikaraṇa* (k1s) - 'Subject Equivalence'

The *kāraka* relation of *samānādhikaraṇa* indicates 'equal to a locus of something'. When a noun phrase complements/equals any other relation in the sentence, this relation is used. This is of typically two types : *katṛu samānādhikaraṇa* (k1s) and *karma samānādhikaraṇa* (k2s).

The relation k1s is used to mark the complement that correspond to the subject in the same sentence. In Telugu, this is expressed using a predicative adjective/noun. A noun phrase in the subject and an adjective in the predicative position. when the noun phrase in the subject position is in either first/second person, the agreement markers *-ni/nu, -vi/vu, -mu, -lu/ru* for the first person singular, second person singular, first person plural, second person plural are used respectively on the noun in the predicative position. These agreement markers are used to identify the equational relation in the parser. Example (3.20) consists of two noun phrases with subject in the third person and an adjective phrase occurring in the predicative position. When an adjective occurs in the predicative position, third person agreement marker is reflected on the adjective k1s is usually observed in equative constructions with the absence of copula verb in Telugu. Copula verb is absent in affirmative constructions whereas it is present in negative constructions. In the absence of copula, a 'null verb' is introduced. See table-(3.3) for agreement pattern on predicative adjective or noun.

Person	**Marker**	**Example**
First person, singular	*-ni*	*nēnu āḍapilla**nu*** 'I am a girl'
First person, Plural -Inclusive/Exclusive	*-mu*	*mēmu āḍapillala**mu*** 'We are girls'
Second Person, singular	*-vi/vu*	*nuvvu āḍapilla**vi*** 'You are a girl'
Second person, plural	-∅	*mīru āḍapillalu* 'You are girls'
Third person, singular	-∅	*āme āḍapilla* 'She is a girl'
Third person, plural	-∅	*vāḷḷu āḍapillalu* 'They are girls'

Table 3.3: Agreement marking on predicative nouns

Consider the example- (3.20) for an example of *katru samānadhikarana.*

(3.20) *sīta* *maMci-di*
sita.NOM good-3.SG.F
'Sita is a good human'

In (3.20), *maMcidi* 'good human' is considered as k1s as it is the complement of *kartā* 'sita'. Both k1 & k1s are connected to the null verb.

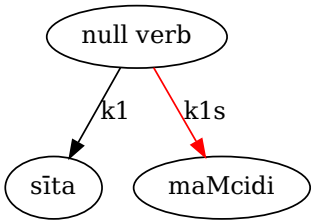


Figure 3.10: Dependency structure for the sentence- (3.20)

(3.21) *jayalalita* *pramukha* *naṭi*
jayalalitha.NOM popular actress
'Jayalalitha is a popular actress'

3.3.2 *karma*(k2)- 'Roughly Object'

In Pāṇinian grammatical tradition, *karma* is defined as the 'most desired participant by the *kartā*'. The relation *karma* roughly corresponds to a patient/theme in modern linguistic theories. It is used to mark the direct object relation of the transitive verb. The accusative case suffix -*ni/nu* is used to mark the direct object in Telugu. However, direct object also occurs with -∅ in certain contexts.
Accusative marker is obligatorily used for animate objects whereas it is optional for inanimate objects in Telugu. This phenomenon of marking objects differently is called the differential object marking (DOM). DOM is a phenomenon first noted by (Bossong, 1985). DOM depends on factors like animacy, specificity and definiteness of the object (Aissen, 2003; Lidz, 2006; Subbārāo, 2012). Here, we provide examples according to the order of the animacy scale:

- **Case-1**
  When the object is a noun[+animate, +human, +definite], it is marked overtly with accusative marker(-*ni/nu*) in Telugu. In the example (3.22), the object *ravi* obligatorily takes the -*ni/nu* suffix.

  (3.22) *sīta* *ravi-ni* *cūs-iM-di*
  she.NOM ravi-ACC see-PST-3.SG.F
  'She saw Ravi'

  Similarly, when the object is noun[+animate, -human, +definite], accusative case is obligatorily used. In the example (3.23), *kukka* is overtly marked with accusative case '-*ni*'.

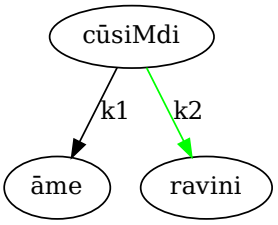


Figure 3.11: Dependency structure for the sentence- 3.22

(3.23) *sīta* *oka* *kukka-ni* *cūs-iM-di*
sita one dog-ACC see-PST-3.SG.F
'Sita saw the dog'

- **Case-2**

  When the object is noun[-animate, -definite] it is not marked with the accusative case (*-ni/nu*) in Telugu. The marker -∅ is used in such contexts (see example-3.24).

  (3.24) sīta sinimā cūs-iM-di
  Sita.NOM cinema-ACC see-PST-3.SG.F
  'He read a book'

  (3.25) vāḍu ī pustakān-ni cadiv-ā-ḍu
  he.he this book-ACC read-PST-3.SG.M
  'He read this book'

  In Telugu, the accusative case marker can be used to denote a sense of specificity. Hence, when accusative case is marked with objects of [-animate] feature, it indicates specificity (see 3.25).

- **Case-3**

  Nouns with [-animate] objects are obligatorily marked with accusative case when the object denotes higher cultural terms (Ramarao, 2017, pg-35) as in (3.26), (3.27)

  (3.26) *nēnu* *nā* *deśān-ni* *prēmis-tunn-ā-nu*
  I.NOM my country-ACC love-PROG-1.SG
  'I love the country'

  (3.27) *āme* *nāṭya* *kaḷa-nu* *kāpāḍu-tuM-di*
  I.NOM dance art-ACC save-HAB-3.SG.F
  'She saves the art of dance'

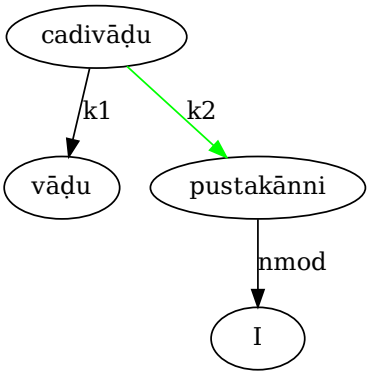


Figure 3.12: Dependency structure for the sentence- (3.25)

When the verb belong to a specific verb type like [+hit] (e.g. kick, beat, hit) or [+positive admire] (eg. love, like, appreciate) (Levin, 1993, pg-69), the object of such constructions, inspite of being [-animate], obligatorily takes an accusative case marker. See (3.28) & 3.29:

(3.28) *ēnugu* *ceṭṭu-ni* *viric-iM-di*
elephant.NOM tree-ACC break-PST-3.SG.N
'The elephant the tree'

(3.29) *kiṭṭu* *vāḍi* *udyōgānni* *prēmis-tunn-ā-ḍu*
kittu.NOM he.POSS job-ACC love-PST-3.SG.M
'Kittu loves his job'

3.3.2.1 *karma* in Double Object Constructions - *gauṇa karma*(k2g) (Secondary karma)

In double object constructions, two arguments act as objects. One among the two objects is marked with *-ni/nu* suffix and the other with the -∅ suffix. The argument which has undergone the effect of action is marked as the *karma* whereas the other who is benefited is marked as ***gauṇa karma*(k2g)** (Kulkarni, 2021b). The relation *gauna karma* is observed with noun[+animate] and *mukhya karma* is noun[-animate]. Double objects occur with communication verbs (Levin, 1993) like 'ceppu', 'ivvu', 'anu', 'māṭlāḍu, etc,. See example-.

(3.30) *ramya* *raghava-ni* *praśna* *aḍig-iM-di*
ramya.NOM raghava-ACC question-ACC ask-PST-3.SG.F
'Ramya asked Raghava a question'

In the construction (3.30), both *raghava* and *pra'sna* are marked for accusative case (one overt (*raghava-ni*) and the other covert (-∅)).It should be noted that *mukhya karma* is [-animate] and *gauṇa karma* is [+animate]. Animacy is used as an extra-linguistic cue to mark k2g.

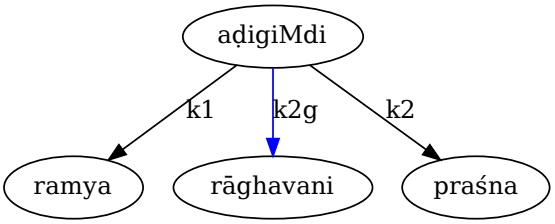


Figure 3.13: Dependency structure for the sentence- (3.30)

3.3.2.2 *karma* in Passive Constructions

In passive constructions, the noun phrase with -∅ marker that agrees in gender, number, person (GNP) with the verb is tagged as the *karma*(k2) (see 3.3.1.3 for more details). Consider (3.31)

(3.31) *sīta* *cēta* *vāḍu* *koṭṭa-baḍḍ-ā-ḍu*
sita by he.NULL beat-PASS-PST-3.SG.M
'He was beaten by Sita'

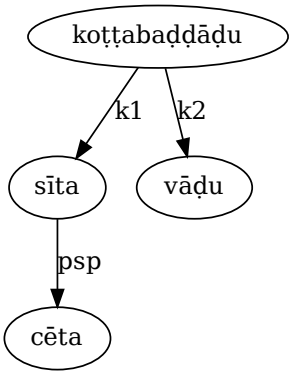


Figure 3.14: Dependency structure for the sentence- (3.31)

3.3.2.3 *karma* in Causative Constructions

In causative constructions, as mentioned in section-(3.3.1.4), three arguments viz., active subject, causative subject and an object occurs. The noun phrase in the object position with *-ni/nu/-∅* without GNP agreement with the verb is marked as k2 .

(3.32)

prajalu	*nāyakula*	*cēta*	*panulu*
people.NOM	leaders-OBL	by	work-PL
cēy-iMc-āru			
do-CAU-PST-3.PL			

'People made leaders do the work'

In sentence-(3.32), *panulu* is considered the *karma* of the verb *cēyiMcu*. Consider the dependency tree 3.15:

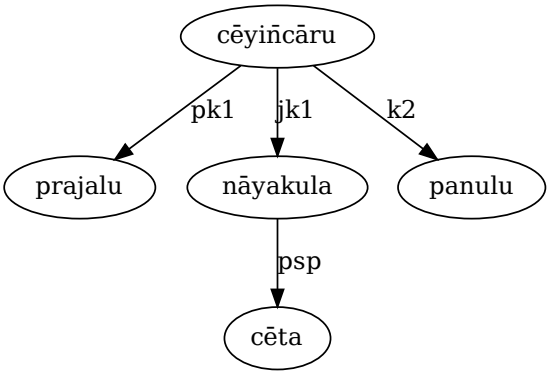


Figure 3.15: Dependency structure for the sentence- 3.32

3.3.2.4 *karma* in Reflexive/Reciprocal Constructions

Reflexive constructions in Telugu are formed using a verbal reflexive/reciprocal verbal stem '*-konu*' obligatorily and a nominal reflexive optionally.
The nominal reflexive occur in the bipartite[1] structure as in 'pronoun+case marker pronoun' (eg. *tana-ni (tānu)*) in which the second part of the pronoun can also be optional. The nominal reflexive cannot occur in the subject position. The first part of the reflexive pronoun takes the case marker of the object or indirect object depending on the position it occurs. The second part of the reduplicated nominal copies case of the subject. This phenomenon of taking the case of the subject is called as case-copying (Subbarao and Murthy, 2000).

[1]having or consisting of two parts

(3.33) *prēma* *tana-ni* *tānu* *meccu-kuM-di*
prema.NOM she-ACC she.NOM praise-REF-3.SG.F
'Prema praised herself'

(3.34) *nēnu* *nan-nu* *nēnu* *addaM-lō* *cūsu-konn-ā-nu*
I.NOM I-ACC I.NOM mirror-LOC see-REF-PST-1.SG
'I saw myself in the mirror'

In reflexive constructions like (3.33, 3.34), the reduplicated reflexive pronoun is given the tag `pof_redup` and the one marked with accusative case is marked as *karma*.

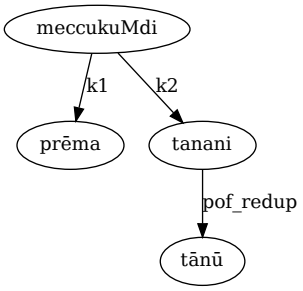


Figure 3.16: Dependency structure for the sentence- (3.16)

3.3.2.5 Goal/Destination as *karma*(k2p)

The case relation k2p indicates the place which functions as a goal or destination with motion verbs (eg: *vaccu* 'come', *veḷḷu'*,'go' *cērukonu* 'reach', *bayaludēru* 'start', *pō* 'go'). This is a special case of *karma* that takes a dative marker *-ki/ku* when occurs exclusively with motion verbs. Hence, it is considered a special kind of *karma* and sub-categorized under it with the tag `k2p`.

(3.35) *aśok* *āfīsu-ku* *veḷḷ-ā-ḍu*
ashok.NOM office-DAT go-PST-3.SG.M
'Ashok went to office'

In sentence-(3.35), *āfīsu-ku* 'to the office' has a *-ku* suffix and functions as a destination of the verb 'go'. Hence, the tag k2p tag is given.

(3.36) *nēnu* *sāyaMtrāniki* *illu* *cērukuM-tā-nu*
I.NOM evening-DAT home reach-PST-1.SG
'I reach home by evening'

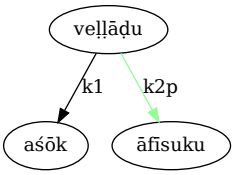


Figure 3.17: Dependency structure for the sentence- (3.35)

However, in the example-(3.36), *illu* functions as the destination but occurs with a null marker. Here, the semantics of the verb is taken as a cue to tag `k2p`.

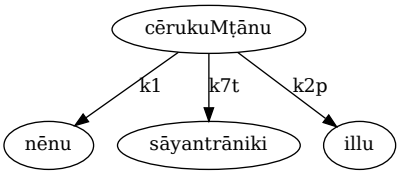


Figure 3.18: Dependency structure for the sentence- (3.35)

Also, verbs consisting of inherent motion like board, climb etc. the case suffix *-ni/nu* used to indicate location. Consider (3.37):

(3.37) *ādya* *ēnugu-nu* *ekk-iM-di*
Aadya elephant-ACC climb-PST-3.SG.F
'Aadya climbed the elephant'

In 3.37, 'ēnugu' is the locus of the activity climbing (a motion verb). Hence, it is marked with k2p[1].

3.3.2.6 Clausal *karma* -'Clausal object'

A clause can function as a *karma* in Telugu. In certain constructions, the finite verb of a subordinate clause can serve the object function of the matrix clause. This is a usual phenomenon in complement clauses.
When the subordinate verb functions as an object, the root of the clause is connected to the finite verb which expects the *karma*. Consider the example-(3.38):

(3.38) *pranav* *pāpā-ni* *cadavam-ani* *cepp-ā-ḍu*
pranav child--MAS-ACC study-QUOT tell-PST-3.SG.M
'Pranav told the child to study'

[1]This can still be a topic of discussion as *ēnugu-nu* can act as just a *karma* of the verb *ekku*

Example-(3.38) is a complement clause construction consisting of two clauses. The matrix clause '*ceppu*' 'tell' has '*Pranav*' as `k1` and the verb 'caduvu' as the *karma*. Hence, the verb of the clause is taken as the head and connected to the matrix verb with the relation `k2`. The internal argument within a clause is connected to the head. Here, *pāpani* functions as a *karma* to the verb *caduvu*.

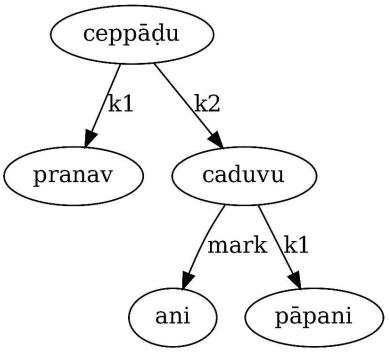


Figure 3.19: Dependency structure for the sentence- (3.38)

3.3.2.7 *karma samānādhikaraṇa* (`k2s`) - 'Object Equivalence'

The relation refers to the complement of *karma*. In a sentence, when a word refers/corresponds to the object, this relation is used. The tag `k2s` is used to mark this relation. Object complement is marked using *-ni*/-∅ suffix in Telugu. In object complement constructions, the order of object and its complement is very important. Whatever comes first in a second remains the object and the later one is treated as the complement.

(3.39) *rāmuḍu* *rāyi-ni* *maniśi-ni* *cēs-ā-ḍu*
Ram-∅ stone-ACC man-ACC do-PST-3.SG.M
'Ram turned a stone into a human'

As shown in (3.39), the object complement *maniśi* 'human' is marked with the suffix *-ni*, preceded by a *karma* of the verb '*ceyyi*' i.e. *rāyi* with the similar suffix and is tagged `k2s`.

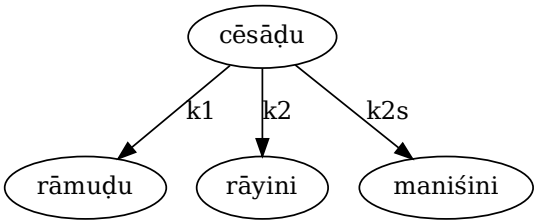


Figure 3.20: Dependency structure for the sentence- (3.39)

3.3.3 *karaṇa*(k3) - 'Instrument'

In Pāṇinian grammatical tradition, *karaṇa* is defined as 'the participant which is most instrumental in the accomplishment of the action'. It is a *karaka* relation referring to a noun phrase that acts as an instrument in carrying out the action intended by the verb. The tag k3 is used to mark this relation. In Telugu, *-tō* marker is used to indicate the relation of instrument.

(3.40) *līla* *peMsil-tō* *bomma* *gīs-tuM-di*
leela.NOM pencil-INS picture.ACC draw-PROG-3.SG.F
'Leela is drawing a picture with a pencil'

The word *peMsil* 'pencil' in the example-(3.40) acts as an instrument and is tagged with k3 relation to its head. Consider the figure-(3.21):

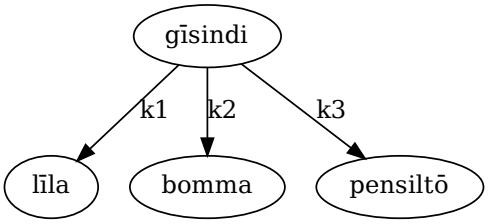


Figure 3.21: Dependency structure for the sentence- (3.40)

3.3.4 *sampradāna*(k4) - 'Beneficiary/Recipient'

The indirect object of the ditransitive verb is marked using the *sampradāna* relation. The *kāraka* relation of *sampradāna* denotes the recipient/beneficiary of the verb and

is marked with the tag `k4`. In Telugu, the default *sampradāna* marker is *-ki/ku*. Consider the example-(3.41):

(3.41) *amma* *atani-ki* *annaM* *peṭṭ-iM-di*
mother.NOM he-DAT rice-ACC keep-PST-3.SG.F
'Mother served him rice'

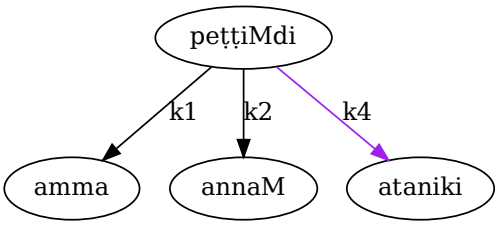


Figure 3.22: Dependency structure for the sentence- (3.41)

In certain cases, the *sampradāna* relation is realised with the case marker *-tō* in Telugu. Communicative verbs like *ceppu* 'to tell', *māṭlādu* 'to talk' optionally assigns either *vibhakti -ki* or *-tō* to express `k4`. See example-(3.42).

(3.42) *nēnu* *atani-tō/atani-ki* *ā* *viṣayaM* *cepp-ā-nu*
I.NOM he-ASS/he-DAT that matter say-PST-1.SG
'I told him that matter'

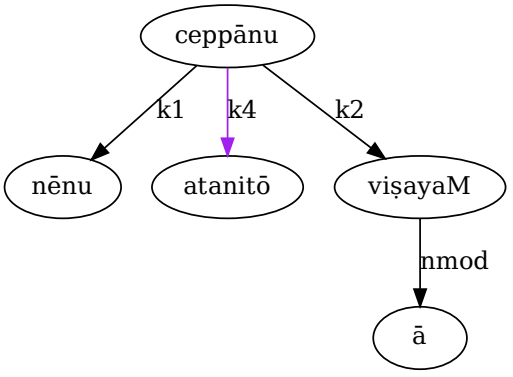


Figure 3.23: Dependency structure for the sentence- (3.42)

3.3.4.1 *kartā* in Non-Nominative Subject Constructions (k4a)

Telugu is productive in non-nominative subject constructions (Subbārāo, 2012). The most commonly used non-nominative subject is the *dative subject* construction. Other non-nominative subject constructions include locative and instrumental case marked subjects. Though, the case marker is in dative (with case-suffix *-ki/ku*) or locative, it serves various functions in a sentence. In non-nominative subject constructions, Subbārāo (2012), argues that the verb is usually intransitive. Hence, the noun that agrees with the verb is marked as the actual *kartā* [1] and the other subject which is not in nominative is marked according to the dependency relation it corresponds to. Examples of each type is provided below:

(i). Dative subject

A noun phrase with the dative suffix *-ki/ku* exhibits the most ambiguous nature in Telugu. Dative subjects as illustrated by Subbārāo (2012) serves various functions depending on the verb as exemplified below:

- **Psychological state**

(3.43) ***srīnu-ku*** *kōpaM* *vacc-iM-di*
srinu.DAT anger.NOM come-PST-3.SG.N
'Srinu is angry'

- **Kinship**

(3.44) ***nā.ku*** *iddaru* *akka-lu* *unnāru*
I.DAT two sister-PL.NOM be-PST-3.PL
'I have two sisters'

- **Physiological state**

(3.45) ***atani-ki*** *kālu* *virig-iM-di*
he.DAT leg.NOM break-PST-3.sg.N
'His leg broke'

- **verbs of cognition**

(3.46) ***ravi-ki*** *kamala* *telusu*
ravi-DAT kamala.NOM know
'Ravi knows Kamala'

[1] Here, in this guidelines, we consider the nominal that agrees with the verb as the *kartā*. However, it can be observed in the examples that it is not that independent kāraka but still is considered as the *kartā*. This can be further discussed and must be given an appropriate tag in the future study

- **Part-whole relationship**

(3.47) ***vāḍi-ki*** *āru* *vēḷḷu* *unnāyi*
he.DAT six fingers be-PRS-3.PL
'He has six fingers'

In the above illustrations, the experiencer who semantically subjects dative-marked. They are tagged as `k4a`. The theme is nominative marked and agrees with the finite verb. They are marked as `k1`. The other noun-phrase with *-ki/ku* suffix is given a different relation (Here, experiencer subject) based on the verb.

1. ***anubhava kartā*(`k4a`)- 'Experiencer subject'**

The relation of *anubhava kartā* was originally not part of the Pāṇinian framework, but is a special relation introduced for Indian languages (Bharati et al., 2012). This is usually used for non-nominative subject constructions. These constructions are prominently observed with verbs of experience, cognition, kinship relations, possession etc. The examples (3.43) to (3.47) constitute *anubhava karta* with *-ki/ku* suffix. These case relations are dependent on the semantic properties of the nouns and verbs involved in it. Consider various examples of dative-subject constructions given below. Also observe the dependency tree for 3.43 in figure 3.24 to see how *anubhava kartā* is represented in a tree:

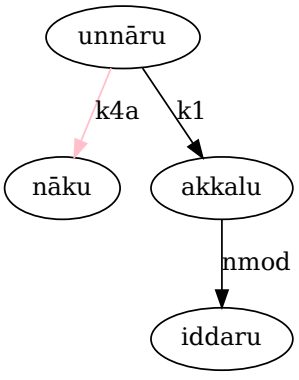


Figure 3.24: Dependency structure for the sentence- 3.44

It is observed from the above examples that dative marker in Telugu is quite ambiguous and require number of linguistic ques to identify the correct relation

between the verb and the noun carrying dative case. Some nouns with dative suffix can also be possessors but to maintain consistency non-nominative constructions with *-ki/ku* suffix are marked as `k4a`. It will be helpful in higher level applications like MT.

(ii). Locative subject

(3.48) *nā* *deggara* *paMḍlu* *unn-ā-yi*
I-POSS near fruits be-PST-3.PL.N
'I have fruits'

(3.49) *atani-ki* *illu* *uM-di*
he.DAT house.NOM be.3.SG.N
'He has a house'

In 3.48, the noun phrase that agrees with the verb having the -∅ is considered the *kartā*. So, *paMḍlu* 'fruits' is considered the *kartā* in (3.48). Likewise, in 3.49, *illu* is the *kartā*.

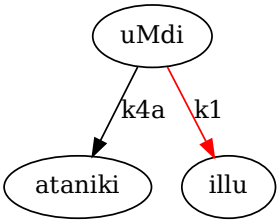


Figure 3.25: Dependency structure for the sentence- 3.49

3.3.5 *apādāna*(k5) - 'Source'

Pāṇini defines *apādāna* as the 'participant which is fixed when there is a movement away'. The ablative case marker *nuMḍi/niMci/nuMci* 'from' in Telugu carries the dependency relation of *apādāna* to the verb. This relation denotes the semantic role of 'a source of separation'.

(3.50) *rāju* *maisuru* *nuMḍi* *ninna* *vacc-ā-ḍu*
Raju.NOM mysore from yesterday come-PST-3.SG.M
'Raju came from Mysore yesterday'

In the example-3.50, *mysore nuMḍi* is place from which the agent is separated. Hence, the tag `k5` is given to *mysore*. Note that *maMdi* is written separately in

Telugu, hence it is attached with the tag postposition(psp).

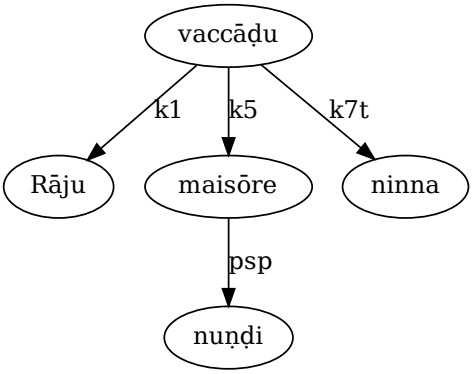


Figure 3.26: Dependency structure for the sentence- 3.50

The postposition *nuMḍi* 'from' also occurs with non-place nouns.

(3.51) *ceṭṭu* *nuMḍi* *ākulu* *rālāyi*
tree.NOM from leaves.ACC fall-PST-3.PL.N
'Leaves fell from the tree'

'*ceṭṭu nuMḍi*' is considered the source of separation and the tag is `k5` is given as in the dependency tree shown in figure-(3.27): Observe the dependency trees in figure 3.27:

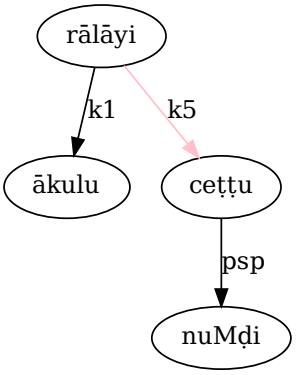


Figure 3.27: Dependency structure for the sentence- (3.51)

3.3.5.1 *prakruti apādāna* -'Source Material' (k5prk)

The relation of *prakruti apādana* can be considered a sub-tag of *apādāna* which refers to the source material. The tag k5prk is used to tag this relation. The default case marker, *nuMḍi* or the case marker *-tō* is used to represent this relation. The verb 'do' or 'make' are typical examples of verbs that can occur in this type of a sentence.

(3.52) a. *kobbari* *ākula* *nuMḍi* *cīpuru* *tayāru* *cēst-ā-ru*
coconut leaves from broom make do-PST-3.PL
'Brooms are made up of coconut leaves'

b. *kobbari* *ākula-tō* *cīpuru* *tayāru* *cēst-ā-ru*
coconut leaves-with broom make do-PST-3.PL
'Brooms are made up of coconut leaves'

3.3.6 *adhikaraṇa*(k7) - 'Locus'

The *kāraka* relation *adhikaraṇa* indicates the locus of the *kartā/karma* (Kulkarni and Sharma, 2019). It denotes the space or location of the *kartā/karma*. There are three kinds of *adhikaraṇa* observed in Telugu, namely, *kālādhikaraṇa*(k7t), *deśādhikaraṇa*(k7p), *viṣayadhikaraṇa*(k7) which are described further in this section.

3.3.6.1 *kālādhikaraṇa*(k7t) - 'Location in time'

The relation *kālādhikaraṇa* which translates to 'location in time' is a dependency relation which refers to the noun phrase indicating time information. The tag k7t is used to mark this relation. The default case suffix for *kālādhikaraṇa* is *-ki/ku* or *-lō*. There can be more than one *time* expression in the same sentence, which means there may be more than one k7t in a dependency tree. Instances of k7t are illustrated (3.53).

(3.53) *prakāś* *ninna* *sāyaMtraM* *5* *gaMṭala-ku*
prakash.NOM yesterday evening 5'o clock
nā-ku kanipiMc-ā-ḍu
I-DAT see-PST-3.SG.M
'I saw prakash at 5'o clock yesterday evening'.

In (3.53), it can be observed that only the time expression *5 gaMṭala-ku* '5'o clock' is marked with *-ki* suffix but the other two expressions *ninna* 'yesterday' and *sāyaMtraM* 'evening' are zero-marked but marked as k7t as they indicate time information.

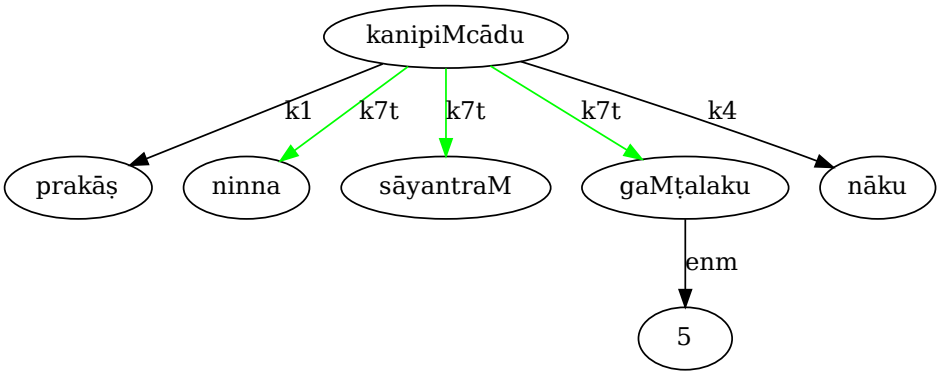


Figure 3.28: Dependency structure for the sentence- 3.53

(3.54) *vēsavi* *kālaM-lō* *māmiḍikāyalu* *meMḍugā* *doruku-tā-yi*
summer time-LOC mangoes many find-HAB-3.PL.N
'Mangoes are abundant in summer season'

In 3.54, the suffix *lō* suffix indicates time. It usually occurs with specific time expressions like 'evening time', 'vacation time' etc.

3.3.6.2 *deśādhikaraṇa*(k7p) - 'Location in space'

The *kāraka* relation *deśādhikaraṇa* indicates 'location in space' and is marked with the tag k7p. This relation is used to mark a noun phrase indicating a concrete location. The case-markers *-lō*, *mīda*, *-ki* in Telugu with the noun[+place] may indicate k7p. Consider (3.55)

(3.55) *kalpana* *svitjarlāMḍu-lo* *caduvu-kuM-ṭuM-di*
kalpana Switzerland-LOC study-REF-PROG-3.SG.F
'Kalpana is studying in Switzerland'

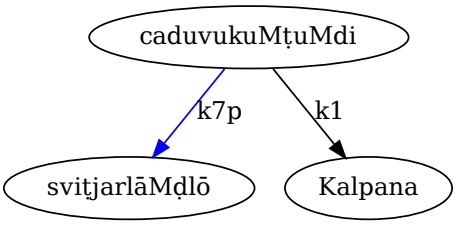


Figure 3.29: Dependency structure for the sentence- 3.55

In sentence-(3.55), *svitjarlāMḍu-lo* 'Switzerland' is case-marked with *-lō* and identified with the tag `k7p`.

(3.56) *ramēṣ* *kurci* *mīda* *kūrcunnāḍu*
ramesh.NOM chair on sit-PST-3.SG.M
'Ramesh sat on the chair'

Another case suffix -*"na"* is also used to determine location. This marker has a restrictive use, it is usually used with [-animate] nouns as observed by Arden(1873) (Ramarao, 2017, pp-43). Examples for the same are illustrated below:

(3.57) *nadi* *oḍḍu-**na*** *paḍavalu* *āgāyi*
river bank-LOC boats halt-PST-3.PL.N
'Boats halted at the river bank'

(3.58) *annī* *panulu* *nā* *netti-**na*** *paḍḍāyi*
all works I-POSS head-LOC fall-PST-3.PL.N
'All the work was assigned to me'

3.3.6.3 *viṣayādhikaraṇa*(`k7`) - 'Location elsewhere'

This tag is used when there is no concrete location that one can define. This tag roughly translates to 'location elsewhere' which means the location which is other than concrete time and space. Such relations are marked with `k7`. In Telugu, the default case suffix for *viṣayādhikaraṇa* is *lō*.

(3.59) *mukhya maMtri* *navarātri* *saMbarāllō*
chief minister navaratri celebrations-LOC
pālgonn-ā-ru
participate-PST-3.SG.HON
'The chief minister participated in the *navaratri*celebrations'

In (3.59) *saMbarāllō* 'in the celebrations' is not a concrete location, hence is marked with `K7`. See dependency tree-(3.59)

3.3.7 *sādriśya* (`k*u`) - 'Similarity & Comparison'

The *kāraka* relation of *sādriśya* is used for both comparison and similarity. The tag `k*u` is used to tag this relation. This tag is given to the noun phrase that is compared to another *kāraka* relation in the sentence. The `*` is used to replace any *kāraka* that the noun phrase is compared with. Any noun phrase can be compared with any other *kāraka* relation in a sentence. Hence, relations like `k1u, k2u, k3u, k4u` and so on are possible.

The post-positions used for similarity are *lāgā, lāMṭi, lā, vaMṭi*. Also, the post-positions used for comparison is *kaMṭe/kanna*. Consider (3.60)

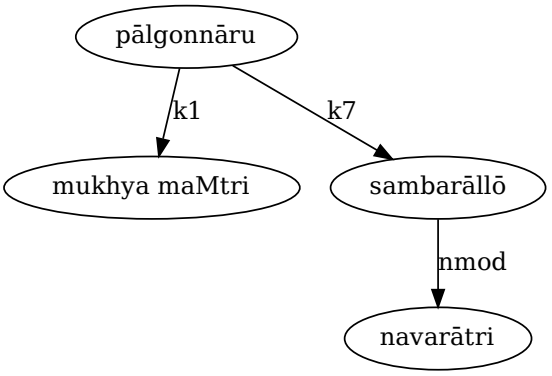


Figure 3.30: Dependency structure for the sentence- (3.59)

- **`k1u` - Similarity**

(3.60) *nēnu* *aMdari-laMṭi* *āḍapillan-e*
I.NOM all-like girl-EMPH
'I am like other girls'

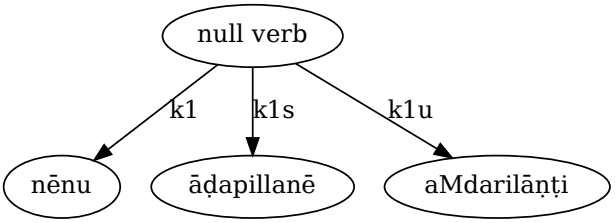


Figure 3.31: Dependency structure for the sentence- (3.60)

- **`k1u` - comparison**

(3.61) *nēnu* *atani* *kaMṭe* *pedda-vāḍi-ni*
I.NOM he.OBL than elder-MAS-1.SG
'I am elder to him'

- **`k2u` - Similarity**

(3.62) *nēnu* *lalita-ni* *nā* *kūturi* *lāgaā*
I.NOM lalita-ACC I-POSS daughter like

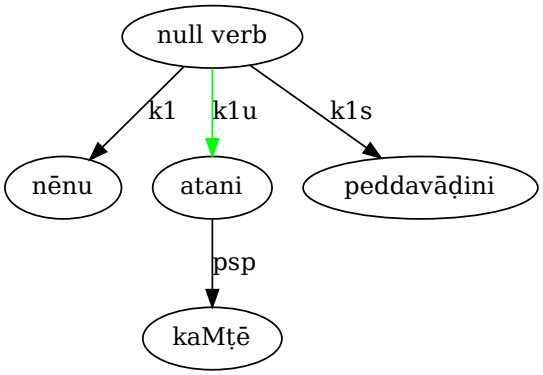


Figure 3.32: Dependency structure for the sentence- 3.62

cūsukuMṭānu
see-REF-PST-1.SG
'I take care of lalitha like my own daughter'

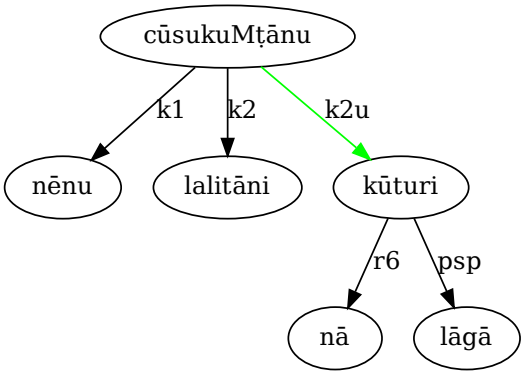


Figure 3.33: Dependency structure for the sentence- 3.62

- **k2u - Comparison**

(3.63) *nēnu* *kūragāyala* *kaMṭe* *paMḍla-nu* *ekkuva*
I-NOM vegetables-OBL than fruits-ACC more
iśṭapaḍatānu
like- HAB-1.SG
'I prefer fruits over vegetables'

The above illustrated examples show the relations of 'k1u' and 'k2u'. Similarly, k4u, k3u and so on are possible.

3.4 Non-*kāraka* Relations

Non-*kāraka* relations are not part of the obligatory relations in a sentence. They are the indirect participants of the action of the verb. All such indirect participants, named as 'r' relations in Pāṇinian framework, are part of the non-*kāraka* relations. They are adjuncts in a sentence that add often add extra, more-detailed information to the sentence. The relations like *hētu* 'reason or cause', *tādrtya* 'purpose', etc come under non-*kāraka* relations. See the table-(3.4) for non-*kāraka* relations:

Sl.No.	Non-*kāraka relations*	Tag	English Equivalent
1.	*hētu*	`rh`	Reason or cause
2.	*tādarthya*	`rt`	Purpose
3.	*prati*	`rd`	Direction
4.	*upapada sahakārakatva*	`ras_k*`	Association
5.	*ṣaṣṭīsambaMdhaḥ*	`r6`	Genitive
6.	Relation between noun and verb	`r6v`	Relation between noun and verb
7.	Duratives	`rsp`	Duratives

Table 3.4: Non-*kāraka* relations

3.4.1 *hētu*(`rh`)-'Reason or Cause'

The *kāraka* relation expressing the semantic role of reason or cause is tagged using the *hētu* relation. The arc indicating noun phrase describing the cause or reason of activity is attached to the verb using the '`rh`' relation. The default post-position used to express reason or cause in Telugu is *valla/valana* as in (3.64).

- **(i) Reason**

(3.64) *varṣaM* *valla* *kāryakramaM* *vāyidā* *paḍ-iM-di*
rain because program postpone fall-PST-3.SG.N
'Program got postponed due to rain'

In (3.64), *varṣaM valla* 'due to the rain' is expressing the reason for the postponement of the program. Hence, *varṣaM* 'rain' is marked as `rh` to the verb.

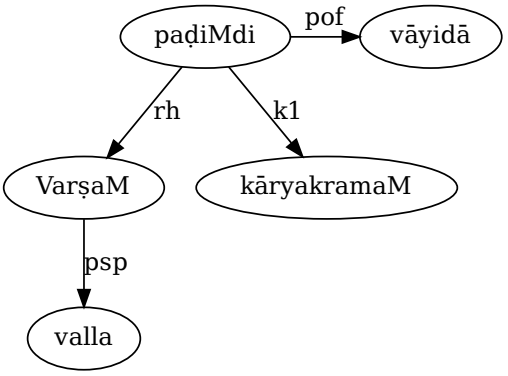


Figure 3.34: Dependency structure for the sentence-(3.64)

- **(ii) Cause**

(3.65) *īgala* *valana* *kalara* *sōku-tuM-di*
flies due-to cholera transmit-HAB-3.SG.N
'Cholera is transmitted by flies'

As illustrated in 3.65, *īgalu* 'flies' are the cause of cholera. Hence, it is given the tag `rh`.

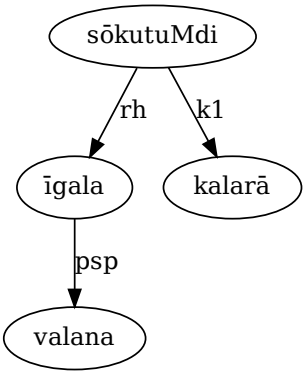


Figure 3.35: Dependency structure for the sentence-(3.65)

Apart from the default case marker, *valla/valana*, reason can be expressed using other case markers like *-ki/ku*, *nuMḍi*, *tō*.

(i). Consider example with the suffix *-ki/ku*

(3.66) ***gāliki*** *ākulu* *rālā-yi*
wind-DAT leavePL fell-PST-3.PL.N
'Leaves fell due to the wind'

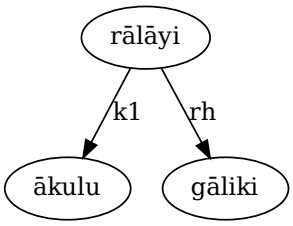


Figure 3.36: Dependency structure for the sentence-(3.66)

(ii). With the suffix *-tō*

(3.67) *kārti* *jvaraM-tō* *vanuku-tunnā-ḍu*
karti.NOM fever-ASS shiver-PROG-3.SG.M
'Karti is shivering due to fever'

(3.68) *mā* *amma* *rōgaM-tō* *maMcaM* *paṭṭ-iM-di*
we-POSS mother disease-INS bed hold-PST-3.SG.F
'My mother is bed-ridden with disease'

(iii). With the suffix *nuMḍi*

Ramarao (2017) shows an instance of 'nuMḍi' functioning as a 'reason marker'. Consider the example-(3.69):

(3.69) *nī* *nuMḍi* *nāku* *ī* *kaṣṭaM*
you-OBL from I-DAT this problem
'I had to face this difficulty because of you'

Another instance of causal interpretation is an infinitive verb with *baṭṭi* suffix in the subordinate clause expressing the cause and effect phenomenon of the main clause. Example-(3.70) exemplifies the cause and effect phenomenon:

(3.70) *manaM* *tondaragā* *cērukō-baṭṭi* *railu* *ekkāmu*
we.NOM.INCL fast-ADV reach- train board-PST.1.PL
'Because we reached fast, we boarded the train'

'Reason' can also be expressed using the verbal noun formative 'aḍaM/aṭaM' followed by *-tō/valla/valana* case suffix. See (3.71):

(3.71) *ārōgyaM* *ceḍaḍaM-tō/valla* *atanu* *udyōgaM* *mānēs-ā-ḍu*
Health spoil-GERU-INS he.NOM job stop-PST-3.SG.M
'He quit his job due to this ill-health'

In the above example, '*ceḍaḍaMtō*' functions as a reason to the action of 'quitting the job'.

3.4.2 *tādarthya* (rt) - 'Purpose'

The non-*kāraka* relation, *tādarthya*, which translates to 'purpose' is a dependency relation used to denote the purpose expressed through nouns intended by the activity of the verb. This, as other non-*kāraka* relations is also an adjunct relation. The postpositions *kōsaM/koraku/-ki* are the default markers used to express purpose in Telugu. *-ki* case suffix being highly ambiguous in Telugu is identified as *tādarthya* when a case-suffix *-ki* is interchangeable with *-kōsaM/koraku*. Such noun phrases expressing purpose is attached to the verb using 'rt' relation.

(3.72) *pavan* ***udyōgaM*** ***kōsaM*** *haidarābād*
pavan.NOM job for hyderabad
vacc-ā-ḍu
come-PST-3.SG.M
'Pavan came to hyderabad for a job'

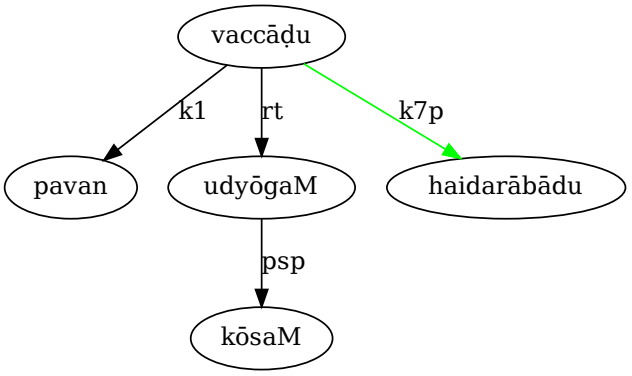


Figure 3.37: Dependency structure for the sentence- (3.72)

In (3.72), *udyōgaM kōsaM* is the purpose of the verb *vaccu*. Hence, it is tagged with the dependency relation *tadārtya*(rt).
This relation can also be expressed using the gerundival marker *aḍaM* followed by *kōsaM/koraku/ki*, like in (3.73):

(3.73) *bhāgya* *lā* ***cadavaḍaM*** ***kōsaM*** *kanaḍa*
bhagya law study for canada
veḷḷ-iM-di
go-PST-3.SG.F
'Bhagya left to Canada to study law'

In the above example-(3.73), *cadavaḍaM kōsaM*, is the purpose of the action '*veḷḷu*' 'go'.

(3.74) *rāju* *bhojanaM* ***cēy-aḍāni-ki/cēyaḍaM kōsaM***
raju food do-GER-DAT/for
iMṭi-ki *vacc-ā-ḍu*
home-DAT come-PST-3.SG.M
'Raju came home to have food'

'*cēyaḍāniki*' in the above example denotes the *tādarthya* or the purpose of the activity of 'coming home'. In such cases, *ki* is replaceable with *kōsaM* or *koraku*. Consider the tree below:

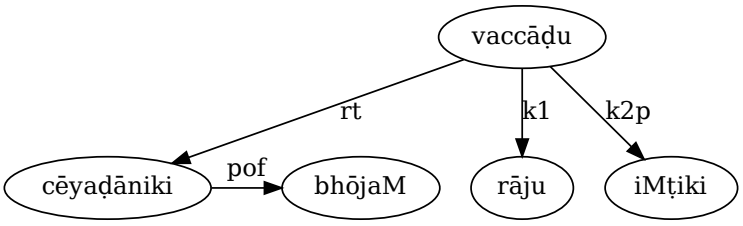


Figure 3.38: Dependency structure for the sentence-(3.74)

3.4.3 *prati* (rd) - 'Direction'

The *prati* relation is used to mark a dependency relation indicating *direction*. The tag `rd` is used to mark the relation *prati* 'direction' in this guidelines. The default postposition used for direction is *vaipu* in Telugu. Other postposition denoting side/direction include *pakka*. Other words indicating direction like up, down, above, below are also be part of this relation. This relation is also a non-*kāraka* relation and is hierarchically dominated by the verb.

(3.75) *nēha* *iMṭi* *vaipu* *parugulu*
neha.NOM house-OBL towards run-PL
tīsiMdi
take-PST-3.SG.F
'Neha ran towards the house'

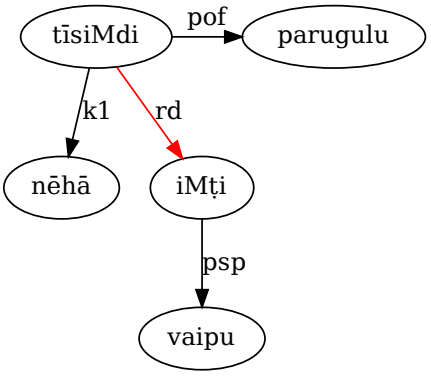


Figure 3.39: Dependency structure for the sentence- (3.75)

In (3.75), *iMti vaipu* is marked with '`rd`' to the verb as shown in the figure-3.42

3.4.4 *upapada sahakārakatva* (`ras-k*`) - 'Associative'

The relation of association is expressed through *upapada sahakārakatva* relation with the tag `ras-k*`. The '*' in the tag indicates *kāraka* it is associated with. The default case marker used for this relation in Telugu is *-tō/tō pāṭu*. The other markers include *-tō pāṭu/veMṭa*. Associative marker is synonymous to instrumental marker in Telugu. Association can be with any *kāraka* in a sentence. So, when a noun phrase is associated with *kartā* the relation is ras-k1. Likewise, if it is with *karma*, it is `ras-k2` and so on. Consider (3.76):

(3.76) *pārvati* *vinay-tō* *peḷḷi-ki* *veḷḷ-iM-di*
parvati.NOM vinay-ASS wedding-DAT go-PST-3.SG.F
'Parvathi went to the wedding with Vinay'.

Here, in (3.76), *vinaytō* 'with vinay' is related to *pārvati* as ras-k1 as it is associated with the *karta*

(3.77) *rāmuḍi* *veMṭa* *sīta* *veḷḷ-iM-di*
Ram.OBL along sita go-PST-3.SG.F
'Sita went along with Ram'

(3.78) *nānna* *ravi-ni* *kriśna-tō* *pāṭu* *veḷḷu-ani*
father.NOM ravi-ACC krishna-ASS along go-QUO
ceppāḍu
say-PST-3.sg.M
'Father told ravi to go along with krishna'

However, in 3.78, *kriśnatō* is related to *ravini* 'ravi' as ras-k2.

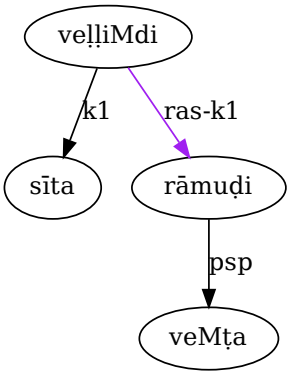


Figure 3.40: Dependency structure for the sentence- 3.76

(3.79) *amma* *cuṭṭāla-tō* *pāṭe* *pillala-ki*
mother.NOM relatives-ASS along-EMPH children-DAT
annaM peṭṭ-iM-di
food.ACC keep-PST-3.SG.F
'Mother served food to the kids along with the relatives'

In (3.79), *cuṭṭalatō* 'with relatives' is related to *pillaki* 'to the children' as ras-k4. And *pāṭe* is related to the noun before it as the 'psp'(used for postpositions).

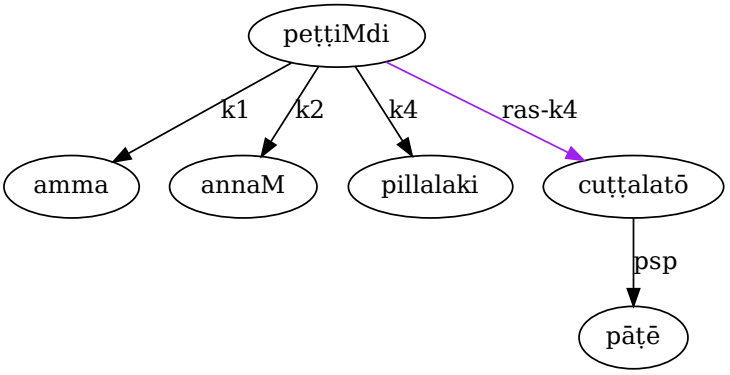


Figure 3.41: Dependency structure for the sentence- (3.79)

3.4.5 *ṣaṣṭīsambandhaḥ*(r6)- 'Genitive'

ṣaṣṭīsambandhaḥ or the genitive relation is the sixth case denoting a possessive relation between two noun phrases. In Telugu, genitive is morphologically realised

using the *yokka* postposition. However, this has become obsolete in modern Telugu and is omitted in most cases. Alternatively, noun in its oblique form when followed by another noun phrase expresses genitive case. In Telugu, some nouns do not distinguish between direct and oblique case morphologically. Genitive case can occur with such nouns as well. Genitive relation is tagged using the tag `r6`.
The tag `r6` is used to denote a typical kind of genitive relation like '*vāḍi yokka pustakaM*' 'his book' or relationship like '*rāmuḍi yokka tammuḍu*' 'Ram's brother' or the part-whole relation like '*ceṭṭu yokka panḍlu*' 'tree's fruits' fall under this category. Genitive relation can be identified using the overt postposition *yokka* or when a noun is preceded by a pronoun in its oblique form or when a noun is preceded by a proper noun.

In the absence of genitive postposition, distinction between a noun modifier and genitive relation can be made by inserting *yokka* between two noun phrases to observe if it results in a genitive relation. When the post-position *yokka* cannot be inserted between two noun phrases, it cannot fall under genitive relation.

Consider (3.80) example with both -*yokka*/-∅ :

(3.80)	*rīma*	*(yokka)*	*pustakaM*	*dorik-iM-di*
	reema.OBL	Genitive	book	find-PST-3.SG.N
	'Reema's book is found'			

In the example-3.80, the phrase *'rīma yokka pustkaM* is in genitive relation. *yokka* is tagged to *rīma* as psp (the relation 'psp' is discussed in further sections) and *rīma* as `r6` to the following noun. In 3.80, *pustakaM* is related to *rīma* as *ṣaṣṭīsambandhaḥ*.

(3.81)	*prakāś*	*prajala*	*maniśi*
	Prakash.OBL	people.NOM	man
	'Prakash is people' man'		

As explicated in example-(3.81), *yokka* is absent. However, *prajala* 'people' is in oblique case and is related to *maniśi* with *ṣaṣṭīsambandhaḥ*

(3.82)	*bālasubramaṇyaM*	*pāṭalaku*	*janālu*	*maMtramugdulu*
	balasubramanyam	songs-DAT	people	enchant-PL
	ayyāru			
	become-3.PL			
	'People got enchanted by Balasubramanyam's songs'			

In (3.82), the noun phrase *bālasubramaṇyaM*, a proper noun, does not change its form in oblique form. However, it is followed by a noun phrase *pāṭalaku* and *yokka* can be inserted between them. Hence, the relation between bālasubramaṇyaM & pāṭalaku is *ṣaṣṭīsambandhaḥ* and tagged as `r6` to the following noun.

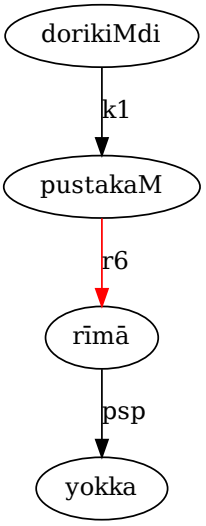


Figure 3.42: Dependency structure for the sentence- (3.80)

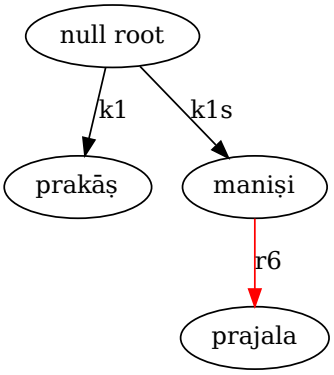


Figure 3.43: Dependency structure for the sentence- (3.81)

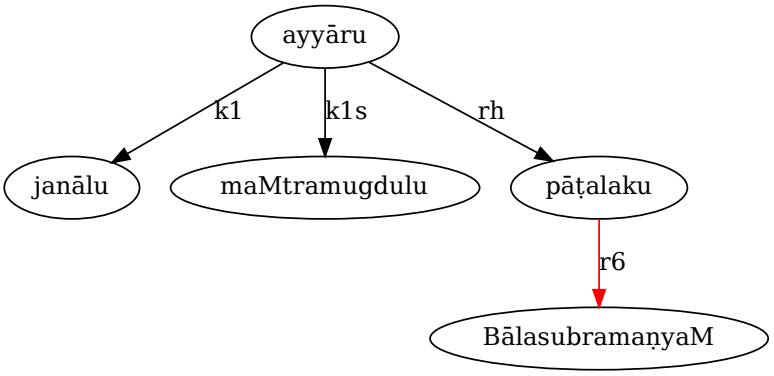


Figure 3.44: Dependency structure for the sentence-(3.82)

3.4.6 Duratives (`rsp`)

Duratives span over a period of time/distance and have a starting and an end point. The start and end points are usually indicated using two different postpositions. In Telugu, postpositions '*nuMḍi.....dākā , varaku, -ku*' are used to indicate duration of a time/distance from a start point to an end point. To the mark the start point `psp` 'nuMḍi' and the end point is marked with various markers such as *dākā*, varaku & *-ku* etc.

This is a relation that is used to connect two elements in the duratives. It does not connect a modifier and a modified relation but it connects two post-positions denoting duration. Consider (3.83):

(3.83) *nidhi* *haidarābād* *nuMḍi* *tirupati* *varaku/dākā*
Nidhi.NOM hyderabad from Tirupati to/till
railu-lō prayānaM cēstuMdi
railway-LOC travel do-PROG-3.SG.F
'Nidhi is travelling from Hyderabad to Tirupati by rail'

The duration of travel from 'Hyderabad to Tirupati' is expressed through the postpositions '*nuMḍi .....varaku/dākā/-ku*'. Hence, the arc connecting these postpositions is marked using the relation '`rsp`'. Consider the dependency tree below:

3.5 Other Dependency Relations

In this section, other dependency relation apart from *kāraka* and non-*kāraka* relations are discussed. Relations between modifiers of nouns & verbs are discussed

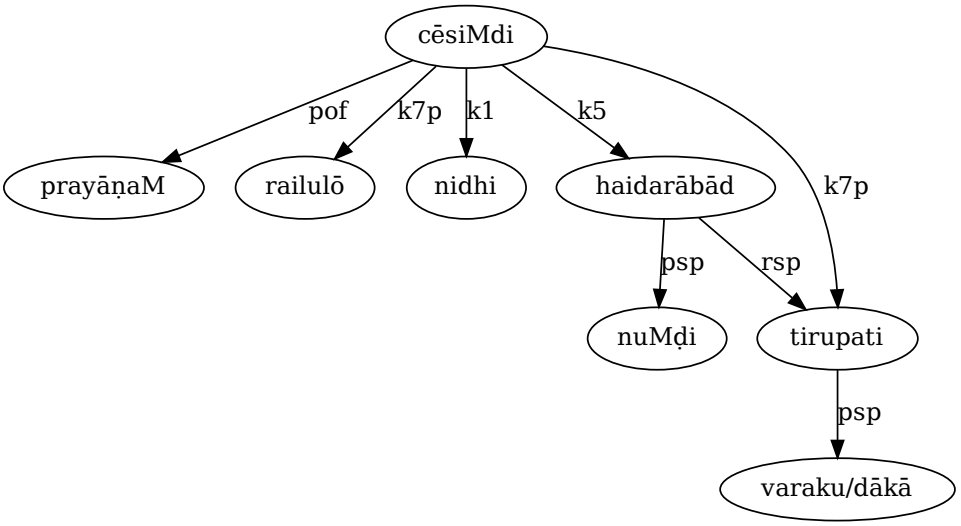


Figure 3.45: Dependency structures for the sentence- (3.83)

here. Table-(3.4) lists the tags that are discussed as part of this section.

3.5.1 Noun Modifier(`nmod`)

The relation '*nmod*' indicates a noun modifier and is used for nouns or any other particles/elements modifying a noun. This tag is used typically for nouns, proper nouns, demonstratives, adverbial nouns and quantifiers that modify nouns. There can be any number of noun modifiers in a sentence. There is no restriction on the number of noun modifiers. Here, we discuss various word categories that function as noun modifiers:

3.5.1.1 Noun as `nmod`

When nominals modify other nominals in a sentence, nmod tag can be used. Examples include the following:

(3.84) *gōḍa* *gadiyāraM*
hand watch
'Wall clock'

(3.85) *iMṭi* *pani*
house- OBL work
'House work'

(3.86) *prapaMca* *stāyi* *sadhupāyālu*
world level facilities

Sl.No.	Other Dependency Relations	Tag
1.	**Noun Modifier**	nmod
2.	Question Words	nmod_wq
3.	Quantifiers	nmod_quant
4.	Relative clause Modifying a noun	nmod_relc
5.	Adjective Modifying Nouns	nmod_adj
6.	Post-positions	psp
7.	**verb Modifier**	vmod
8.	serial Action	vmod:cp_serial
9.	Simultaneous Action	vmod:cp_simul
10.	Cause	vmod:cp_cause
11.	Conditional	vmod:cond
12.	Conditional: Serial Action	vmod:cond_serial
13.	Conditional: Cause	vmod:cond_cause
14.	Concessive Clause	vmod:conc
15.	Infinitive Clause	vinf:k1
16.	with Temporal Particles	vmod:temp
17.	**Manner Adverbs**	adv
18.	**Sentential Adverbs**	sent_adv

Table 3.5: List of other dependency relations

'world class facilities'

In the above examples, the nouns *gōḍa*, *iMṭi* and *prapaMca stāyi* modify the other nouns following them. Hence, they are given the tag nmod

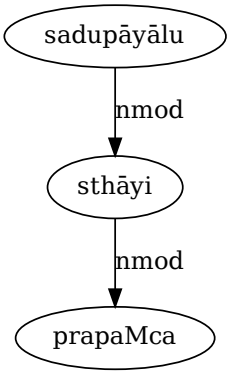


Figure 3.46: Dependency structure for the sentence- 3.86

3.5.1.2 Demonstratives as `nmod`

Demonstratives can also modify nouns when they occur before them. In Telugu, *ā*, *ī* are frequently used before nouns to indicate deixis. Consider (3.87):

(3.87) *ā* *rōḍḍu*
that road
'That road'

(3.88) *ī* *parvatālu*
those mountains
'Those Mountains'

As seen in ex-3.88, even demonstratives when preceded by nouns is considered as part of noun modifiers. Demonstratives can also occur with adverbs as in 'ī vidhaMga' 'like this'.

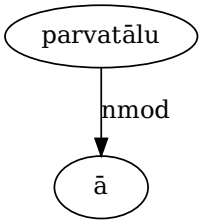


Figure 3.47: Dependency structure for the sentence- 3.88

3.5.1.3 Adverbial Nouns as `nmod`

Adverbial nouns in Telugu indicate time and space. They occur as nouns but indicate adverbial information. When such nouns precede a noun/pronoun, they function as `nmod`. Words like *appuḍu*'then', *ippuḍu* 'now' in their oblique form followed by a noun function as modifiers. Consider (3.89):

(3.89) *appaṭi* *sinima*
then cinema
'Cinema of those times'

(3.90) *akkaḍi* *paMṭalu*
there crops
'The crops there'

3.5.1.4 `nmod_wq` - 'Question words'

The tag `nmod_wq` is used for question words modifying nouns. This is a language specific tag specially used for Indian languages (Nallani et al., 2020a) .

(3.91) *sīta* *kōsaM* ***evar-ō*** *ammāyi* *vacc-iM-di*
sita.NOM for who-DUB girl come-PST-3.SG.F
'Some girl came looking for sita'

(3.92) ***ē*** *taragati*
which class
'which class'

(3.93) *mēku* ***enni*** *bhāśalu* *vaccu?*
you-SG.HON how many languages come
'How many languages do you know?'

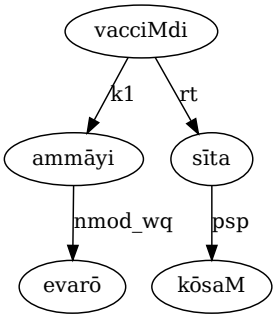


Figure 3.48: Dependency structure for the sentence- 3.91

3.5.1.5 Quantifiers as Noun modifiers `nmod_quant`

In Telugu, quantifiers also function as modifiers of nouns. Quantifiers precede the nouns as well as follow them. When quantifiers follow nouns, typically the final vowel is lengthened (see 3.95). In whichever case, they typically function as modifiers of nouns. Consider the table-3.6 for quantifiers that can function as nouns as well as noun modifiers.

(3.94) *koMdaru* *ammāyilu*
some girls
'Some girls'

(3.95) *ammāyilu* *aMdarū*
girls everyone
'All girls'

Quantifiers	English Equivalent
Human	
aMdaru	'everyone'
koMdaru	'some people'
iMdaru	'these many'
Non-human singular	
aMta	'that much'
iMta	'this much'
koMta	'some (indicating quantity) '
Non-human plural	
anni	'that number'
inni	'this number'
konni	'some (number)'

Table 3.6: Quantifiers list

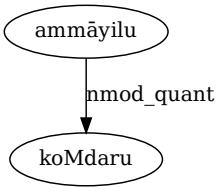


Figure 3.49: Dependency structure for the sentence- 3.94

Other quantifiers indicating numeric quantity like acre, half, 3/4th etc are also marked using this relation.

(3.96) *ekarā* *bhūmi*
acre land
'An acre land'

3.5.1.6 `nmod_relc` - Relative Clause

`nmod_relc` is a special tag used for relative clauses. A simple sentence in Telugu can be changed into a relative clause by replacing its finite verb by a relative participle (or verbal adjective) in the corresponding tense-mode and shifting the noun that it qualifies as head of the construction (Krishnamurti and Gwynn, 1985). Relative participle clauses occur immediately before nouns which they qualify. In Telugu, they show the distinction in tense in affirmative construction whereas in negative

they do not show the tense.

Relative participles are tagged as `nmod:relcl` in RBP. `nmod:relcl` is added with the argument relation of the noun which is relativized. In the sentence 3.97, the relativized nouns holds the object (k2) relation with the relative participle whereas the sentence 3.98 with the subject (k1) relation. There are tagged as `nmod:relcl_k2` and `nmod:relcl_k1` respectively in Figures (3.50) and (3.51).

(3.97) *nēnu* *cūs-ina* *maniṣi* *iMṭi-ki* *vacc-ā-ḍu*
I.NOM see-RP.PST man home-DAT came-PST-3.SG.M
'The man whom I saw came home'

(3.98) *nan-nu* *cūsina* *maniṣi* *iMṭi-ki* *vacc-āḍu*
I-ACC see-RP.PST man home-DAT come-PST-3.SG.M
'The man who saw me came home'

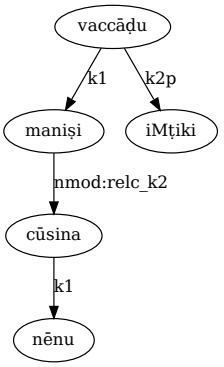


Figure 3.50: Dependency tree for 3.97

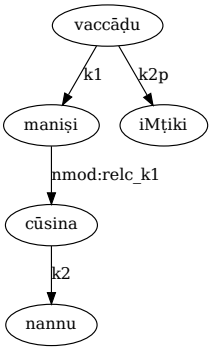


Figure 3.51: Dependency tree for 3.98

Relative participle clause constructions are ambiguous when the noun in the

relative clause has the potential to be an agent followed by the relative participle form of the verb which is transitive.

(3.99) *nēnu* *tin-ina* *kaMcaM* *pāta-di*
I.NOM eat-RP.PST plate old-3.SG.N
'The plate in which I ate is old'/'The plate which I ate is old'

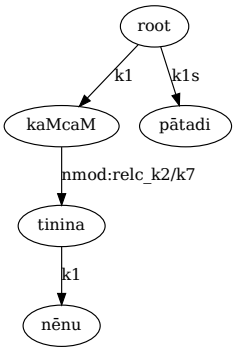


Figure 3.52: Dependency tree for 3.99

The token *kaMcaM* 'plate' can be interpreted with the tag `nmod:relc_k2` as well as `k7` (location) as in figure 3.52. However, selectional restriction rules will be operated and one tag is preserved. Here, in this case, *kaMcaM* 'plate' with the tag `nmod:relc_k2` is semantically not possible as it is not the object in relative clause.

3.5.1.7 Adjectives as nmod_adj

Adjectives are a syntactic class of words that modify nouns. *nmod_adj* relation is for adjectives modifying nouns. In Telugu, the adjectives when they occur before nouns, certain sufffixes as -∅ (as in **??**∅ in 3.100), *-ti* suffix as shown in **??**, *-ni* as explicated in 3.102. And when adjectives derive from nouns (as in 3.103), the suffix *-aina* is added.

Adjectives in Telugu precede nouns when occurred pronominally. However, there are also predicative adjectives which usually take GNP suffixes and take a different relation (k1s/k2s).

(3.100) ***pāta*** *baMḍi*
old vehicle
'old vehicle'

(3.101) ***nalla-ṭi*** *purugu*
black-ish insect
'Black insect'

(3.102)***pacca-ni*** *cēnu*
green-ish field
'Green fields'

(3.103)***aMdam-aina*** *ammāyi*
beautiful girl
'Beautiful girl'

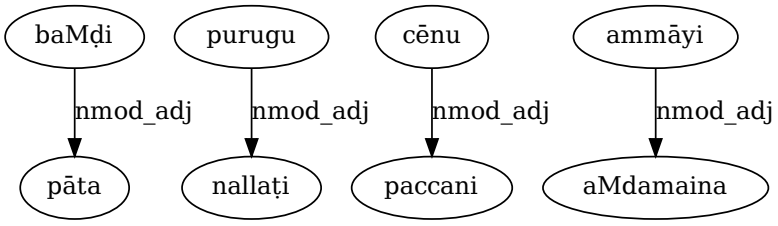


Figure 3.53: Dependency structures for the sentence- 3.100-3.103

3.5.2 Verb modifier (`vmod`)

Another most commonly occurring dependency relation is between a verb and a verb. The relations that exist between a verb and a verb are mainly between subordinate clause verb and the verb of matrix clause in Telugu. Anncora (Bharati et al., 2012) guidelines use the tag `vmod` for all relations between a verb and a verb.

Subordinate clauses (non-finite/participial constructions) in Telugu which function as modifiers are realized by verbs in their conjunctive, conditional, concessive and infinitive forms. It is observed that vmod is an underspecified tag and covers a wide range of relations. Hence, `vmod` is further divided into several subtags including clausal information. Consider the table-(3.7) for all the language-specific `vmod` tags included as part of this study:

3.5.2.1 Conjunctive Participle - Serial Action (`vmod:cp_serial`)

The conjunctive participle clause occurs as a subordinate clause and modifies the matrix clause. This conjunctive participle clause can be used to express verbal modifier (`vmod`) functions such as serial action, manner, cause and simultaneous action in Telugu. Example (3.104) explicates conjunctive participle as a serial verb. The figure-(3.54) is shown with the tag `vmod:cp_serial` for the sentence (3.104) with conjunctive participle expressing serial action.

Subordinate Clause Type	Enhanced Tag for Telugu
conjunctive participle	`vmod`
serial action	`vmod:cp_serial`
simultaneous action	`vmod:cp_simul`
Manner	`vmod:cp_manner`
Cause	`vmod:cp_cause`
conditional clauses	
Condition	`vmod:cond`
Serial action	`vmod:cond_serial`
Cause	`vmod:cond_cause`
Concessive clause	`vmod:conc`
Infinitive clause	`vinf:*`
with temporal particles	`vmod:temp`

Table 3.7: Dependency Tags for Subordinate Clauses in Telugu

(3.104) *rāmuḍu annaM* ***tin-i*** *paḍukunn-ā-ḍu*
Ram.NOM food.ACC eat-CP.PST sleep-PST-3.SG.M
'Ram ate food and slept'

3.5.2.2 Conjunctive Participle - Simultaneous Action (`vmod:cp_simul`)

The conjunctive participles express simultaneous action when the participle is durative as in the sentence-(3.104).

(3.105) *prakāsh.∅ sinimā* ***cūs-tū*** *cūldriMk tāg-ā-ḍu*
prakash.NOM cinema watch-CP.DUR cool-drink drink-PST-3.SG.M
'Prakash drank cool drink while watching a cinema'

3.5.2.3 Conjunctive Participle - Manner (`vmod:cp_manner`)

The conjunctive participle can express manner as explicated in the sentence-(3.106) with the figure-(3.58). Here, the type of verb class i.e. *motion class* is used as a cue to identify the manner in the verb modification with the tag `vmod:cp_manner`.

(3.106) *vimala.∅ āphīsu-ku nadic-i veḷt-uM-di*
vimala.NOM office-DAT walk-CP.PST go-HAB-3.SG.F
'Vimala goes to office by walk'

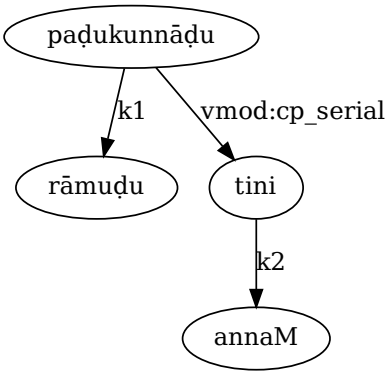


Figure 3.54: Dependency structure for the sentence- 3.104

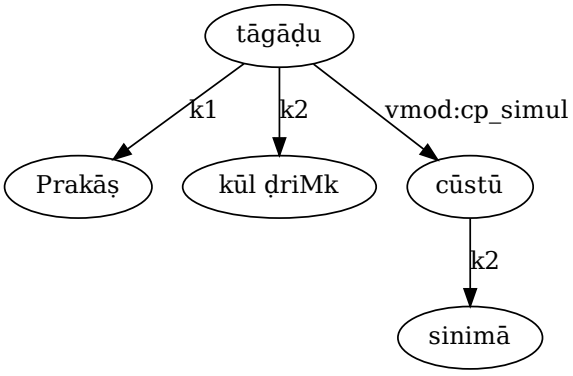


Figure 3.55: Dependency structure for the sentence- 3.105

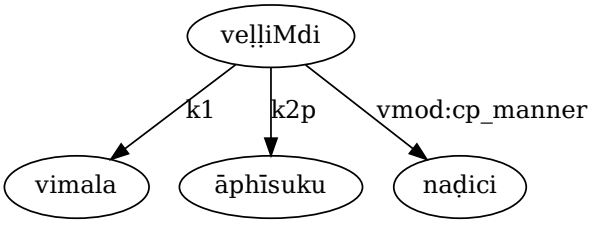


Figure 3.56: Dependency structure for the sentence- 3.106

3.5.2.4 Conjunctive Participle - Cause (`vmod:cp_cause`)

There is an instance of conjunctive participle expressing a causal sense. But it is difficult to capture causal information using syntax. Hence, the verb semantics may be used. However, a tag `vmod:cp_cause` can be used for sentences like-(3.107).

(3.107) *pāmu* ***karic-i*** *pillā-ḍu* *canipōy-ā-ḍu*
snake bite.CP.PST child-SG.M die-PST-3.SG.M
'The child died due to snake bite'

3.5.2.5 Conditional Clauses - Condition (`vmod:cond`)

Conditional clauses in Telugu not only express conditional sense but also show other interpretations leading to several parsing analyses. Such constructions are identified and tagged with language-specific tags in RBP.

In sentence (3.108), if the finite verb of a complex sentence is in non-past tense, it is considered as a conditional clause and is tagged with `vmod:cond`.

(3.108) *rāyi-tō* ***koḍi-tē*** *kāya* *kiMda* *paḍu-tuM-di*
stone-INST hit-COND fruit-NOM down fall-NON.PST-3.N.SG
'If you hit with a stone, the fruit falls'

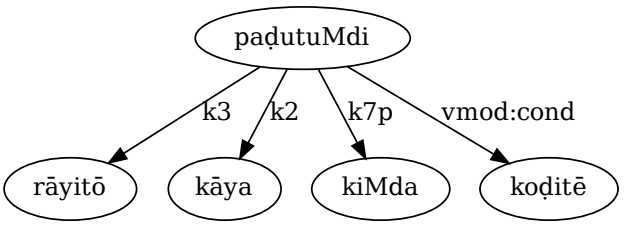


Figure 3.57: Dependency structure for the sentence- 3.108

3.5.2.6 Conditional Clauses - Serial Action (`vmod:cond_serial`)

Conditional clauses can also express serial action in Telugu. In such cases, the finite verb is realised with past tense unlike the case in vmod:cond(3.108). When the matrix verb is in the past tense, the conditional verb expresses the serial action and is given the tag `vmod:cond_serial` as the sentence 3.109.

(3.109) *rāyi-tō* ***koḍi-tē*** *kāya* *kiMda* *paḍ-iM-di*
stone-INST hit-COND fruit-NOM down fall-PST-3.N.SG
'The fruit fell when hit with a stone'

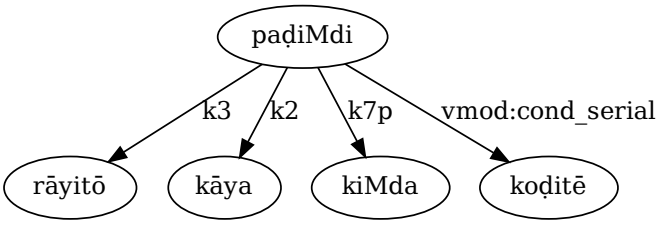


Figure 3.58: Dependency structure for the sentence- 3.108

3.5.2.7 Concessive Clauses(`vmod:conc`)

Concessive clauses in Telugu are formed by adding the suffix -*inā* to the verb stem and express the meaning 'even if/even though'. It functions as adverbial modifiers to the matrix verb. The negative concessive form is formed by the suffix '*akapoyinā*'. In such cases, this clause is tagged as `vmod:conc` in RBP.

(3.110) *nēnu* *cadiv-inā* *pāsu* *avva-lēdu*
I-NOM study-CONC pass become-NEG
'Even after studying, I did not pass (the examination)'

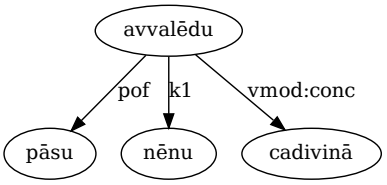


Figure 3.59: Dependency tree for 3.110

It is interesting that a temporal particle *appuḍu* + -*kī* suffix when added to adjectival participial form of the verb results in a concessive form. Consider the example (3.111):

(3.111) *āmē* *twaragā* ***veḷḷin-appaṭi-kī*** *railu* *dorak-alēdu*
she.NOM quickly go-TEMP-DAT train get-NEG
'She came early, even then, did not catch the train'

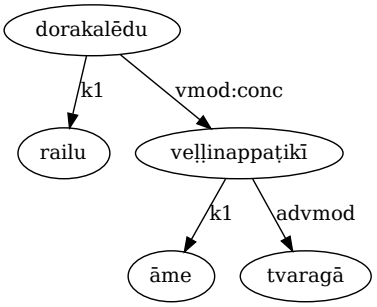


Figure 3.60: Dependency tree for 3.111

3.5.2.8 Infinitive Clauses (`vinf:k1`)

Infinitive clauses are not very common in Telugu. The infinitive suffix in Telugu is *-an* and the tag `vinf:k1` is used in tagging infinite clauses when they occur in the subject position as in the sentence-(3.112) and the respective dependency tree in figure- 3.61.

(3.112) *mīru nā-tō ā viṣayaM cepp-an akkar-lēdu*
You-HON I-INST that matter tell-INF need-NEG
'You need not tell me that matter'

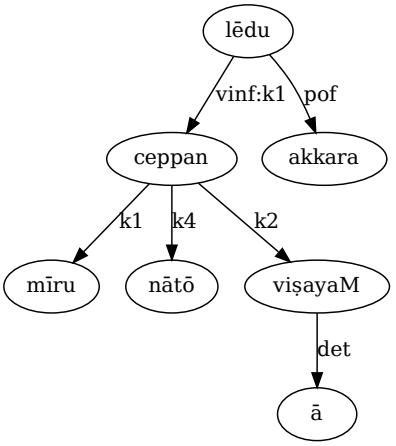


Figure 3.61: Dependency tree for 3.112

3.5.2.9 Verbal Modifier - Temporal (vmod:temp)

Relativized form of a verb when added to time particles indicates temporal information. For instance, the verb *tinu* in its relativized form *tine+appuḍu* results in *tineṭappudu*. `vmod:temp` is a dependency relation used for verbs denoting temporal information. Temporal particles like *appuḍu, lōpu, taruvāta, veMṭanē, dākā, muMdu, aMtavaraku, aMtadākā, aMtasēpu* etc., to the verb in its relativized form. The following verb paradigm of 'eat' *tinu* when added with temporal particles will result in the following forms:

Verb + Temporal particles	**English Equivalent**
nēnu tineṭappuḍu	'When I eat'
nēnu tinēlōpu	'By the time I eat'
nēnu tinnaveMṭane	'As soon as I eat'
nēnu tinnataruvāta	'After I eat'
nēnu tinēdākā	'Till I eat'
nēnu tinēaMta-varaku/dākā/sēpu	'up to the point I eat'
nēnu tinnamaṭuku	'To the extent I ate'
nēnu tinekoddi	'The more I eat'
nēnu tinesariki	'By the time I eat'
nēnu tineṭappaṭiki	'By the time I eat'
nēnu tinnappuḍaMtā	'Whenever I eat'

Table 3.8: Verb paradigm of temporal verb modifiers

(3.113) *nēnu* *caduvu-kunēṭ-appuḍu* *nannu* *pilav-a-kaMḍi*
I.NOM study-REF-TEMP I-ACC call-NEG-2.PL
'Do not call me when I am studying'

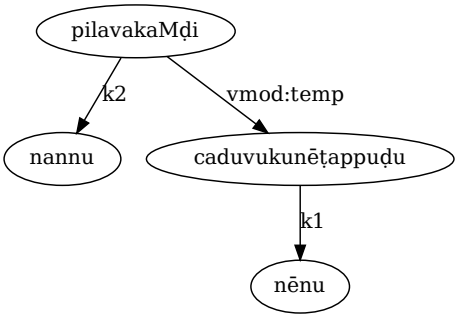


Figure 3.62: Dependency tree for 3.113

In addition to this, the infinitive form of the verb with *-gā* suffix also provides temporal information (See (3.114)). When this *gā* suffix is followed by an emphatic *ē*, it refers to 'as soon as...'. For instance, *vāḍu rāgānē nāku ceppaMḍi* 'Tell me as soon as he comes'.

(3.114) *mēmu* *goa* *veḷ-tūMḍa-gā* *pramādaM* *jarigiMdi*
I-NOM.EXC goa go-be- accident happen-PST.3.SG.N
'We met with an accident while going to Goa'

Hence, *veḷtūMḍagā* 'while going' in (3.114) is attached to the finite verb with `vmod:temp`.

3.5.3 Adverbs (`adv`)

Adverbs are an open syntactic class of words that occur as the modifiers of the verbs. Adverbs express three semantic functions namely time, place and manner. Time and place information in Telugu is expressed using nouns which are called as adverbial nouns of place and time (Krishnamurti and Gwynn, 1985). In the previous sections of `k7p` and `k7t`, it is mentioned that place or time information is part of those tags. Hence, adverbs here only mean manner adverbs.

Manner adverbs in Telugu are formed typically by adding a *-gā* suffix to adjectives. Consider example-3.115

(3.115) *vēṭūri* *adbhutangā* *pāta* *rās-ā-ru*
Veturi.NOM amazing-ADV song write-PST-3.SG.HON
'Veturi wrote the song amazingly'

In addition to this, *-gā* when added to predicative nominals also result in adverbs when followed by verbs like *uMḍu/kanabaḍu/naṭiMcu* etc. When the *-gā* suffix is added to the predicative nouns expressing physical or psychological states, such words also function as adverbs. Consider example-**??** below that show adverb with *-gā* suffix expressing psychological state:

(3.116) *nāku* *talanoppigā* *uM-di*
I.DAT headache-ADV be-3.SG.N
'I have headache'

Other instances of adverbs without *-gā* suffix include the following:

(3.117) *āme* *kaMṭi-niMḍā* *nidrapoi* *chālā* *rōjulu* *ayy-iM-di*
she.NOM eyes-full sleep-CONJ many days be-PST-3.SG.N
'Its been many days since he slept peacefully'

In the example-3.117, *kaMṭiniMḍā* 'lit. eye-fully' functions as an adverb. Other examples include *kaḷḷārā* 'lit. eye-fully', *kaḍupuniMḍā* 'stomachful', etc.

Other onomatopoeic words like *gaba gaba*, *jala jala*, *gala gala*, *tala tala* when followed by a verb function as adverbs.

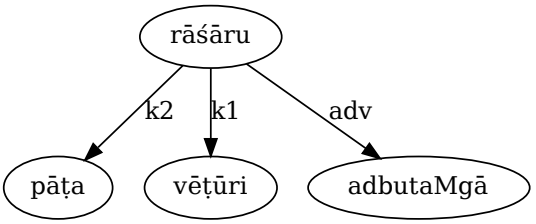


Figure 3.63: Dependency structure for the sentence- 3.115

Further, a noun phrase with *-tō* suffix can also indicate manner in Telugu (Ramarao, 2017). Consider (3.118)

(3.118) *āme* *bhakti-tō* *pūja* *cēs-iM-di*
she.NOM devotion-INS worship do-PST-3.SG.F
'She worshipped with devotion'

'bhaktitō' 'with devotion' in example-3.118 indicates the manner of the action done by the verb.

The *-lō* suffix also modifies the verb in certain constructions. Consider

(3.119) *amarēś* *kōpaM-lō* *arust-ā-ḍu*
amaresh.NOM angry-LOC shout-HAB-3.S.M
'Amaresh shouts when he is angry'

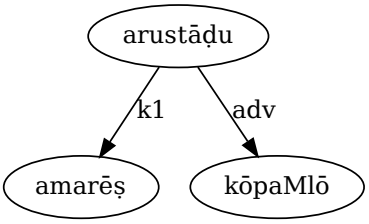


Figure 3.64: Dependency structure for the sentence- 3.119

(3.120) *nijāni-ki* *nēnu* *rān-u* *ani* *vāddiki* *telusu*
true-DAT I.NOM come-NEG-2.SG COMP he.DAT know
'Indeed, he knows that I will not come'

3.5.4 Sentential Adverbs (`sent-adv`)

Some adverbs function as modifiers of the whole sentence. They occur as connectors to the previous and the current sentence. All such adverbs that function as modifiers of the sentence are tagged using the tag `sent-adv`.

(3.121) *aMduvallana* *kumār* *peḷḷiki*
due-to-which kumar-DAT marriage-DAT
come-be-NEG-GO-PST-3.SG.M
rā-l-ēk-apōy-ā-ḍu
'Therefore, Kumar could not attend the wedding'

'*aMduvallana*' 'due to which' is considered as `sent-adv` because it modifies the whole sentence.

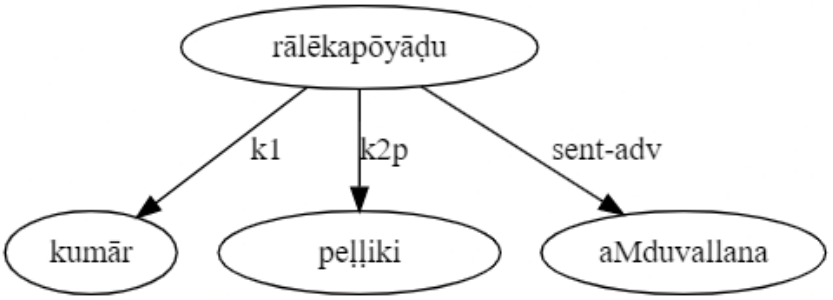


Figure 3.65: Dependency trees for 3.122

Some connectors like *idē kāka*, *mari*, *ayitē*, etc, when their scope is the whole sentence, are given the tag `sent-adv`.

3.6 Non-Dependency Relations

Relations between words in a sentence do not always comprise of dependency relations. Some relations can also be non-dependency relation where one words is not dominated by the other and are hierarchically at the same level. Relation between conjoints in a coordinate constructions, relation between light verb (NV compounds), reduplicated elements fall under this category.

Sl.No.	Non-Dependency Relations	Tag
1.	Conjunction and conjunct	`cc,conj`
2.	Part of Relation	`pof`
3.	Part of relation-Reduplication	`pof_redup`

Table 3.9: Non-Dependency Relations

3.6.1 Coordination(cc,conj)

coordination is a much debated concept among the linguistics and computational linguistics. Both theoretically and computationally, dealing with coordination has been a challenge. Many theoretical frameworks within dependency differ from each other in describing coordination as discussed in chapter-2. (Hudson, 1984; Mel'cuk et al., 1988; Tesnière, 1959).

Coordination can be defined as grouping of similar category words or sometimes clauses to provide a sense of collection. This can be formed between two or more noun, adjectives, verbs, adverbs etc. It should be noted that coordination is not a dependency relation. There is no mutual expectancy between words in a coordinate structure nor they are dependent on each other. Hence, coordination is not an asymmetric structure. There is no hierarchy between words.This relation is used to connect arcs between two words. However, providing a symmetric coordinate structure is quite a task. Symmetric structure can be achieved using two ways: 1) Considering the conjunction as a head and the conjuncts as the dependents like in (Bharati et al., 2012). 2) Placing the conjuncts at the same level and they are joined using a conjunction horizontally like in (Tesnière, 1959).

In Telugu, coordination is expressed through the conjunction 'mariyu' and an elongation of the last vowels of the conjuncts is often observed. But the use of conjunction is optional in Telugu. Sometimes, special characters like ',' is used but is still optional. The elongation of vowel after the conjuncts is often omitted in modern Telugu. In such cases, marking a coordinate relation gets complicated as this is a rule-based parser. A rule-based parser requires morpho-syntactic cues in-order to parse a structure. In cases where a conjunction, lengthening of vowels and special characters are absent, the system relies on 'agreement' on the verb to identify conjunction in Telugu.

For the conjunct, we use the tag 'conj' and for the conjunction we use 'cc' like in nivre-2016.

Consider the sentences below:

(3.122)	*nēnū*	*nā*	*tammuḍu*	*sinimā-ki*	*veḷḷ-ā-mu*
	I.NOM	I-POSS	brother	cinema-DAT	go-PST-1.PL

'I and my brother went to a movie'

In (3.66), it can be observed that the conjunct which is closer to the verb is marked as 'kartā' and the other conjunct is marked as 'conj' to the conjunct closer to the verb. This way, both the conjuncts are placed at the same level not dominating each other.

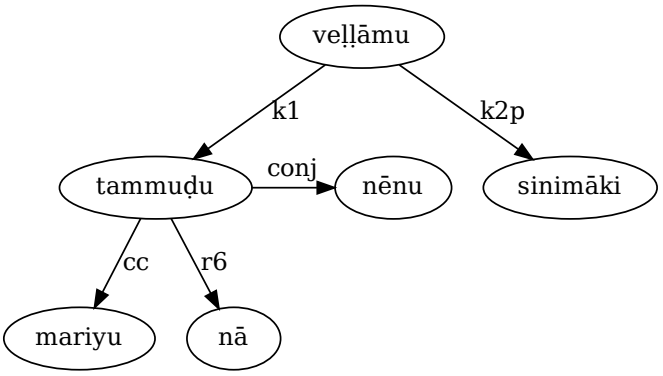


Figure 3.66: Dependency trees for 3.122

3.6.2 Part of Relation (pof)

POF expands to parts of relation/units of certain categories. When multiple fragments of a unit together functions as a single syntactic unit, we link those fragments using POF tag.

POF is not a dependency relation as mentioned in table **??**. These relations are used to mark two relations which do not have a head-modifier relation. This is especially used for Noun + Verb compounds in Telugu. Light verb/conjunct verb constructions are noun+verb compounds which function as a single unit in languages. Light verb constructions are very common in Indian languages. Noun carries the major meaning in such constructions and the verb acts as a verbalizer. Observe the data from Telugu :

(3.123) *sahāyaM* *cēyu*
help do
'to help'

(3.124) *pūrti* *cēyu*
complete do
'to complete'

(3.125) *peḷḷi* *cēsu-konu*
marriage do-REF-
'To marry'

(3.126) *peṭṭubaḍi* *peṭṭu*
investment take
'to invest'

(3.127) *lekka* *peṭṭu*
count take

'to count'

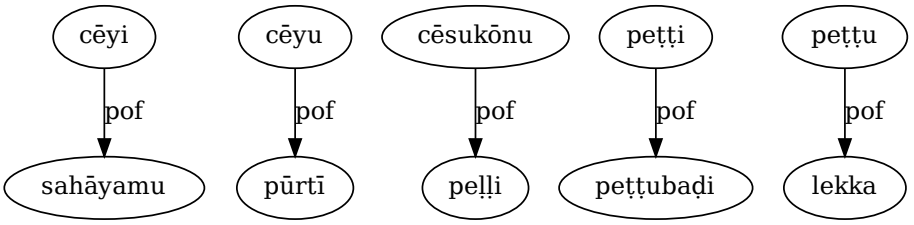


Figure 3.67: Dependency structures for the sentence- 3.123 -**??**

In examples [3.123-3.127], the verb 'cēyu/peṭṭu' do not actually have the meaning intented by the verb but their sense is bleached. The meaning of these verbs is drawn from the noun preceding it. Hence these fragments jointly renders the meaning of the verb. In (3.123), it is not about *doing* but it is about *helping*.
In all the cases like above, noun is linked to the verb using the tag *pof*.

3.6.2.1 Part of Relation - Reduplication (`pof_redup`)

This non-dependency relation is used to tag reduplicated compounds. Many verb-verb compounds are reduplicated to render a certain sense in languages. This is true of Telugu too. Reduplication can happen on nouns, adverbs verbs. Consider examples of reduplication of nouns below:

(3.128) *talupulu* ***kiṭa*** ***kiṭā*** *koṭṭukunnāyi*
doors ONOM ONOM bear-PST-3.PL
'Doors slammed with a bang'

(3.129) *caMṭipāpa* ***kila*** ***kilā*** *navv-iM-di*
infant ONOM ONOM laugh-PST.3.SG.F
'The child smiled'

(3.130) *nagalu* ***daga*** ***dagā*** *merisāyi*
nagalu ONOM ONOM shine-PST-3.PL.N
'The jewellery shined bright'

There can be both full and partial reduplicatives. The above illustrated examples are fully reduplicated. Consider (3.131) for partial reduplication:

(3.131) ***tiMḍi*** ***giMḍi***
food REDUP
'food'

(3.132) ***nagā naṭra***
'jewellery'

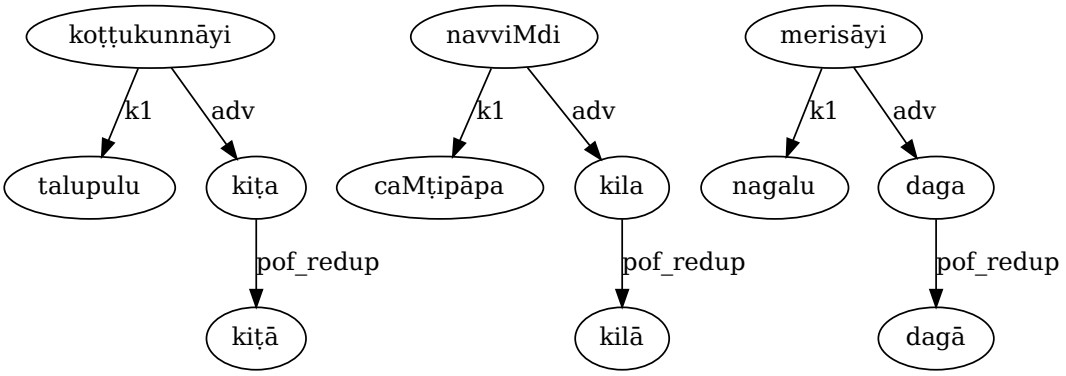


Figure 3.68: Dependency structures for the sentence- 3.128 -3.130

(3.133) *vāḍu* ***navvanē*** ***navv-a-ḍu***
He.NOM REDUP laugh-NEG-3.SG.M
'He never laughs'

For all such cases above, the tag pof_redup is to be used.Consider the dependency tree of the same.

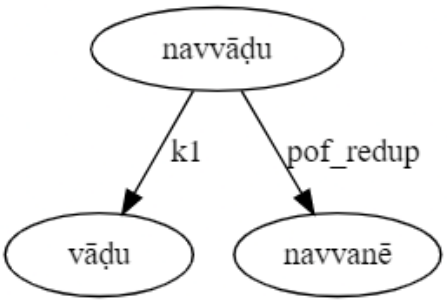


Figure 3.69: Dependency trees for 3.122

3.7 Miscellaneous Relations

Miscellaneous include all such relations that are functions either as connectors, particles or any other functional elements that are part of the sentence.

3.7.1 Subordination (`mark`)

This tag is used for 'complementizer' in complement structures. Complementizer in Telugu is 'ani'. Complementizer is obligatory in all structures except when the verb

Sl.No.	Miscellaneous	Tag
1.	Subordination	`mark`
2.	Intensifier	`intf`
3.	Negation	`neg`
4.	Particle	`rp`
5.	Interjection	`uh`
6.	Fragment of	`fragof`
7.	Address terms	`rad`
8.	Enumerator	`enm`
9.	Symbols	`rsym`

Table 3.10: Miscellaneous relations

'say' occurs in a sentence. Other elements that are included as part of this relation is *anṭē*. Consider (3.134) in which 'ani' is present:

(3.134) *rājeśwari* *vacc-iM-di* *ani* *vimala*
rajeshwari.NOM come-PST-3.SG.F QUO vimala.NOM
grahiMc-iM-di
realise-PST-3.SG.F
'Vimala realised that Rajeshwari came'

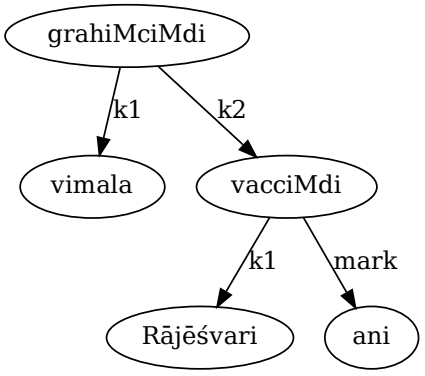


Figure 3.70: Dependency trees for 3.134

consider (3.135) in which 'ani' is absent.In this study, we do not supply any null element as a complementizer, this is out of the scope of this study.

(3.135) *abhi* *javābu* *ivva-ddu* *annā-ḍu*
abhi.NOM answer-ACC give-NEG say-PST-3.SG.M
'Abhi asked me to not to give an answer'

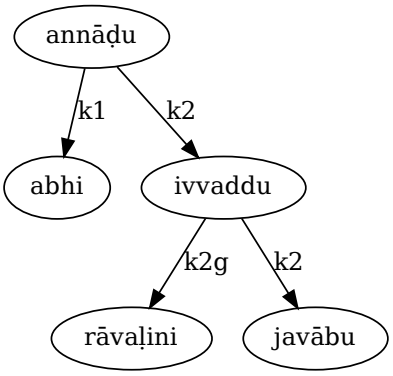


Figure 3.71: Dependency trees for 3.135

(i) 'ani' as a reason marker

(3.136) *mā* *nāyana* *ostāḍ-emō* *ani* *jadusu-kun-ēdi*
our father come-DUB QUOT scare-REF-HAB-3.SG.F
'She is scared that my father will come'

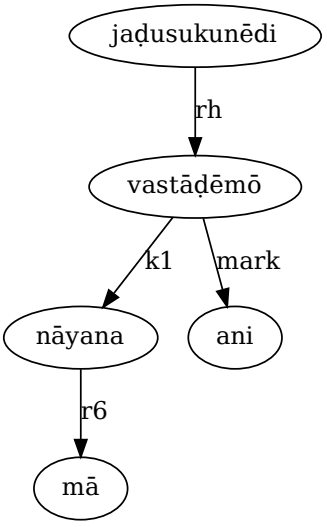


Figure 3.72: Dependency trees for 3.136

3.7.2 Intensifier(`intf`)

Intensifiers are elements that intensify/modify the adverbs/adjectives. Elements like *ati*, *marī*, *cālā* etc are used as intensifiers in Telugu. Intensifiers in Telugu occur before adjectives like in (3.137). And also occur before adverbs as in 3.138.

(3.137) *ati* *madhuramaina* *ṭi*
most delicious tea
'The most delicious tea'

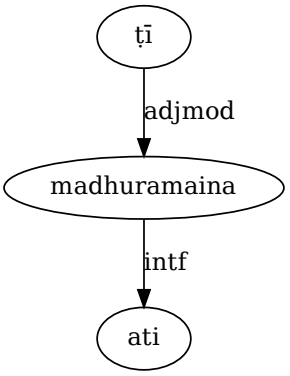


Figure 3.73: Dependency structure for the sentence- 3.137

(3.138) *cālā* *tīvraMgā* *śramiMcāḍu*
very intense-ADV work-PST-3.SG.M
'He worked very intensely'

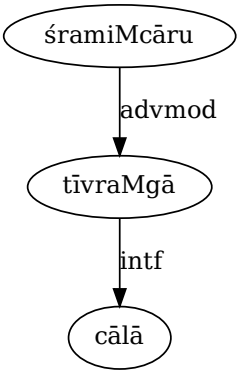


Figure 3.74: Dependency structure for the sentence- 3.138

3.7.3 Negation (neg)

This is used for negative words like *kāka, lēka, lēkuMḍa, lēdu* etc, that occur with both nouns and verbs.

(3.139) *hēma* *kākuMḍā*
hema without/excluding
'Excluding hema'

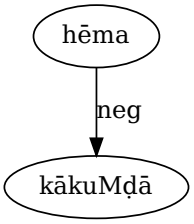


Figure 3.75: Dependency structures for the sentence- 3.139

3.7.4 Particles (rp)

rp is used for particles of all kinds to be attached to its respective head. Classifiers, Honorific words, etc are to be marked using this relation. Words like *gāru, kūḍā, etc* are to be tagged with this.

(3.140) *nēnu* *kūḍā* *sinimā-ki* *vas-tā-nu*
I.NOM also cinema-DAT come-PST-1.SG
'I will also come with you'

In 3.140, *kūḍā* 'also' is linked to *nēnu* 'I' as rp.

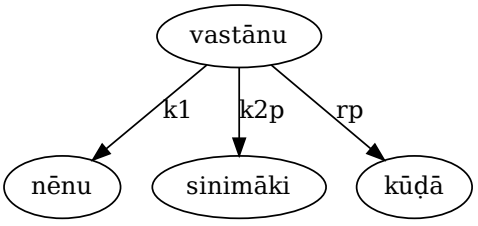


Figure 3.76: Dependency structures for the sentence- 3.140

3.7.5 Interjection(uh)

This relation is used for interjection expressions like *ayyō!, pāpaM!, cha cha!* in Telugu.

(3.141) *ayyō!* *ippuḍu* *em* *cēyāli?*
Interjection! now what do-POSS
ayyo! what to do now! In such cases, interjections are attached to the immediate word using uh relation.

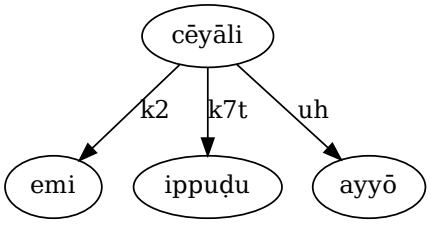


Figure 3.77: Dependency structures for the sentence- 3.141

3.7.6 Fragment of (fragof)

'fragof' is another non-dependency relation which is used to mark a fragments of a phrase which are either separated/bracketed abbreviations/particles of phrases that are separated etc. All such fragments which are related to other phrases are part of the 'fragof' relation.

(3.142) *telugu* *deśaM* *pārti* *[te.de.pa]* *ennikalalō*
telugu Desam party [TDP] elections-LOC
ōḍipōiMdi
defeat-PST-3.SG.N
'Telugu Desam Party (TDP) was defeated in the last elections'

The bracketed abbreviation is related to the preceding full form before it. The abbreviation 'TDP' is related to the element before it using 'fragof'.

3.7.7 Address terms (rad)

This is another non-dependency relation which is used to tag address terms. Words like 'ayyā', 'amma', '$\bar{}$srimati', etc are annotated using this relation. 'rad' is connected to the root.

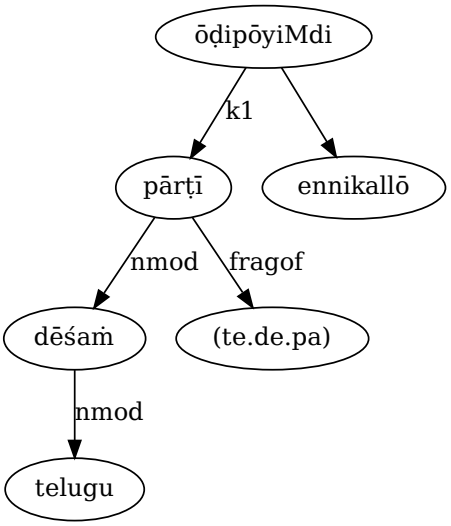


Figure 3.78: Dependency structures for the sentence- 3.142

3.7.8 Enumerator (`enm`)

'enm' is a tag used to annotate any numbers/number words found in a sentence. For example, see below:

(3.143) *vāḍi-ki* *16* *ēḷḷu* *vacc-ā-yi*
he.DAT 16 years come-PST-3.PL
'He is 16years old now'

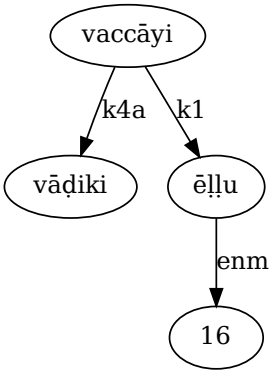


Figure 3.79: Dependency structures for the sentence- 3.143

3.7.9 Symbols (`rsym`)

This non-dependency relation is used to mark any special symbols that occured in a sentence like '.', '*', '/', '%', '$'etc. These symbols were directed towards the root with the 'rsym' relation.

(3.144) *puvvulu* , *paMḍlu* *guḍiki* *tīsukuni* *veḷḷ-āM*
flowers , fruits temple-DAT take go-PST-3.PL
'We took flowers, fruits to the temple'

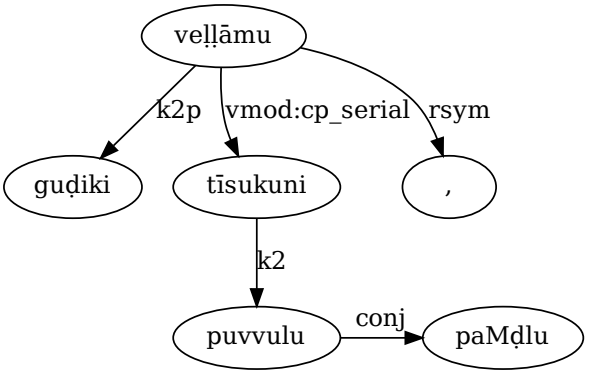


Figure 3.80: Dependency structures for the sentence- 3.144

Chapter 4

Architecture of the Rule-Based Dependency Parser

4.1 Introduction

In this chapter, we present a detailed architecture of the Rule-Based Parser (RBP) which is attempted to be utilized for Telugu. The parser exploits the Indian grammatical theories (discussed in chapter-2) in order to parse the sentence. Indian grammatical theories enabled the modelling of Indian languages over the years. The rule-based parser under study considers grammar as a set of constraints wherein parsing problem can be viewed as a constraint solving problem. The parsing problem in rule-based systems are solved by eliminating the structures that violate the constraints until we arrive at a single analysis. The currently available Indian language rule-based parser's algorithm for Sanskrit (Kulkarni, April 2021) is adopted for this study. Here, an attempt is made to adapt the algorithm of Sanskrit and enrich it with rules to parse Telugu sentences. Each component of the parser is explained in this chapter using illustrations wherever necessary.

4.2 Rule-based Parser for Telugu

Parsers, majorly, are modelled in two ways viz., grammar-driven or data-driven. Data-driven parsers use a collection of correctly parsed sentences called treebanks to train the parser using supervised machine learning algorithms. Popular approaches to data-driven dependency parsing include the graph-based and transition-based dependency approaches. On the other hand, grammar-driven or rule-based parsers use language generalizations to model the parser. Data-driven parsers are the state-of-the-art parsers. The number of treebanks for languages around the world has increased dramatically. But grammar-driven parsers have their own advantages. This study adopts the rule-based approach to parsing considering the advantages it provides for Telugu.

4.2.1 Why a Rule-based Parser?

To begin with, the advantages of rule-based parser are outlined which are also discussed in chapter-1(Pp-24) of this thesis as well:

1. One of the important reasons to opt for a rule-based parser is the agglutinative nature of Telugu which encodes prominent syntactic information in the form of suffixes as a morphological manifestation.

2. Rule-based parser allows wide-coverage of language structures. Data-driven parsers heavily rely on the corpus that is considered for annotation. If the corpus does not contain certain type of construction, the system developed henceforth also does not parse such constructions. This problem does not arise in grammar-driven approaches as the majority of sentence structures are covered in language grammars as rules.

3. Rule-based parsers does not require huge-corpus annotation. Corpus size starting from 5k is considered the minimal amount to build a reasonably good parser using data-driven approach. Building a corpus of 5k is a highly tedious task and requires skilled annotators. In rule-based parsers, a single rule can easily parse a specific relation in all the sentences with the inclusion of certain exceptions wherever required

4. Error identification and correction are possible in rule-based parsers. Accuracy of the system can be further improved using the inclusion of extra rules, lexical databases & linguistic cues

5. Analysis of the sentences will not be inconsistent in rule-based parser like in data-driven approach as the sentences are based on the grammatical information of the language. Whereas data-driven approach analysis purely depends on the annotators. Inter-annotator agreement is not a concern for rule-based parsers

6. In the context of ambiguous structures, all possible analysis can be shown in rule-based parsers whereas in data-driven parsers, one analysis for the given sentence is provided.

Considering all the above advantages of rule-based approaches, the current research attempts to build a rule-based parser for Telugu.
Th parser is built following the Indian theories of verbal cognition including three factors viz. *ākāMksā* (expectancy), *yōgyatā* (meaning compatibility), and *sannidhi*

(proximity) discussed in chapter-(2) of this thesis. The most common way of depicting syntactic analysis is through trees or graphs. We model the parser as a tree where the nodes of a tree correspond to words in the sentence and the edges between nodes correspond to the semantic relation between the corresponding words.

4.3 Architecture of the parser

The architecture of the parser is divided into 4 phases viz.,.

1. The Cleaning Phase
2. The Pre-processing Phase
3. The Parsing Phase
4. The Post-processing Phase

Parsing being a higher-level (syntactic level) analysis requires an adequate amount of pre-processing before the parsing module begins. For RBP, we require an input which is pre-processed till tokens with appropriate morphological analysis. Each step of the parser is presented in-detail like visualized in the fig-(4.1). This study uses CALTS lab tools for pre-processing the input sentences. Each step of pre-processing is illustrated using the following example-(4.1):

(4.1)

nā	*kadha*	*aMdari*	*āḍapillala*
I.POSS	story	everyone-POSS	girls-POSS
kadh-e			
story-EMP			

'My story is similar to the story of all other girls'

4.3.1 The Cleaning Phase

In the cleaning phase, the input sentence is made free from any unwanted junk information which is later normalized and converted to WX[1] notation.

4.3.1.1 Input Sentence Cleaning

Cleaning is the preliminary step of the corpus pre-processing. One of the major problems in creating an analyzable corpus is the unwanted pieces of information like logos, headers, HTML tags, other navigational symbols especially if one works on

[1]wx notation is provided as part of the transliteration schema

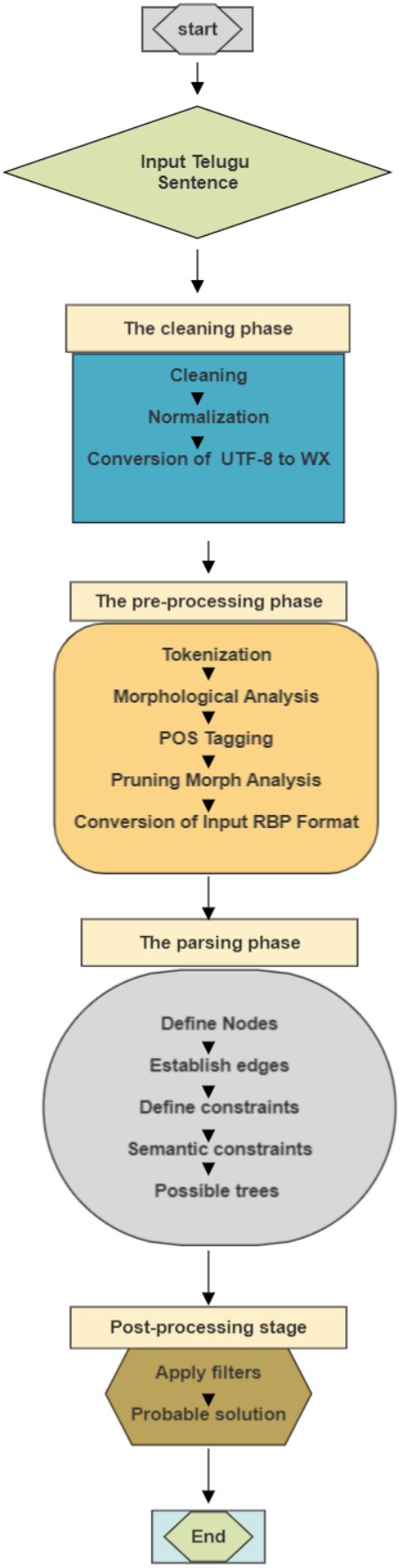


Figure 4.1: Architecture of the Rule-based Dependency Parser

web-based corpus. Cleaning refers to steps that one takes to standardise the text and to remove the text and characters that are not relevant for the study.

4.3.1.2 Normalization

Normalization involves spelling normalization and Unicode normalization. In spelling normalization, non-standard spellings are converted to standard ones. Spelling normalization is one of the components of normalization which corrects the wrong spelling. For example, the following words are normalized before parsing.
saduvu = *caduvu* 'read'
prakkana = *pakkana* 'beside'
In unicode normalization, the unicode character encoding is normalized. A single unicode value may be canonically equivalent to a sequence of code points. In texts, both these types of unicode values may occur. In such cases, the sequence of unicode values is normalized here to a single value (Uma Maheshwar Rao, 1999).

4.3.1.3 Conversion of UTF-8 to WX

Conversion involves converting the input text from Unicode (here, Telugu script) to WX format and vice versa. The WX transliteration schema can be seen in page-**??** of this thesis. The sentence-(4.1) is given in UTF-8 format which is converted into WX notation as given below:

UTF-8 Format

నా కథ అందరి ఆడపిల్లల కథే

WX format

```
nA kaWa aMxari Adapillala kaWe
```

4.3.2 The Pre-processing Phase

Pre-processing phase stands as a crucial phase for the current parser. This includes tokenization, Parts of Speech (POS) tagging, morphological analysis, pruning morph analysis and conversion of input to RBP format.

4.3.2.1 Tokenization

Tokenization is a process in which text is converted into tokens. Tokens refer to each individual occurrence of words, punctuations, symbols, numbers, acronyms etc. in the text. Tokenizer segments a text into words and sentences before any processing can be done. Texts are taken as input by tokenizer and output is given in Shakti Standard Format(SSF). 'Shakti Standard Format (SSF) is a highly readable representation for storing language analysis. It is designed to be used as a common format or common representation on which all modules of a system operate' (Bharati et al., 2007). The output of example-(4.1) is given below:

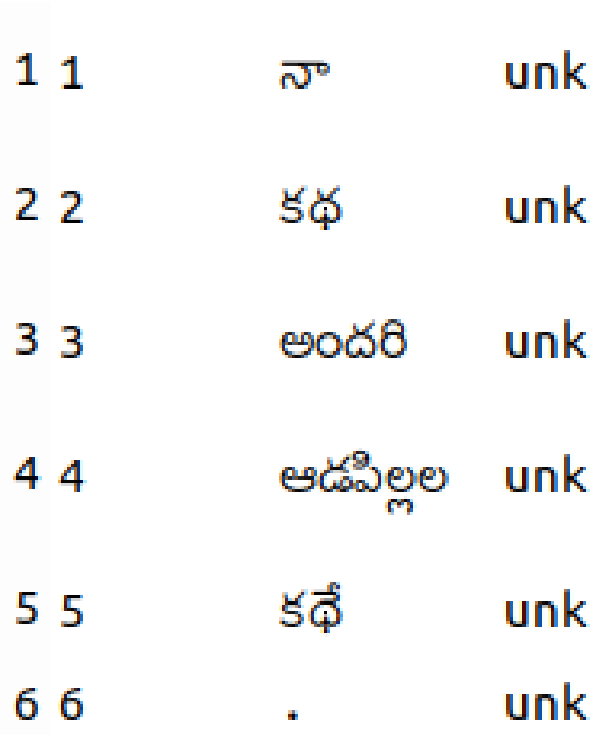


Figure 4.2: Tokenized output

4.3.2.2 Morphological Analysis

A morphological analyzer is a computational device to analyze a word into its root, category and morpho-syntactic information in terms of its constituent morphemes. This analyzer takes the word as the input and produces the root and its grammatical features as the output. The CALTS morphological analyzer uses word paradigm model (Hockett, 1954) to analyze the words. The sample output of example-(4.1) for the morphological analyzer is given below (Rao et. al., 2010):

```
1       nA      unk     <fs af='na,avy,,,,,A,A_avy'>|<fs af='na,avy,,,,,V,V_avy'>|<fs af='nenu,pn,any,sg,1,o,ti,ti'>|<fs
af='ni,avy,,,,,A,A_avy'>|<fs af='nu,avy,,,,,A,A_avy'>
2       kaWa    unk     <fs af='kaWa,n,,sg,,d,0,0'>|<fs af='kaXa,n,,sg,,o,ti,ti'>
3       aMxari  unk     <fs af='aMxaru,pn,,pl,,o,ti,ti'>
4       Adapillala      unk     <fs af='Adapilla,n,,pl,,o,ti,ti'>
5       kaWe    unk     <fs af='kaWa,n,,sg,,,0,0_e'>
6       .       unk     <fs af='&dot;,punc,,,,,,'>
```

Figure 4.3: Morphological Analysis

The morph analysis is given in the fourth column of SSF format. It consists of feature structure(fs) with attributed features(af) in eight fields of information as output. They consist of root, lexical category, gender, number, person, case(direct/oblique), case marker and Tense Aspect Mood(TAM) suffix.

4.3.2.3 POS Tagger

Parts of Speech tagger is a device which assigns unique parts of speech to a word. In this study, BIS POS tag [1] set based POS tagger is used which is developed in CALTS. These POS tagged output is given in the third column of SSF. Based on POS output, token with multiple catgeory is disambiguated by removing the unmatched POS category with morphological analysis category.

```
<Sentence id="1">
1       nA      PRP     <fs af='na,avy,,,,,A,A_avy'>|<fs af='na,avy,,,,,V,V_avy'>|<fs
af='ni,avy,,,,,A,A_avy'>|<fs af='nu,avy,,,,,A,A_avy'>
2       kaWa    NN      <fs af='kaWa,n,,sg,,d,0,0'>|<fs af='kaXa,n,,sg,,o,ti,ti'>
3       aMxari  PRP     <fs af='aMxaru,pn,,pl,,o,ti,ti'>
4       Adapillala      NN      <fs af='Adapilla,n,,pl,,o,ti,ti'>
5       kaWe    NN      <fs af='kaWa,n,,sg,,,0,0_e'>
6       .       SYM     <fs af='&dot;,punc,,,,,,'>
</Sentence>
<Sentence id="2">
</Sentence>
```

Figure 4.4: POS tagged output

4.3.2.4 Pruning Morph Analysis

In the case of a token with same category, the previous and the following tokens' morphological analysis is used and the unwanted morph analysis is pruned out. When the system comes across some such entities, system chooses one candidate analysis at random and predicts this to be the correct morphological analysis for the given word. The existing CALTS pruner is used for this study.

[1] http://tdil-dc.in/tdildcMain/articles/134692Draft%20POS%20Tag%20standard.pdf

Later, based on the context, at pruning stage, the system selects appropriate morphological analyzer's output where there is more than one analysis for a given word. Rules to prune out analyses are identified and provided as a database.

```
<Sentence id="1">
1.1     nA      PRP     <fs af='nenu,pn,any,sg,1,o,ti,ti'>
1.2     kaWa    NN      <fs af='kaWa,n,,sg,,d,0,0'>|<fs af='kaXa,n,,sg,,o,ti,ti'>
2.1     aMxari  PRP     <fs af='aMxaru,pn,,pl,,o,ti,ti'>
2.2     Adapillala      NN      <fs af='Adapilla,n,,pl,,o,ti,ti'>
2.3     kaWe    NN      <fs af='kaWa,n,,sg,,,0,0_e'>
2.4     .       SYM     <fs af='&dot;,punc,,,,,,'>
</Sentence>
```

Figure 4.5: Pruning morph output

4.3.2.5 Conversion of Input to RBP Format

The format provided from all the modules in the pipeline, mentioned earlier, align the data in SSF. SSF is converted to RBP i.e. input with specific format including important features viz., the actual word (word), word's root (rt), lexical category(lcat), gender(gen), number(num), person(per), case(case), *vibhakti*(viBakwi), suffix(suff) in brackets delimited by spaces. In this stage, POS information is removed. A token with morph analysed(morphana) information consists of a specific id(token number), word(actual token), rt(root), lcat(lexical category), gen(gender), num(number), per(person), case(direct/oblique case), *vibhakti*(case suffix), suff(any other suffixes).

However, the parser is not programmed based on this output. It requires the input to be converted to a Consider the converted output below:

```
1 (morphana (id 1) (word nA) (rt nenu) (lcat pn) (gen any) (num sg) (per 1) (case o) (viBakwi ti) (suf
2 (morphana (id 2) (word kaWa) (rt kaWa) (lcat n) (gen X) (num sg) (per 3) (case d) (viBakwi ∅) (suff
3 (morphana (id 3) (word aMxari) (rt aMxaru) (lcat pn) (gen X) (num pl) (per 3) (case o) (viBakwi ti)
4 (morphana (id 4) (word Adapillala) (rt Adapilla) (lcat n) (gen X) (num pl) (per 3) (case o) (viBakwi
5 (morphana (id 5) (word kaWe) (rt kaWa) (lcat n) (gen X) (num sg) (per 3) (case X) (viBakwi ∅) (suff
```

Figure 4.6: RBP Format

4.3.2.6 Null-verb Insertion

In verbless sentences, i.e. nominal or adjectival predicate sentences, a null verb is introduced as the parser required a verb to be a head of the sentences as discussed in chapter-2. At this stage, after the conversion of the input is done, the system checks each input for its availability of the verb. Hence, if the input lacks a verb like in the following example, the system provides a null verb in-order to make sure

the trees are headed by the verb. Consider the fig-4.7

```
1 (morphana (id 1) (word nA) (rt nenu) (lcat pn) (gen any) (num sg) (per 1) (case o) (viBakwi ti) (suff ti))
2 (morphana (id 2) (word kaWa) (rt kaWa) (lcat n) (gen X) (num sg) (per 3) (case d) (viBakwi ∅) (suff ∅))
3 (morphana (id 3) (word aMxari) (rt aMxaru) (lcat pn) (gen X) (num pl) (per 3) (case o) (viBakwi ti) (suff ti))
4 (morphana (id 4) (word Adapillala) (rt Adapilla) (lcat n) (gen X) (num pl) (per 3) (case o) (viBakwi ti) (suff t
5 (morphana (id 5) (word kaWe) (rt kaWa) (lcat n) (gen X) (num sg) (per 3) (case X) (viBakwi ∅) (suff ∅_e))
6 (morphana (id 6) (word dummy) (rt dummy) (lcat v) (gen X) (num X) (per 3) (case X) (viBakwi ∅) (suff ∅))
~
```

Figure 4.7: Null verb insertion

4.3.3 The parsing Phase

In this section, a detailed algorithm of the parser using illustrations is presented. Algorithm consists of multiple stages and each stage is described theoretically and computationally.

4.3.3.1 Algorithm of the parser

The basic algorithm for parsing is as follows: Kulkarni (2021a)

1. "**Define one node each** corresponding to every word in a sentence
2. **Establish directed edges** between the nodes, if there is either a mutual or unilateral expectancy (*ākāṅkṣā*) between the corresponding words. In order to hypothesize a possible edge between two words, we refer to the expectancies of the verbs and the corresponding *vibhaktis* and then postulate a possible relation
3. **Define constraints**, both local on each node as well as global on the graph as a whole. One of these constraints corresponds to *sannidhi* (proximity)
4. Use **semantic constraints** to filter out the meaning-wise non-congruent solutions
5. **Extract all possible trees** from this graph that satisfy both local and global constraints
6. Produce the **most probable solution** as the first solution by defining an appropriate cost function. The cost C associated with a solution tree is defined as $C = \sum_e d_e \times r_k$, where e is an edge from a word w_j to a word w_i with label k, $d_e = |j - i|$, r_k is the rank[1] of the role with label k.

[1]All the roles are ranked, on the basis of heuristics, from 1 to 99.

Then the problem of parsing, a sentence may be modeled as the task of finding a sub-graph T of G such that T is a Directed Tree (or a Directed Acyclic Graph)".

4.4 An elaboration of the algorithm

The steps mentioned in (4.3.3.1) is elaborated in this section.

4.4.1 Step-1 - Define Nodes

Step-1 of the algorithm is the preliminary step wherein each word of the sentence will be assigned a specific node as part of the graph. For instance, for a sentence *nāku ī ūru kotta* 'I am new to this village', 4 nodes corresponding to each word will be defined at this stage. Number of nodes in a graph will be equal to number of words in a sentence.

4.4.2 Step-2 - Directed Edges

Step-2 of the algorithm deals with the expectancies of verbs and the corresponding *vibhaktis* which enable the parser to postulate a possible relation. Once the verbs' expectancies are fulfilled, a directed edge will be formed between them. Expectancy, as discussed in chapter-2, can be between any modified-modifier pairs. Here, we discuss how expectancies are identified by RBP using rules and how directed edges are established between the words.
In order to establish edges, it is important to understand how expectancies between words are expressed in language grammars and how it can be modelled in NLP. We will look into this phenomenon from two perspectives viz., theoretical and computational.

4.4.2.1 Theoretical Perspective

Pāṇini, theoretically, discusses assigning case-markers to each relation. However, Computationally, parser uses a reverse process viz., assigning a relation using case-markers.
Pāṇinian treatment of *kāraka* relations consider a system of default *vibhakti* for each relation. This *vibhakti* assignment is independent of verb semantics. (Table-4.1) provides the default *vibhakti* for *kāraka* relations in Telugu. Apart from these default *vibhaktis*, there exists cases of deviation in Telugu. These deviations arise

when verbs do not follow linguistic generalizations or when a structure is out of scope of linguistic generalization. In order to handle these deviations, Pāṇini discusses a model wherein he proposes two methods to handle deviation (Preeti K, 2010):

1. Assigning a different *vibhakti*
2. Imposing a new *kāraka* relation.

Preeti K (2010) summarizes the ways of mapping semantic relations to *vibhaktis* through *kārakas* in Pāṇinian Grammar(PG). Classification of mapping of semantic relations to case suffixes can be observed in fig-(4.8). All the *kāraka* and non-*kāraka* relations fall under the categories mentioned in the mapping. Using this classification as a theoretical base, rules for RBP are framed, thereby, establishing directed arcs. However, establishing directed edges are not a straightforward task as case suffixes as small as 7 in number exhibit around 40 relations which need adequate information about the expectancies of verbs and certain features of the nouns. Based on the fig-(4.8), the semantic relations between noun-verb are divided into the following types and illustrations are provided:

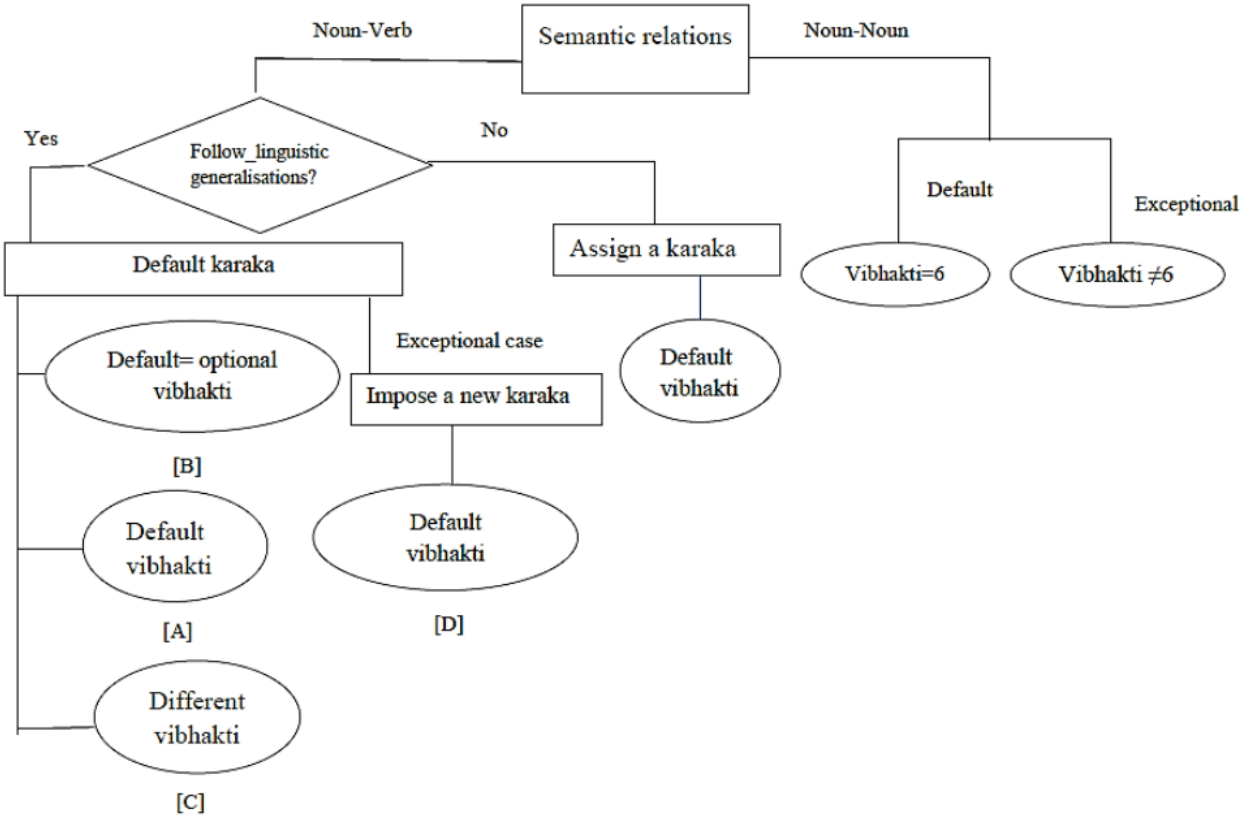


Figure 4.8: Semantic relations-*vibhakti* mapping

Type-A: Default Marking

The first type of semantic relation is when the language follows the linguistic generalisation and takes a default *kāraka* as listed in (Table-4.1). In example (4.2) as explicated, *kartā* (k1) and *karma* (k2) are marked with the default *vibhakti* i.e -∅ and *-ni* respectively.

(4.2) *nēnu-∅* *ravi-ni* *cūs-ā-nu*
I.NOM Ravi-ACC see-PST-1.SG
'I saw Ravi'

In this type of mapping, the system do not require any kind of semantic information or in case of non-availability of semantic cues, the parser considers in the default *vibhakti* and provides the relation. Consider the (Table 4.1) for default case-marking in Telugu. we provide a table as to how each type in the semantic mapping will be programmed in the Rule-based parser.

Sl.no	***kāraka* relation**	***vibhakti***
1	*kartā* 'Roughly subject'	∅
2	*karman* 'Roughly object'	*-ni/-nu*
3	*karaṇa* 'instrument'	*-tō*
4	*sampradhāna* Recipient/beneficiary'	*-ki/-ku*
5	*apādāna* 'source'	*nuMḍi/niMci/nuMci*
6	*adhikaraṇa* 'locus'	*-lō, -ki/ku*

Table 4.1: *kāraka* relations and default *vibhaktis* in Telugu.

Type-B: Optional Marking

In certain relations, there exist instances of verbs in addition to the default case marking which deviates from the default case marking and assign optionally other case-suffixes as in (Table 4.3) and (Example 4.4).

The verb *ceppu* 'to tell' assigns either *vibhakti -ki* or *-tō* to express the relation *sampradana* (k4) i.e the recipient of an action as in (Example 4.3).

(4.3) *nēnu-∅* *prakās-ki/-tō* *ā* *viSayaM* *cepp-ā-nu*
I.NOM Prakash-DAT/ASS that matter tell-PST-1.SG
'I told that matter to Prakash.'

Similarly, the verb *ekku* 'to climb' in Telugu, has an expectancy of a noun expressing the location 'to climb'. In this case, the noun is marked either with the *vibhakti -nu* or *mīda* as in (4.4).

(4.4) *nēnu-∅* *ēnugu-nu/mīda* *ekk-ā-nu*
I-NOM elephant-ACC/on climb-PST-1.SG

'I climbed the elephant.'

Type-C: Different Marking

In certain cases, it is found that a different *vibhakti* is assigned instead of the default one to indicate a particular semantic relation. For instance, the default *vibhakti* to indicate the source of separation, *apādāna* i.e. the ablative case as in (Example 4.5). However, in the case of mental separation as in (Example 4.6) where the *kartā*, *vāḍu* 'he' separates himself mentally due to the fear of *siṃhaM* 'lion' which is considered as *apādāna* in PG but it is realized by the different *vibhakti* i.e. *-ki*, not *-nuMḍi*.

(4.5) *ceṭṭu* *nuMḍi* *āku-lu-∅* *rāl-ā-yi*
tree from leave-PL-NOM fall-PST-3.PL
'Leaves fell from the tree.'

(4.6) *vāḍu-∅* *siṃhāni-ki* *Bayapaḍa-tā-ḍu*
He.NOM lion-DAT scare-HAB-3.SG.M
'He is scared of a lion.'

Type-D: Imposition of a new *kāraka*

In certain exceptional cases, it is found that a new *kāraka* is imposed using a default *vibhakti*. This can be due to the extension of the case relation as explicated in (Example 4.7) where *iḷḷu* 'home' is the *karma* to the verb *veḷḷu* as per PG, however it is marked with the *vibhakti '-ki'*.

(4.7) *nēnu-∅* *iMṭi-ki* *veḷḷ-ā-nu*
I.NOM home-DAT go-PST-1.SG
'I went home.'

From the above classification, it can be observed that single case-marker can serve different roles based on the context. Here, context refers to the properties of verbs and nouns. For example, a noun with *'-ki'* suffix when occurs with a verb of [+motion] property serves a different role in a sentence. Whereas a noun with *'-ki'* having the properties of [+time] serves a different role. This leads us to the main process of the step-2 which is the use of 'lexical semantics of nouns and verbs'.

4.4.2.2 Computational Perspective

In the theoretical perspective, we have observed that case-markers are assigned to each relation. Based on the above types, computationally, a reverse process is employed wherein each relation is assigned using case markers. However, this is not a straightforward process. The fact that each relation is not marked using a unique case-marker makes the process of building a parser tedious. This leads to ambiguity in marking relations. Hence, other linguistic cues like semantics of nouns and verbs

coupled with case-suffixes are considered to parse sentences. An elaborate on how relational disambiguation happens in the parser is discussed in the section-4.7.

4.5 Define Constraints

The **step-3** of the algorithm is to define local and global constraints. The local constraints used in the parser to postulate the best possible result using local information. Kulkarni (2021b). The properties of local constraints are given below:

1. A node can have one and only one incoming edge.
2. There cannot be more than one outgoing edges with the same label from the same node, if the relation corresponds to a *kāraka* relation.
3. There cannot be self loops in a graph.

In addition to the local constraints, we also use global constraints like *sannidhi* 'proximity' which is a constraint. The important property of it is to restrict the crossing of edges.

4.6 Use of Semantic Constraints

The use of semantic constraints is dealt in **step-4** of the algorithm. It is quite important to include semantic constraints in a parser to arrive at correct solution. For instance, the sentence *colorless green ideas sleep furiously* is a syntactically well-formed sentence but semantically ill-formed. The natural language feature which enables the use of semantically well-formed constructions is termed as *yōgyatā* in PG or the selectional restriction in western terminology. Selectional restriction is defined as the semantic constraint imposed on the arguments of verbs. We use selectional restriction of arguments of the verb to prune out the non-congruent solutions and arrive at a single parse. Let us consider the following examples-(4.8) & (4.9):

(4.8) *tūPānu-∅* *iḷḷu-∅* *kūlc-iM-di*
storm.NOM house-ACC destroy-PST-3.SG.N
'The storm destroyed houses.'

(4.9) *iḷḷu-∅* *tūPānu-∅* *kūlc-ā-yi*
house.NOM storm-ACC destroy-PST-3.SG.N
'Houses destroyed the storm.'

Both the examples (4.8) and (4.9) are syntactically well-formed sentences, when *yōgyatā* is applied, the example (4.9) stands semantically ill-formed because of the fact that 'Houses destroying the storm' is a semantically unacceptable sentence. In order to solve such issues, the canonical word order of a language is used as a cue.

The other instance in which we use selectional restriction is to disambiguate *kartā* and *karma* in Telugu. When the *karma* is [-animate], the *vibhakti* ∅ is used which is synonymous with the marker for *kartā*. In such cases, two ontological features [+/- animate] and [+/- human] could resolve the ambiguity in Telugu as well as in other Indian languages as examined by Bharati et al. (2008b). *Kartā* is considered to be higher in its hierarchical order in comparison with *karma*. Consider the following example:

(4.10) *nēnu.∅* *pāṭa.∅* *pāḍ-āli*
I-NOM song.ACC sing-HOR
'I should sing a song'

Here, the verb *pāḍu* 'sing' is in hortative mood and does not agree with the *kartā*. However, the verb *pāḍu* 'sing' expects *kartā* with a semantic feature of [+human] thus, (*nēnu*) 'I' is prioritized over a [-animate] entity (i.e. *pāṭa*) 'song'. These two semantic features proved to be quite helpful in resolving the most ambiguous relation of *kartā* and *karma*. As seen earlier, this parser exploits various linguistic information which stands crucial in disambiguating certain cases.

4.7 Use of Lexical Semantics of Nouns and Verbs

The lexical semantics of verbs provides cues in certain cases to disambiguate *vibhaktis* and their corresponding semantic relation. Relational disambiguation is a challenging task as a single case suffix can result in several relational interpretations without the interference of semantics. For instance, *-ki/-ku* is the highly ambiguous case marker in Telugu. It can serve many semantic roles in a sentence based on the features of the respective verbs and nouns. All the semantic information regarding various classes of verbs and nouns is stored as part of the database (see table-(4.9)) to disambiguate relations. Database is augmented with RBP rules in-order to produce the correctly parsed trees. Here, we discuss various

case markers and their respective ambiguous relations[1] along with the technique adopted by RBP to resolve them. Consider the figure-(4.9) of default *vibhaktis* and the number of case relations it can exhibit:

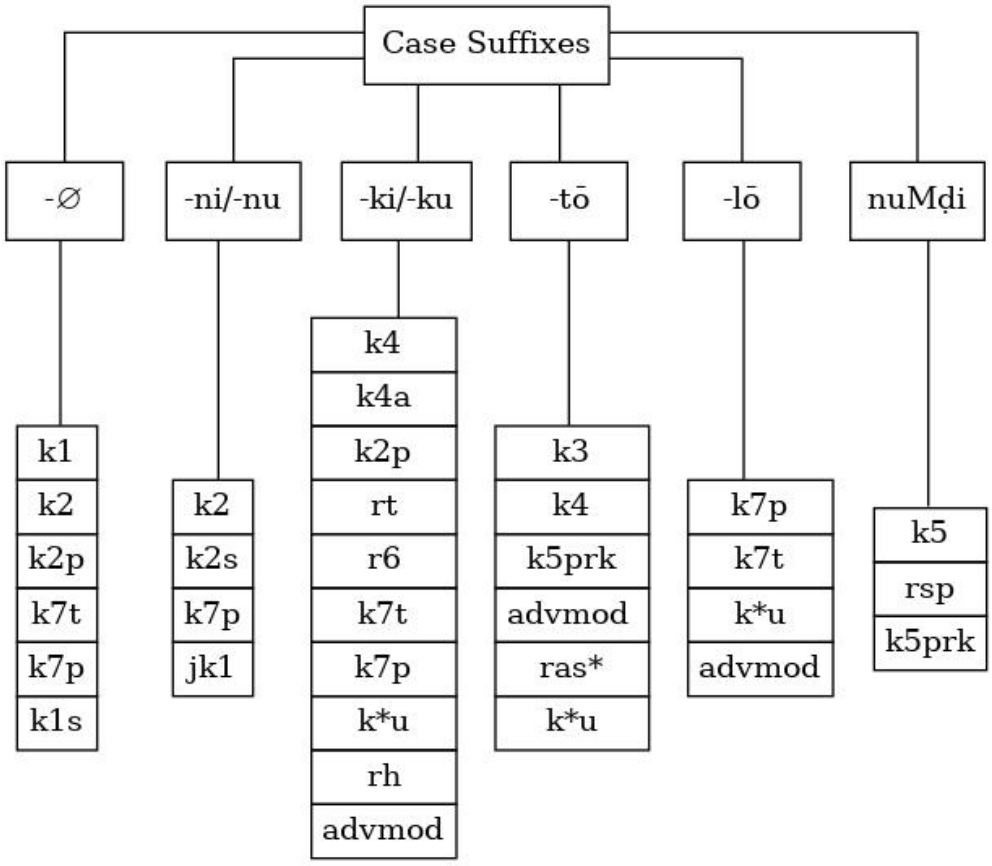


Figure 4.9: *vibhakti-kāraka* Mapping

4.7.1 Relational Disambiguation - [-∅] suffix

A nominal with null suffix, '-∅', can exhibit several case relations in Telugu like *kartā*, *karma*, *kartā samānadhikarana*, can indicate goal or destination, *kālādhikarana*, *deśādhikarana*. In RBP, ambiguous sentences are disambiguated using syntactic and database cues. Consider the table-(4.7) that include the syntactic cues, lexical semantics and case-suffixes used to mark the probable semantic relation. Reference examples are also provided [2].

To illustrate consider (4.11)

[1]It should be noted that providing all possible semantic relations or an exhaustive list of semantic relations for each case-marker can be a highly tedious task and does not fall under the scope of this study. However, most common ambiguous relations are mentioned in this section

[2]All the reference example numbers mentioned in tables-4.7, are part of chapter-3

Sl.No.	Case Suffix	Lexical Semantics	Linguistic Cues	Semantic Relation	Example Reference
1.	-∅	• Noun[-Time]	GNP agreement with finite verb	k1	(3.8)
2.	-∅	• Verb[+Transitive] • Verb[+Ditransitive] • Noun[-Animate] • Verb[-Passive]	attached to the nearest verb	k2	3.24
3.	-∅	• Verb[+Passive] • Noun[-Time] • Noun[-Place]	*-abad̄u* suffix on verb	k2	(3.31)
4.	-∅	• Verb[+Motion] • Noun[+Place]		k2p	(3.36)
5.	-∅	• Noun[+Time]		k7t	(3.53)
6.	-∅	• Verb[+Copula] • Noun[-Time]	Nearest nominal before finite verb	k1s	(3.21)

Table 4.2: Relational Ambiguity -∅ Suffix

(4.11) *paigā* *mā* *iMṭlō* *aMdarū.∅* *polaM-∅* *pōv-ā-li*
moreover our house-LOC everyone field go-HAB-HORT
'Moreover, everyone in our house must go to the field'

In the example-(4.11), words *aMdaru* 'everyone' and *polaM* 'field' are null-marked. However, they exhibit two different relations, namely, *kartā* and k2p ''goal/destination. In order to differentiate between this relations, RBP uses the two databases viz., (1) Motion verbs & (2) Place nouns. Hence, if a null-marked noun is part of the place nouns and is followed by a motion verb, it is tagged as k2p is not the *kartā*.
Likewise, when a nominal with -∅ case suffix occurs but is part of the [+time] nouns [eg. *sāyaMtraM* 'evening']. It will be given the tag k7t.

When a nominal is null marked and occurs with the copula verb or occurs in a verbless sentence. RBP marks both *kartā* and *kartā samānadhikarana* which will be later filtered out.

The same technique is followed for disambiguating all case-suffixes.

4.7.2 Relational Disambiguation - [*ni/nu*] suffix

The *-ni/nu* suffix is as ambiguous as the null marker in Telugu. It expresses mutliple case relations. Consider the table-(4.3)

Sl.No.	Case Suffix	Lexical Semantics	Syntactic Cue	Semantic Relation	Example Reference
1.	*-ni/nu*	• Verb[+Transitive] • Verb[+Ditransitive]		`k2`	(3.22)
2.	*-ni/nu*	• Verb[+Transitive] • Verb[+Create]	Immediately followed by k2	`k2s`	(3.39)
3.	*-ni/nu*	• Noun[+Place] • Verb[+Motion]		`k2p`	(3.37)
4.	*-ni/nu*	• Verb[+Causative]		`jk1`	(3.17)
5.	*-ni/nu*	• Noun[+Animate] • Verb[+communication]	Usually followed by a Noun[-animate]	`k2g`	(3.30)

Table 4.3: Relational Ambiguity *-ni/nu* Suffix

For disambiguating *-ni/nu* suffix, databases of transitive verbs, ditransitive verbs and causative verbs proved to be very useful. These lists should be improved whenever a new verb is encountered.

4.7.3 Relational Disambiguation - [*-ki/ku*] suffix

-ki/ku suffix in Telugu is the most ambiguous among all the case suffixes. Consider the following set of examples:

(4.12) *nēnu-∅ vāḍi-**ki** pustakaM-∅ icc-ā-nu*
I-NOM he-DAT book-ACC give-PST-1.SG
'I gave a book to him.'

(4.13) *nēnu-∅ baḍi-**ki** veḷḷ-ā-nu*
I-NOM school-DAT go-PST-1.SG
'I went to the school'

(4.14) *nā**ku** telugu telusu*
I-DAT telugu know-PST-3.SG.N
'I know Telugu'

Sl.No.	Case Suffix	Lexical Semantics	Semantic Relation	Example Reference
1. (3.41)	*-ki/ku*	• Verb[+Ditransitive]	k4	
2.	*-ki/ku*	• Verb[+Experience] • Noun[+Animate]	k4a	(3.43)
3.	*-ki/ku*	• Verb[+Experience] • Verb[-Ditransitive]	rt	(3.73)
4.	*-ki/ku*		rh	(3.66)
5.	*-ki/ku*		r6	(3.6)
6.	*-ki/ku*	• Noun[+Time]	k7t	(3.53)
7.	*-ki/ku*	• Verb[+copula] • Noun[-time]	k*u	
8.	*-ki/ku*		adv	(3.120)
9.	*-ki/ku*	• Noun[+place] • Verb[+motion]	k2p	(3.35)

Table 4.4: Relational Ambiguity *-ki/ku*

(4.15) *ravi-∅* *padi-∅* *gaṁṭala-**ku*** *haidarābādu-**ku***
ravi-NOM 10 hour-DAT Hyderabad-DAT
cērukuMṭ-ā-ḍu
reach-PST-3.SG.M
'Ravi will reach Hyderabad at 10 O'clock.'

The *vibhakti -ki* is used to express different relations viz. *sampradāna* (k4) as in 4.12 and goal/destination (k2p) as in 4.13, *anubhava kartā* as in 4.14, *kāladhikarana* & *dēśādhikarana* as in 4.15. In such cases, the semantics of verb is considered to disambiguate the *vibhakti*. In example-4.13, the verb belongs to the class of [+motion] hence it has a requirement of k2p unlike the example-4.12. The other verbs in the class of [+motion] verbs include *vaccu* 'come', *parigettu* 'run', *bayaluderu* 'start', etc. This semantic information is augmented with syntactic rules in order to mark the appropriate relation.

Now, when we compare 4.12, 4.13, 4.14, it can be observed that all the three sentences contains a nominal phrase with the *-ki* suffix. However, it is evident that three nominal phrases represent different relations. In order to differentiate the above two from this, we utilize the semantics of verbs. In 4.14, the verb *telusu* requires an

experience subject. This information of verbs requiring experience subjects is stored as part of the database.

4.7.4 Relational Disambiguation - [*-tō*] suffix

Sl.No.	Case Suffix	Lexical Semantics	Semantic Relation	Example Reference
1.	*-tō*	• Noun[+Instrument]	`k3`	(3.40)
2.	*-tō*	• Verb[+Communicate]	`k4`	(3.42)
3.	*-tō*	• Noun[+Animate]	`ras*`	(3.76)
4.	*-tō*	Noun[+Abstract]	`adv`	(3.118)
5.	*-tō*	• Verb[+Create]	`k5prk`	((3.52-b))
6.	*-tō*		`rh`	(3.67)

Table 4.5: Relational Ambiguity *-tō*

(4.16) *sīta* *atani-**tō*** *nijaM* *cepp-iM-di*
sita.NOM he-INS truth tell-PST-3.SG.F
'sita told him the truth'

(4.17) *sīta* *atani-**tō*** *bājāruku* *velliMdi*
sita-NOM he-ASS market go-PST-3.SG.F
'Sita went to the market with him'

(4.18) *sīta* *kaḷḷa-**tō*** *saiga* *cēsiMdi*
sita-NOM eyes gesture do-PST-3.SG.F
'Sita made a gesture using her eyes'

In the examples-(4.16-4.18), one can notice a nominal with *-tō* marker. *-tō* in all the instances serve a different semantic role.

In order to deduce the relation of 'k4' in 4.16, verbal features are taken into consideration. It so happens that with a verb of [+communication] (Levin, 2015), the beneficiary/receipient of the action of communication verb, the default case marker(*-ki*) can often be replaced with *-tō*. Hence, [+communication] verbs like *ceppu* 'to tell', *sambhāśiMcu* 'converse', *māṭlāḍu*, etc will be stored in the database. Thus, when a nominal phrase with a *-tō* marker occurs along with [+communication] verb, it will be provided the tag *sampradāna*/beneficiary/recipient.

The second instance with *-tō* -4.17, is a relation of association which presumes a [+human] nominal to be part of the action. This cue about *animacy* of the noun is considered to decide if the relation is association (ras*). Any nominal that is

[+animate] and does not contain [+communication] verb, can be given the tag of '*upapada sahakārakatva*(ras_*)/associative'.

Likewise, another instance that exploits the *animacy* feature of nouns is the role of *instrument*. Currently, the system considers [+neuter] nouns and [-communication] verbs as the checkpoints to mark the relation of instrument(*karaṇa*).

4.7.5 Relational Disambiguation- *-lō*

Sl.No.	Case Suffix	Lexical Semantics	Semantic Relation	Example Reference
1.	*-lō*	• Noun[+place]	`k7p`	(3.55)
2.	*-lō*	• Noun[+Time]	`k7t`	(3.54)
3.	*-lō*	Noun[+Abstract]	`k7`	(3.59)
5.	*-lō*	• Noun[+abstract]	`adv`	(3.119)

Table 4.6: Relational Ambiguity *-lō* suffix

4.7.6 Relational Disambiguation- *-nuMḍi/nuMci/niMci*

Sl.No.	Case Suffix	Lexical Semantics	Semantic Relation	Example Reference
1.	*-nuMḍi/nuMci/niMci*	• -Verb[Create]	`k5`	(3.50)
2.	*-nuMḍi/nuMci/niMci*	• Verb[+Create]	`k5prk`	((3.52-b))
3.	*-nuMḍi/nuMci/niMci*		`rh`	(3.69)

Table 4.7: Relational Ambiguity *-nuMḍi/nuMci/niMci* suffix

4.8 Implementation

Rule-based parser is programmed using Ocaml[1]. It is a general purpose programming language efficient in language modelling as well. The run-time is simple and its

[1] https://ocaml.org/

portable. Here, we begin with pre-requisites for the installation process of the system and then discuss its components. The installation process of Ocaml version used to build this parser and various packages required to run the parser are provided [1]. The following commands are used for Installation:

Pre-requisites:
apache HTTP server
bash
bison
flex
graphviz
gcc
g++
lttoolbox
make
perl
python
default-jdk
timeout

4.8.1 Rules

This section explains the rules formulated for analyzing Telugu sentences. The rules are written in Ocaml language. A sample rule is provided below-4.10. All the other rules are given as part of the appendix-2.

```
value rlk1 m1 m2 = match m1 with
 [ Morphana (id1,mid1,word1,rt1,lcat1,gen1,num1,per1,case1,vib1,suff1) ->
        match m2 with
        [ Morphana (id2,mid2,word2,rt2,lcat2,gen2,num2,per2,case2,vib2,suff2) ->
              if lcat2 = "v" && ((lcat1 ="n") || (lcat1="pn")) && ((vib1 = "Z"|| vib1= "Z_e") || (vi
(vib1="I")||(vib1="Z_V")) && not(member_of rt1 time_nouns) && not(member_of rt1 psp)&& id2>id1
              then [ Relation (id1,mid1,"k1",id2,mid2,"k1.1")]
             else if lcat2 = "v" && ((lcat1 ="pn" && vib1="e_vAdu" || lcat1="n" && vib1="A")) && id
              then [ Relation (id1,mid1,"k1",id2,mid2,"k1.2")]
              else []
        ]
 ]
;
```

Figure 4.10: Sample Rules

[1]For any further installation procedure follow the instructions provided in SCL repository at https://github.com/samsaadhanii

Installation commands for Pre-requisites
sudo apt-get install apache2 bash bison flex graphviz gcc lttoolbox make perl python xsltproc default-jdk timeout g++
Install Objective Caml (ocaml-4.07.1.tar.gz) available at http://ocaml.org/releases/ ./configure make world.opt sudo make install
Install Ocamlbuild from (ocamlbuild-0.13.0.tar.gz) available at https://github.com/ocaml/ocamlbuild/releases make configure ./configure make sudo make install
Then install Camlp4 (camlp4-4.07-1.tar.gz) https://github.com/ocaml/camlp4/releases ./configure make all sudo make install
Install Zen (zen-master.tar.gz) available at https://gitlab.inria.fr/huet/Zen.git cd Zen/ML make
sudo a2enmod cgid sudo systemctl restart apache2

Table 4.8: Installation pipeline

Rule-based parser consists of rules for various relations in Telugu. This is not an exhaustive list, however, can be improved whenever a new relation is encountered. Sample rules along with their algorithm will be provided in the section-(4.8.3) of this chapter. Rules of the parser are based on the descriptive grammar of Telugu which cover almost all types of constructions, exceptions and probable structures in the language. Hence, grammar books were referred to frame the rules (Also, mentioned in chapter-1 of this thesis) viz.., *telugu vākhyaM* (Ramarao, 1975), A reference grammar of modern Telugu (Ramarao, 2017), *bālavyākaraṇaM* (Chinnaya Suri, 1855), A grammar of Modern Telugu (Krishnamurti and Gwynn, 1985), Non-Nominative Subjects (Bhaskararao and Subbarao, 2004), The Dravidian Languages (Krishnamurti, 2003b), Experiencer

Subject in South Asian Languages (Verma and Mohanan, 1990).

4.8.2 Database

Database is the lexical reserve for the parser to provide a required environment to parse the sentence and mark exceptions. This component of the parser consists of lists of noun classes, verb classes, indeclinables, specific semantic features of lexical categories and the like. These lists stand as cues for disambiguating certain relations. Database is augmented with RBP rules in-order to produce the correctly parsed trees. Efficiency of the parser depends on the richness of database for which a continuous updation is needed. The role of database in a rule-based parser is emphasized throughout this thesis. Table-(4.9) provides the list of databases used in RBP:

4.8.2.1 Filter Module

Rule-based parsers are affected by over-generation and this parser is no exception. Hence, a filter module is introduced to facilitate pruning out of extra, unwanted relations between words after all other rules & databases have been applied. This will be explicated in detail as part of section-(4.19).

4.8.2.2 Filtering

In this step, if two words are joined by more than one directed arc, filtering is evoked. This step ensures the pruning out of impossible edges. For instance, when a sentence(consisting of a single verb) is marked for two *kartas*, filtering program filters out the edge which is least probable. After filtering, graphs are free from ambiguity. In case of a Telugu parser, filtering was used for majorly for certain relations like *karta*(k1) vs *karma*(k2), *karta(k1)* vs *karta samānadhikarana*(k1s), *karma* vs *karma samānadhikarana* etc. Let us consider an example to understand filtering:

(4.19) *āyana* *mā* *nānna*
He.NOM our father
'He is my father'

Step-1

Each word in the sentence *āyana*, *mā*, *nānna* are considered as separate nodes. As there is no verb in the sentence, a null verb is inserted in the pre-processing stage.

Step-2

If the noun contains a -∅ suffix & the verb is part of the copula list, then the noun

Sl.No	Name of the database	No. of words
1	Transitive verbs	2478
2	Ditransitive verbs	18
3	Experience verbs	25
4	Motion verbs	18
5.	Create verbs	20
6.	Light verbs	112
7.	Quantifiers	168
8.	Question words	271
9.	Interjections	15
10.	Place nouns	25
11.	Time nouns	35
12.	Animate nouns Male nouns Female nouns	5263 4120 1143
14.	Abstract nouns	65
15.	Adjectives	360
16.	Nouns of Place and Time	559
17.	Post-positions	107
18.	Intensifiers	12
19.	*avyaya* *avyaya_vmod* *avyaya_particles* *avyaya_determiners*	134

Table 4.9: Statistics of database

immediately preceding the verb is the *karta samānādhikarana*.

Step-3

Though this rule works for "k1s", a noun with -∅ suffix, can also be a *karta* leading to two interpretations. The graph for the above sentence will be provided like seen in the figure-(4.11).

Step-4

In the figure-(4.19), we can see that there are two incoming nodes from the root verb and *nānna*. Also, *karta* is marked twice. Filtering works on such input by filtering out the redundant tags. Hence, if `k1` is already available in the input text, it prunes out *karta* that has more than one directed arc. Consider the final parsed output

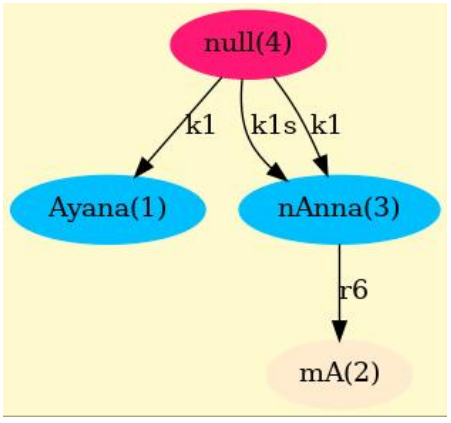


Figure 4.11: Parser analysis of 4.19

after filtering stage: Similarly, other relations are also pruned-out using this process.

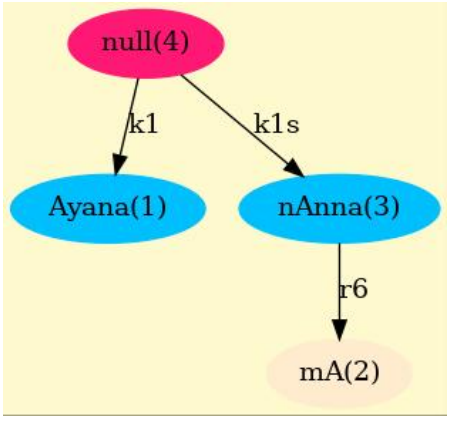


Figure 4.12: Parser analysis after filtering of 4.19

4.8.2.3 Generating trees

After the filtering stage, when every node in the tree is connected to only one edge, the parser generates the trees.

4.8.3 Parser Rules

In this section, each rule in the parser program (coded in wx notation) will be listed with a rule description followed by its algorithm. Rules are framed looking at input sentence and their disambiguated morphological information. A verb is identified and it's corresponding relations are marked based on various linguistic cues. To identify *kāraka* and non-*kāraka*(both dependency and non-dependency) relation; root, and other morphological features such as lexical category, gender, number, person, case marker and other suffixes are marked with expectancy of the

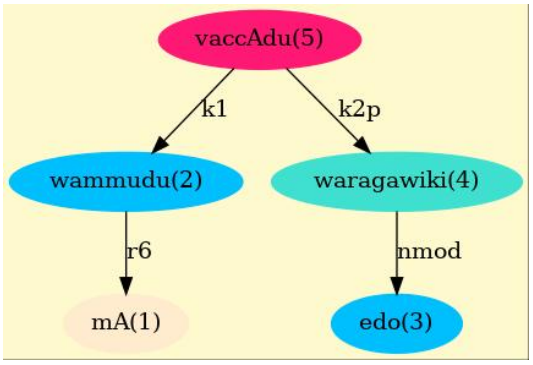


Figure 4.13: Sample generated trees from the parser

word

In the program, we frame rules at two levels: In the first level, rules will be based on only two words in the sentence. One of which is mandatorily a verb for *kārakā* relations. The other one include the noun. However, in other modifier relations like `nmod, r6, psp` etc., a pair of noun and noun will be considered and other categories wherever necessary. It depends on the type of relation under study. However,sometimes, it requires more than two words to form a relation. For example, *mēmu haidarābādu nuMḍi vaccāmu* 'we came from Hyderabad', *haidarābādu nuMḍi* together should be an *apādāna* to the verb *vaccāmu*. This analysis requires three words to form a relation. In such circumstances, next level of rules are formed. It involves three words.

4.8.3.1 Rules for *kartā* (`k1`)

kartā in Telugu is identified with null case marking on the lexical category of nominal[1] in active voice. In passive voice, *kartā* takes the suffix cē/cēta.

Rule-1

Cue 1: The token with the category (lcat) noun/pronoun (N/PN) with the case [-∅], when its gender-number-person(GNP) matches with the finite verb(V)'s GNP.

AND

Cue 2: When the verb is not in passive voice i.e. the suffix in the TAM[2] slot is not equal to "a_baḍu",i.e., the infinitive form followed by the passive suffix.

AND

Cue 3: The noun (ROOT) is not a part of temporal noun list N[+time]

Code

[1]Nominal can be a noun/pronoun/proper noun/place noun and the like

[2]Tense, Aspect, Mode

```
1. [cat == N || PN] && vib =="-∅";
2. if:
3.    N[GNP] == V[GNP]
4.    N[ROOT] != N[+time];
5.    V[TAM] != "a_badu";
6. then:
7.    tag = "k1"
```

Rule-2

Cue 1: The word with the category (lcat) noun/pronoun (N/PN) with the case *cē/cēwa*, when its gender-number-person(GNP) does not match with the finite verb(V)'s GNP.

AND

Cue 2: Verb must be part of [+transitive] verb list. And when the verb is in passive voice i.e. the suffix in the TAM slot is equal to "a_baḍu"

AND

Cue 3: When the root noun (ROOT) is not a part of temporal noun list (N[+time)]

Code

```
1. [lcat == N || PN] && [vib == "-ce/cewa];
2.   if:
3.     N[GNP] != V[GNP];
4.     N[ROOT] != N[+time];
5.     V[ROOT] = V[+transitive];
6.     V[TAM] = "a_badu";
7.   then:
8.     tag = "k1"
```

Rule-3

Cue 1: The word with the category (lcat) noun/pronoun (N/PN) with the case suffix *aḍaM*, when its gender-number-person(GNP) matches with the finite verb(V)'s GNP.

AND

Cue 2: Verb must be part of [-transitive] verb list.

AND

Cue 3: When the root noun (ROOT) is not a part of temporal noun list (N[+time)]

Code

```
1. [lcat == N || PN] && [suffix == "aḍaM];
2.   if:
3.     N[GNP] = V[GNP];
4.     N[ROOT] != N[+time];
5.     V[ROOT] = V[-transitive];
6.   then:
8.     tag = "k1"
```

4.8.3.2 Rules for *karma*(k2)

A combination of two words in a sentence, one of which is mandatorily a verb. And a noun/pronoun with the case marking *-ni/nu* is marked as k2 when it satisfies the following cues:

Rule-1

Cue 1: The token with the category (lcat) of noun/pronoun (N) with the *vibhakti -ni/nu*. Or when the *vibhakti* is -∅ and gender(GEN) of the noun is equal to neuter N[+neuter] or token with lcat verb and suffix=aḍaM/aḍaM_nu

AND

Cue 2: When the verb is not in passive voice i.e. the suffix in the TAM slot "a_baḍu", the infinitive form_the passive suffix & verb is in transitive verb list

AND

Cue 3: When the root noun (ROOT) is not a part of temporal noun list (N[+time)]

Code

```
1.[lcat == N||PN && [suffix =="-ni/nu/"] ||[lcat == N||PN] && [suffix
=="-∅"] || [lcat == v && [suffix =="-aḍaM_nu/ni"] ;
2.  if:
3.    N[GNP] != V[GNP];
4.    N[ROOT] != N[+time]
5.    V[ROOT]== [+transitive];
5.    V[TAM] != "a_badu";
6.  then:
7.    tag = "k2"
8.  elif:
9. lcat == N || PN && suffix =="-∅";
10.  if:
11.    N[GNP] != V[GNP];
12.    N[GEN] == N[+neuter];
13.    N[ROOT] != N[+time];
14.    V[ROOT] ==[+transitive];
15.    V[TAM] !="a_badu";
16.  then:
17.    tag = "k2"
```

Rule-2

Cue 1: The word with the category (lcat) noun/pronoun (N/PN) with the case [-∅], when its GNP does match with the finite verb(V)'s GNP.

AND

Cue 2: Verb must be a part of [+transitive] verb list. And when the verb is in passive voice i.e. the suffix in the TAM slot is equal to "a_baḍu", the infinitive form of the passive suffix.

AND

Cue 3: When the root noun (ROOT) is not a part of temporal noun list (N[+time)]

Code

```
1. lcat == N || PN && vib == "-∅";
2.   if:
3.     N[GNP] = V[GNP];
4.     N[ROOT] != N[+time];
5.     V[ROOT] = V[+transitive];
6.     V[TAM] = "a_badu";
7.   then:
8.     tag = "k2"
```

4.8.3.3 Rule for *gauna karma* (k2g)

Rule-1

Cue 1: The word with the category (lcat) noun/pronoun (N/PN) with the case [-ni/nu], when its GNP does match with the finite verb(V)'s GNP.

AND

Cue 2: Verb must be a part of [+Communicative verbs] verb list.

AND

Cue 3: When the root noun (ROOT) is not a part of temporal noun list (N[+time)]

AND

Cue-4: A noun with vib=*ni/nu* should precede it

Code

```
1. lcat == N || PN && vib == "-ni/nu";
2.   if:
3.     N[GNP]! = V[GNP];
4.     N[ROOT] != N[+time];
5.     V[ROOT] = V[+communicative] || V[+Transitive];
6.   then:
7.     tag = "k2"
```

4.8.3.4 Rules for *karana*(k3)

Rule-1

A noun/pronoun with the case marking -*tō/dwārā* is marked as k3 when it satisfies the following cues:

Cue 1: The word with the category (lcat) noun/pronoun (N/PN) with the case -*tō/dwārā* and gender(GEN) is N[+neuter]

Code

```
1. lcat == N || PN && suffix == "-wo/XvArA";
2.   if:
3.     N[GEN] = N[+neuter];
4.     V[ROOT] != V[+copula];
5.   then:
6.     tag = "k3"
```

4.8.3.5 Rule for *sampradāna*(k4)

A noun/pronoun with the case marking *--ki/ku* is marked as k4 when it satisfies the following cues:

Rule-1

Cue 1: The word with the category (lcat) noun/pronoun (N/PN) with the case *--ki/-ku*

AND

Cue 2: Verb is verb[+ditransitive]

Rule-2

Cue 1: The word with the category (lcat) noun/pronoun (N/PN) with the case *--ki/-ku/tō*

AND

Cue : When the verb is part of the communicative verb list

Code

```
1. lcat == N || PN && suffix == "-ki/-ku";
2.   if:
3.     V[ROOT] = V[+ditransitive];
4.   then:
5.     tag = "k4"
```

Code

```
1. lcat == N || PN && suffix == "-ki/-ku/wo";
2.   if:
3.     V[ROOT] = V[+communicative];
4.   then:
5.     tag = "k4"
```

4.8.3.6 Rule for *apādā*na (k5)

A noun/pronoun with the vib-*nuMḍi/nuMci/niMci* is marked as k5 to the nearest verb it encounters.

Code

```
1. lcat == N || PN] && vib == "nuMḍi/nuMci/niMci";
2.   then:
3.     tag = "k5"
```

4.8.3.7 Rule for *prakruti apādāna* (k5prk)

Cue-1

A noun/pronoun with the case marking -*nuMḍi/nuMci/niMci* is marked as k5 to the nearest verb it encounters.

AND

Cue-2

Verb belong to V[+create] list of verbs.

Code

```
1. lcat == N || PN && vib == "nuMḍi/nuMci/niMci";
2.   if:
3.   V[ROOT] = V[+create]
2.   then:
3.     tag = "k5prk"
```

4.8.3.8 Rule for*kāladhikarana* (k7t)

Rule-1 A noun/pronoun with the case marking --*lō/ki/ku/-∅/-∅_e* is marked as k7t when it satisfies the following cues:

Cue 1: The word with the category (lcat) noun/pronoun (N/PN) with the case -*ki/ku/-lō/-∅/-∅_e*

AND

Cue 2: Noun is part of the N[+Time] list

Code

```
1. lcat == N || PN && (vib == "-ki/ku/ || vib=lo ||vib=-∅/;)
2.   if:
3.     N[ROOT] = N[+time];
4.   then:
5.     tag = "k7t"
```

4.8.3.9 Rule for *deshādhikarana* (k7p)

Rule-1 A noun/pronoun with the case marking --*lō/ki/ku/pai/mīda/lōpala* is marked as k7t when it satisfies the following cues:

Cue 1: The word with the category (lcat) noun/pronoun (N/PN) with the case --*lō/ki/ku/pai/mīda/lōpala*

AND

Cue 2: Noun is part of the N[+place] list && verb is not part of the V[+motion] list

Code

```
1.  lcat == N || PN && (vib == ‘‘--lō"|| vib=‘‘ki" || vib= ’’ku" || vib =
"pai" || vib=‘‘mīda" || vib=‘‘lōpala");
2.  if:
3.    N[ROOT] = N[+place];
4.    V[ROOT] != V[+motion]
5.  then:
6.    tag = "k7p"
```

4.8.3.10 Rule for *vishyādhikarana* (k7)

Rule-1 A noun/pronoun with the case marking *--lō/lōni/* is marked as k7 when it satisfies the following cues:

Cue 1: The word with the category (lcat) noun/pronoun (N/PN) with the case *--lō/lōni*

AND

Cue 2: Noun is not part of the N[+place] or N[+time] list

Code

```
1. lcat == N || PN && vib == "--lō/lōni";
2.  if:
3.    N[ROOT] != N[+place];
4.    N[ROOT] != N[+time]
5.  then:
6.    tag = "k7"
```

4.8.3.11 Rule for Goal/Destination (k2p)

A noun/pronoun with the case marking *--ki/ku/∅* is marked as k2p when it satisfies the following cues:

Cue 1: The word with the category (lcat) noun/pronoun (N/PN) with the case *--ki/ku/∅*

AND

Cue 2: Noun is part of the N[+place] && V[+motion] list of verbs

Code

```
1.[lcat == N || PN] && [suffix == ‘‘-ki/ku"] || vib=‘‘∅";
2.  if:
3.    N[ROOT] = N[+place];
4.    V[ROOT] = V[+Motion]
5.  then:
6.    tag = "k2p"
```

4.8.3.12 Rule for *kartā samānādhikarana*(k1s)

Cue 1: The word with the category (lcat) noun/pronoun/adjective (N/PN/ADJ) with suffix (number person) [*-∅/ni/nu/mu/vu/vi/gāru/ni_e*], when its GNP does

not match with the finite verb(V)'s GNP.

AND

Cue 2: Verb must be a part of [+copula] verb list.

AND

Cue 3: When the root noun (ROOT) is not a part of temporal noun list (N[+time)]

Code

```
1. lcat == N || PN || ADJ && suffix == "--∅/ni/nu/mu/vu/vi/gāru/ni_e";
2.   if:
3.     N[GNP] != V[GNP];
4.     N[ROOT] != N[+time];
5.     V[ROOT] = V[+Copula];
6.     ID of verb = ID of noun + 1;
6.   then:
7.     tag = "k1s"
```

4.8.3.13 Rule for *anubhava kartā*(k4a)

Cue 1: The word with the category (lcat) noun/pronoun (N/PN) with the case [*-ki/ku*], when its GNP does not match with the finite verb(V)'s GNP.

AND

Cue 2: Verb must be a part of [+experience] verb list.

AND

Cue 3: When the root noun (ROOT) is not a part of animate noun list (N[+animate)]

Code

```
1. lcat == N || PN && suffix == "-ki/ku";
2.   if:
3.     N[GNP] != V[GNP];
4.     N[ROOT] != N[+animate];
5.     V[ROOT] = V[+experience];
6.   then:
7.     tag = "k4a"
```

4.8.3.14 Rule for `k*u`

Cue 1: The word with the category (lcat) noun/pronoun (N/PN/ADJ) with the case [*-kaMṭe/ki/kanna*], when its GNP does not match with the finite verb(V)'s GNP.

Code

```
1. lcat == N || PN && vib == "kaMṭe/ki/kanna";
2.   if:
3.     N[GNP] != V[GNP];
4.     N[ROOT] != N[+animate];
5.     V[ROOT] = V[+experience];
6.   then:
7.     tag = "k4a"
```

4.8.3.15 Rules for `rh`

Rule-1

A noun/pronoun with the case marking *-vala/valana* is marked as `rh` to the nearest verb it occurs with.

Algorithm

```
1. lcat == N|PN && case =="-valla/valana"
2.     tag = "rh"
```

4.8.3.16 Rules for `rt`

Rule-1

A noun/pronoun with the case marking *-kōsaM/koraku/aḍaM_ki/koM_aḍaMki/ki* is marked as `rt` when it satisfies the following cues:

Cue 1: The word with the category (lcat) noun/pronoun (N) with the case *-kōsaM/koraku/aḍaM_ki/koM_aḍaMki/ki*

AND

Cue 2: When the verb is not part of experience verb list or ditransitive verb list

Code

```
1. lcat == N|PN && case =="-kōsaM/koraku/aḍaM_ki/koM_aḍaMki/ki"
2.     V!== member_of_experience_verbs ||member_of_ditranisitive_verbs
2.     tag = "rt"
```

Other dependency relations (as shown in table-3.5) and non-dependency relations(see table 3.10) and miscellaneous relations are identified using the default markers in Telugu. Mapping the marker with the dependency label is attempted to identify them like the rules mentioned earlier.

Chapter 5

Evaluation of the Parser and Error Analysis

5.1 Introduction

Evaluation of the system is one of the crucial phases of any language technology tool. As NLP involves with different tasks on language automation, different evaluation techniques are used. Parsers are evaluated using a defined metrics. In this chapter, an in-detail evaluation of rule-based parser for Telugu as well as the evaluation of each dependency relation is provided. In addition to this, confusion matrix pertaining to each relation is outlined thereby deducing the sources of errors. Further, errors sources involving pre-processing tools, databases and parser algorithm are explicated.

5.2 Metrics to Evaluate a Parser

The correctness of parser output can be measured using various metrics. One common method employed for almost all computational modules is to compare the system generated output with the gold-data created by human annotators. However, this does not provide a complete evaluation of the parser as this looks for an exact (0-1 metric) or word-word mapping (here, relation to relation mapping). Hence, other metrics are used for evaluating the parser. This parser considers the following metrics:

1. Attachment Scores
2. Precision and Recall
3. Relation-based Performance Index
4. Confusion Matrix.

5.2.1 Attachment Scores

Attachment scores (Nivre, 2009, pp-4) primarily include, Labelled Attachment Score (LAS) and Unlabelled Attachment Score (UAS). LAS is a metric that considers both the relation label and edges of the relations (directed edges from the head to the dependent). In other words, LAS evaluates the 'proportion of words that are assigned the correct head and a correct dependency relation'. LAS requires both edges and labels to be correctly parsed and to match the total number of edges in the actual tree. For instance, if a sentence contains 4 words, typically, it should contain 3 relations. The sentence with 4 words has an LAS of 100% if the 3 labels along with their edges are correctly parsed.
In UAS, only the edges are considered not the relation label. Hence, if 3 words are correctly connected to their heads, irrespective their labels, UAS is 100%.
Another metric include the Label Accuracy (LA) in which only the label irrespective of the edge correctness is calculated.
In addition to this, One Wrong Attachment Score (OWAS) is also employed when necessary. OWAS constitutes sentences that are wrong only with respect to one attachment. There are also other metrics like Dependency Accuracy (DA), Root Accuracy (RA) and Complete Match (CP) which are not considered for this study.

5.2.2 Precision and Recall

The other common evaluation metric used to evaluate the performance of the parser is the precision and recall (Jurafsky, 2000). Precision measures the total number of correct relations that are identified by the system. Whereas recall measures the correct relations identified by the system with respect to the total number of relations in the input sentence. The formulae of precision and recall are as follows:

- **Precision**

$$\frac{\text{Number of correct relations marked by the system}}{\text{Total number of relations identified by the system}}$$

- **Recall**

$$\frac{\text{Number of correct relations marked by the system}}{\text{Total number of relations}}$$

5.2.3 Relation-based Performance Index

Relation-based performance index is an evaluation of each relation in order to identify which relation performs better and which performs poorly. Each relation is evaluated for its precision & recall or LAS & UAS. This metric enables to further improve the performance of relation that performs poorly.

5.2.4 Confusion Matrix

Confusion matrix attempts to evaluate the system for the number of times a relation is confused with some other relation. This metric is useful in predicting the frequently confused relations as well as designing strategies for further improvements.

5.3 Evaluation of Pre-processing tools

Parser, as discussed in the earlier chapter, being a higher-level application requires a number of pre-processing tools for its functioning. Hence, we begin with the evaluation of pre-processing tools. For RBP, we have used a tokeniser, morphological analyser, POS tagger and Pruning modules. The evaluation of pre-processing tools is as shown in the table-(5.1):

Sl.No.	Pre-Processing Tool	Precision	Recall
1.	Tokenizer	100%	97.2%
2.	Morphological Analyser	99.66%	99.22%
3.	POS tagger	98.68%	98.66%
4.	Pruning	98.23%	97.87%

Table 5.1: Module-wise evaluation of pre-processing tools

5.4 Evaluation of Rule-Based Parser for Telugu

In this section, the data used for evaluating parsers is presented followed by the results. Based on the results, we present the error analysis and some observations.

5.4.1 Data

The present study selects **1000** sentences to test the Telugu rule-based parser. The test corpus is extracted from Telugu corpus (3 million words CALTS[1] corpus). The corpus consists of the following sentence types:

Sentence Type	**No. of sentences**
Simple Sentences	448
Verbless	
Intransitive	
Transitive	
Ditransitive	
Passive	
Causative	
Non-nominative	
Interrogative	
Complex Sentences	489
Complement Clauses	
participial clauses	
conditional clauses	
concessive clauses	
Compound	63
Coordination	
Total Number	1000

Table 5.2: Types of sentences in the test data

Table-(5.3) shows the number of words in each sentences in the test data. The average length of the sentence is around 5 words per sentence.

5.4.2 Results

This section presents the results of the parser using the above discussed evaluation metrics. It is found that the test data contains a total of 3867 relations. The precision and recall of the system is calculated as below:

- **Precision & Recall**
- **Attachment Scores**

[1]Centre for Applied Linguistics and Translation Studies, University of Hyderabad

Number of words	No. of Sentences
Two words	93
Three words	191
Four words	220
Five words	198
Six words	151
Seven words	70
Eight words	47
Nine words	6
Ten words	8
Eleven words	9
Twelve words	7
Total number of tokens	4967

Table 5.3: Length of sentences and distribution of test data

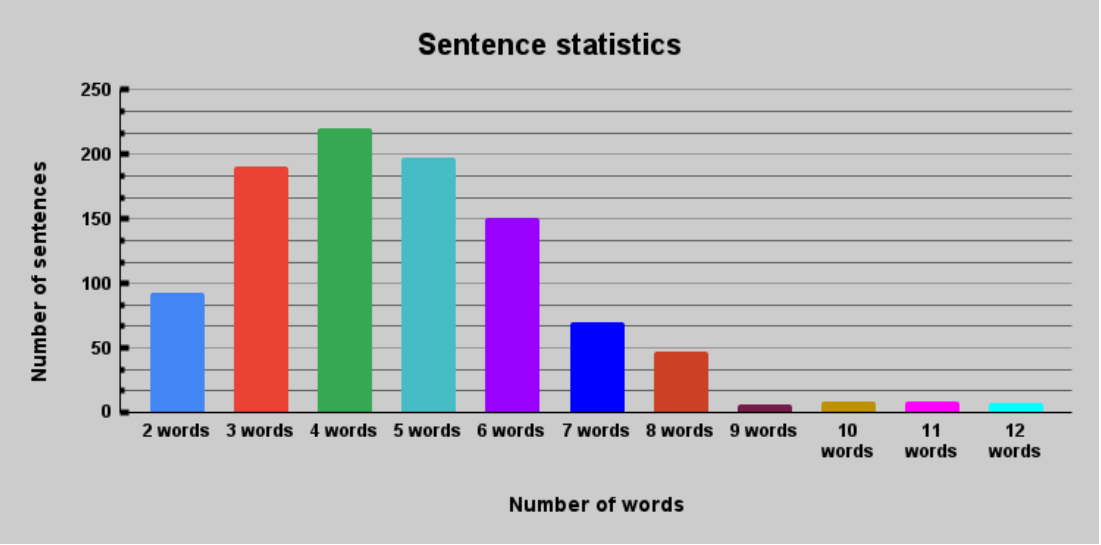


Figure 5.1: Sentence Statistics

Total number of relations	3867
Total number of identified relations	3601
Total number of correctly marked relations	3255
Precision	90.3%
Recall	84.1%

Table 5.4: Precision and Recall

LAS	UAS	LA
84.1%	90.3%	86.2%

Table 5.5: Attachment Scores

- **Comparison of existing Telugu Parsers**
  Gatla (2019) in the paper titled 'Dependency Parsing for Telugu Using Data-driven Parsers' provided following results of the dataset using MST & MALT:

Parsers	LAS	UAS	LA
MALT	**79.67**	**92.35**	**82.45**
MST	73.62	91.44	76.30

Figure 5.2: Results (retrived from http://languageinindia.com/jan2019/praveengatladependencyteluguparser1.pdf)

Other work on Telugu dependency parsing include the work by Rama and Vajjala (2018) titled 'A Dependency Treebank for Telugu'. The results are as follows:

Input features	POS Acc.	LAS	UAS
Tagging + Parsing	90.43%	74.76%	87.79%
Parsing (Gold POS tags)	–	78.50%	89.74%

Figure 5.3: Results (retrieved from https://aclanthology.org/W17-7616.pdf)

Work on dependency parsing for Telugu by Nallani et al. (2020c) is provided in the following table-(5.4):

5.4.3 Relation-based Performance Index

For Relation-based performance index, we considered almost all the dependency relations encountered in the test data. Some minor relations like the negative particles, address terms, etc are calculated as part of miscellaneous relations. A total of 35 relations including the miscellaneous relations are included in the table (5.6). The statistics of occurrence of each relation is provided in the bar graph-(5.5).

Intra-chunk	UAS	LS	LAS
Max	95.43	83.05	81.81
Min	85.04	67.09	64.17
Average	90.92	71.95	70.49

Table 1: Parser 10-fold cross-validation results on intra-chunk annotated treebank.

Inter-chunk	UAS	LS	LAS
Max	94.50	78.90	77.20
Min	78.16	56.14	52.14
Average	90.37	67.57	65.74

Table 2: Parser 10-fold cross-validation results on inter-chunk annotated treebank.

Figure 5.4: Results (retreived from https://aclanthology.org/2020.acl-srw.19.pdf)

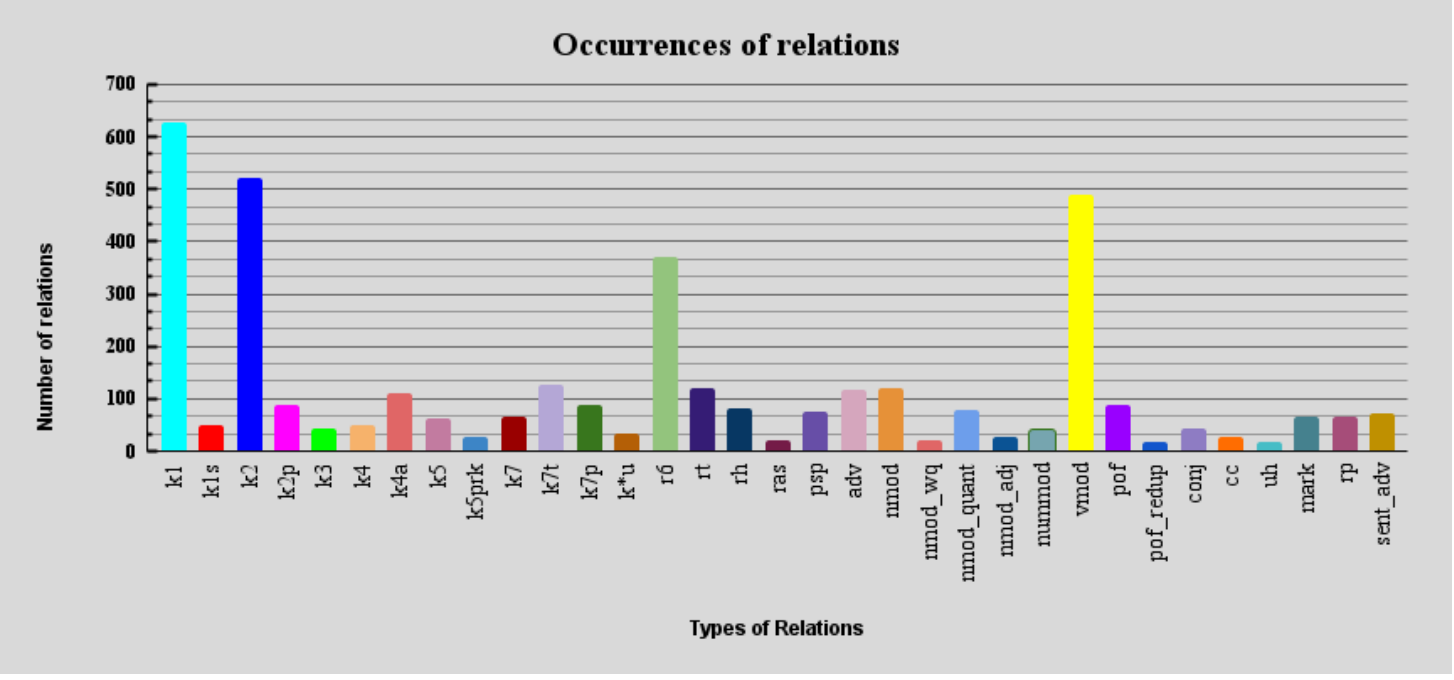


Figure 5.5: Statistics of Relations in Test Data

5.5 Error Analysis and Observations

In this section, we discuss the cases where the rule-based parser fails to provide the appropriate results. We present the reasons for failure of the rule-based parser in certain cases. We also show how rule-based parser proves to be better in solving with language ambiguities than the existing data-driven parsers. We divide this section into three stages:
(1). Pre-processing Errors
(2). Database Errors
(3). Issues with Rules

5.5.1 Pre-Processing Errors

It is found that in certain cases, reasons for the failure of the parser in certain cases is due to the wrong output from the pre-processing tools. The following sections discuss the pre-processing:

5.5.1.1 Tokenization and *Sandhi* Split Errors

Sandhi Split errors contribute to the system errors to an extent. Sometimes, *sandhi* splitter leads to errors that cannot even generate a sentence. Consider the following error from the test data:

(5.1) `(morphana (id 4) (mid 1) (word appulapAlayyAru)`

`(rt appulapAlayyAru) (lcat unk) (gen X) (num X) (per 3)`

`(case X) (viBakwi Z) (suff Z))`

In (5.1), *appulapāluayyāru* 'to make debt', should be split into two words viz., *appulapālu* & *ayyāru*. However, because the *sandhi* splitter fails to do so, the category of the word is not identified. Some such instances included the un-identification of the finite verb which is the root of the graph. So, the sentence do not get generated.

(5.2) `(morphana (id 1) (mid 1) (word ippudalAkAxu)`

`(rt ippudalAkAxu) (lcat unk) (gen X) (num X) (per 3) (case X)`

`(viBakwi Z) (suff Z))`

The *sandhi* splitter fails to split the word *ippuḍalākādu* 'It is not so now' into three words *ippuḍu* 'now', *alā* 'like that', *kādu* 'not'.

5.5.1.2 Morphological Errors

Morphological errors affects the output of the parser the most. This is due the fact that rule-based parser requires correctly morphologically analysed analysis. Hence, a minor error in this analysis can lead to multiple parser errors. In this section, we discuss words which are not identified for the lexical category, wrong GNP analysis and incorrect roots.

5.5.1.3 Unknown Words

Morphological analyser fails to provide the lexical category of certain words like proper nouns and when the *sandhi* splitter does not split the words into its respective constituents. Consider the following instances of unknown lexical category.

(5.3) `(morphana (id 3) (mid 1) (word akRaraxIpaM) (rt akRaraxIpaM)`
`(lcat unk) (gen X) (num X) (per 3) (case X) (viBakwi Z)`
`(suff Z))`

(5.4) `(morphana (id 1) (mid 1) (word parAyixAnnilA)`
`(rt parAyixAnnilA) (lcat unk) (gen X) (num X) (per 3) (case X)`
`(viBakwi Z) (suff Z))`

(5.5) `(morphana (id 2) (mid 1) (word pacciwAgubowu)`
`(rt pacciwAgubowu) (lcat unk) (gen X) (num X) (per 3)`
`(case X)(viBakwi Z) (suff Z))`

In ex-(5.3), *akśaradīpaM* is a proper noun and the system could not identify the lexical category leading to the unidentified relation. Likewise, examples-(5.4),(5.5) words 'parāyidānilā' 'as an outsider' & *paccitāgubōtu* 'drunkyard' also are unknown words and are classified as part of unknown word errors. All such unknown word errors contribute to the decreased recall rate.

5.5.1.4 Lexical Category, Gender, Number, Person Errors

Morphological analysis constitute 8 levels of analysis viz. root, lexical category, gender, plural, person, case, *vibhakti* and suffix. All these categories are considered for building the parser. Error in any of these information will lead to parsing errors. The following examples include errors and some missing information.

(5.6) *oVkati [fs af='oVkati,n,,sg,,d,0,0']*

In example-(5.6), lexical category is expected as a number(num). But it is marked as a noun.

(5.7) *pattinavAru [fs af='pattu,pn,,pl,,,ina_vAru,ina_vAru_0']*

Likewise in (5.7), lexical category for the root word is 'verb', but the category for the derived word is given.
In order to mark several relations, gender information is utmost important. For instance, to mark `k2` gender information is a must. If the morphological analyser fails to mark gender information, it contributes to wrong relations thereby decreasing the precision.

5.5.1.5 Incorrect Root Errors

Incorrect root errors indirectly contribute to the database issues. In the database, words are stored in the form of respective roots and are retrieved accordingly. However, when the root is incorrectly marked by the system, system cannot retrieve the correct information. Hence, this leads to wrong relations. In example-(5.8), the English word 'cars' is incorrectly given its root as *kārl*.

(5.8) `(morphana (id 1) (mid 1) (word kArlu) (rt kArl) (lcat n)`

`(gen X) (num sg) (per 3) (case d) (viBakwi Z) (suff Z))`

5.5.1.6 Pruning errors

Different lexical categories are disambiguated by POS. However, if the token has same lexical category, POS does not distinguish between them. For this reason, pruning analysis is required. So, when a word does not show any difference in its direct or oblique form, pruning analysis randomly picks one of them for further processing. This also might lead to wrong analysis. Consider (5.9):

```
1 (morphana (id 1) (mid 1) (word nAku) (rt nenu) (lcat pn) (gen any) (num sg) (per 1) (case X) (viBakwi ki) (suff ki))
2 (morphana (id 2) (mid 1) (word peVlYli) (rt peVlYli) (lcat n) (gen X) (num sg) (per 3) (case d) (viBakwi Z) (suff Z))
3 (morphana (id 3) (mid 1) (word BayaM) (rt BayaM) (lcat n) (gen X) (num sg) (per 3) (case d) (viBakwi Z) (suff Z))
4 (morphana (id 4) (mid 1) (word pattukoVniMxi) (rt pattuko) (lcat v) (gen fn) (num sg) (per 3) (case X) (viBakwi A) (suff
```

Figure 5.6: Pruning error

In (5.6), *peḷḷi bhayaM* 'the fear of marriage' is a noun phrase in which *peḷḷi* 'marriage' is in oblique form. However, the pruning analysis marks its case as direct case. Hence, the relation of `nmod` is not identified leading to a wrong relation.

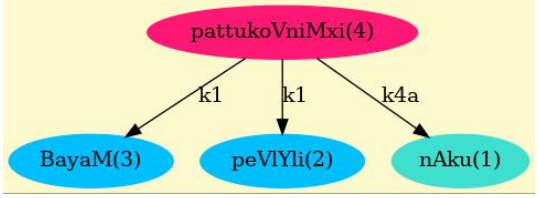


Figure 5.7: Error in parsed output due to pruning errors

5.5.2 Database Issues

Database issues give rise to ambiguity in relation marking. When the required item is missing from the database, the parser cannot identify the relation or marks wrong relations. Consider (5.9):

(5.9)

mā	*amma*	*kōpaM-tō*	*vāḍini*	*koṭṭ-iM-di*
we-POSS	mother	anger-INS	he.ACC	beat-PST-3.SG.F

'My mother beat him in anger'

Here, *kōpaMtō* 'in anger' functions as a manner adverb. The only way to mark is relation is to list *kōpaM* as part of the abstract nouns. However, when it is missing from the list, there is a possibility of marking it either as a `k3` or `ras`.
Likewise, for `k2` relation, if the input contains [-animate] noun as part of the object and if it is missing from the database, it will be marked as `k1`. There are many such cases where database plays a crucial role in marking correct relation. Hence, database issues contribute immensely to the increase in wrong relations. Consider the wrongly parsed (5.8) graph below:

5.5.3 Issues with rules

It so happens that when there is no environment to mark a relation, words will not be connected to the root or their respective heads. This can happen due to issues with rules as well as database issues. Consider (5.10):

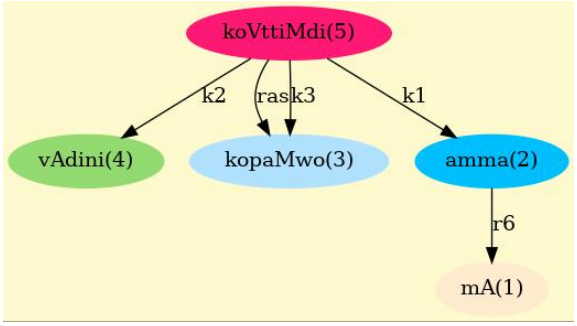


Figure 5.8: Error in parsed output due to database issues

(5.10) *ippuḍu* *nā* *kaMṭiki* *pedda* *ayipōyāḍu*
now I-POSS eyes-DAT big become-PST.3.SG.M
'Now, he seems big to my eyes'

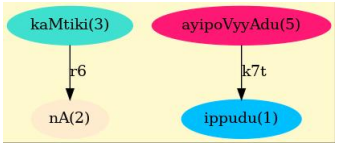


Figure 5.9: Rule issue

In the figure-(5.9), *nā kaMṭiki* 'my eyes' is not connected to the root. This happens when there is no appropriate rule that matches this relation. These also contribute to unidentified relations leading to less recall.

5.5.4 Dummy verb Insertion

The importance of verb for RBP has been emphasized in the previous chapters. For a parse to generate in RBP, a finite verb is a pre-requisite. However, for verb-less sentences, RBP automatically adds a dummy verb. This process gets difficult when a word with POS tag 'verb' already exists in a sentence which is not a finite verb. In such cases, system provides a wrong analysis. Consider (5.11):

(5.11) *adi* *cūḍaḍaM* *nāku* *bāgā* *gurtu*
that see-GER I-DAT more remember
'I remember seeing it profoundly'

In the above example, a gerund *cūḍaḍaM* 'seeing' is present. POS tagger marks it as verb with *aḍaM* suffix. In such cases, when a verb is already present, RBP might not insert a dummy verb leading to missing root. Hence, the rules are further improved.

5.5.5 Over-generation

In certain cases, rule-based parsers affect from over-generation. Over-generation can increase the load on the filter module to arrive at a single parse output. However, overgeneration also proves to be beneficial for constructions like shared argument constructions, pro-drop constructions etc.

(5.12) *mā* *nāyana* *vāḍini* *tīcaru-ku* *appagiMci*
we-POSS father he-ACC teacher-DAT handover
vacc-ē-vāḍu
go-HAB-3.SG.M
'My father used to handover him to the teacher'

Like in example-(5.12), *nāyana* 'father' is the subject shared between the subordinate clause verb and the matrix verb. In such cases, rule-based parser parser marks the *kartā* onto both verbs.

(5.13) *tellāri* *lēci* *illū* *vākili* *ūḍvāli*
early morning wakeup house sweep-HORT
'One should wake up early morning and clean the house'

In the example-(5.13), the subject is absent and the verb do not carry any agreement. In such cases, rule-based parser provides all possible relation which can be further filtered out. Consider the figure-(5.10):

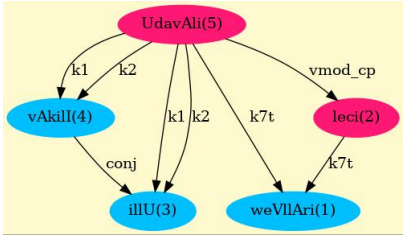


Figure 5.10: Over-generation-1

In the above figure, the relation of `k7t` is marked onto both verbs. This is an example of over-generation. Secondly, when the sentence is ellipsed for its subject, both `k1` & `k2` are marked.

5.6 Confusion Matrix - A Discussion

Confusion matrix is created in order to identify the frequently wrongly marked relations. The following observations were made:

- The *kartā* (`k1`) relation is wrongly marked as `k1s` and `k2` for about 4.6 % and 8.3% respectively. The probable reason for the failure is missing database or filtering of relations.

	k1	k1s	k2	k2g	k2p	k3	k4	k4a	k5	k5prk	k7	k7t	k7p	r6	rt	rh	ras	adv	pof	nmod
k1		4.6	8.2																	
k1s	15.5																			
k2	7.94			4.56																
k2g	9.21																			
k2p											6.57									
k3															16.21					
k4								2.08												
k4a														3.96	2.97					
k5																				
k5prk										6.25										
k7																				
k7t	5.04																			
k7p	4.94																			
r6	4.2																			1.4
rt								4.27								11.1				
rh								5.13							5.13					
ras						38.9														
adv																	4.35			
pof	8.64		9.87																	
nmod	19.04																			

Figure 5.11: Confusion Matrix-2

- The *kartā samānadhikarana* (`k1s`) relation is incorrectly marked as *kartā* (`k1`) 15.5% of the time. However, in most cases, both `k1` & `k1s` are marked. Later, using filtering module, we prune-out the incorrect relation. But the filter module could have failed in some instances leading to incorrect relations.

- *karma* (`k2`) relation is incorrectly marked as *kartā* for 7.94% and 4.56% as *gauna karma* (`k2g`). This because of the fact that *karma* can also be null-marked and the parser can easily mark incorrect relations when the database does not complement the rules.

- The most commonly mismatched relations include `k7t`, `k7p`, `r6`, `nmod` & `pof` marked as `k1` for 5.04%, 4.94%, 4.3%, 8.64% and 19.04% respectively. These wrong relations contribute to the decrease in precision.

Here, we will present some mismatched graphs in the test data. We present issues with (`k2` vs`k1`), (`k2p` vs `k1`), (`k7t` vs `k1`), (`k7p` vs `k7`), (`k1` vs `k1s`), (`pof` vs `k1`).

5.6.1 Dependency relation `k2` wrongly marked as `k1`

It is observed that in many cases, `k2` is marked as `k1` in the test data. This happens due to reasons like missing morphological cues, database issues or pro-drop cases.

- **Missing morphological Information**
  In example-(5.14), *pāṭalu* is marked with both `k1` & `k2` tags. The verb *pāḍu*

mandatorily requires a noun[+animate], however, animacy information is missing on the noun *pāṭalu* 'songs' which made the parser provide both the analysis.

(5.14) *picci* *pāṭalu* *pāḍu-kuM-ṭā-nu*
mad songs sing-REF-HAB-1.SG
'I sing crazy songs'

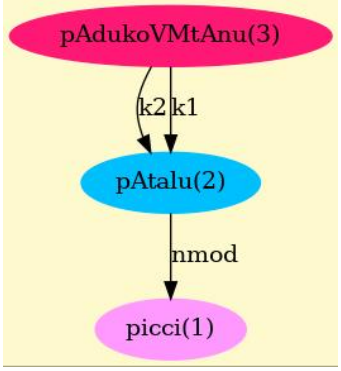


Figure 5.12: k2 vs k1

- **Pro-drop**

  Pro-drop cases are difficult for any parser to come up with unambiguous interpretation. Especially when the finite verb lacks agreement with the subject. In ex-(5.15), finite verb is in the hortative mood and does not show any agreement with the subject and the subject is dropped as well. Hence, the rule-based parser provides both the interpretations of `k1` & `k2`

(5.15) *annaM* *tinip-iMc-ā-li*
food-null eat-CAUS-PST-1.SG
'I made him eat food'

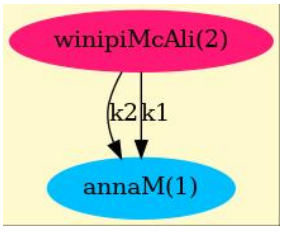


Figure 5.13: k2 vs k1

(5.16) *vāḍiki* *kotta* *guḍḍalu* *kuṭṭ-iMc-ā-ru*
he.DAT new clothes stitch-CAU-PST-3.PL
'They got him new clothes stitched'

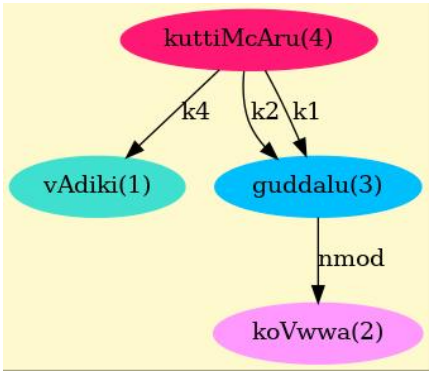


Figure 5.14: k2 vs k1

Example-(5.16) also has a pro-drop and is interpreted with both relations which can be pruned out later.

5.6.2 Dependency relation `k2p` wrongly marked as `k1`

Another case of case misinterpretation occurred with `k2p` & `k1` in the test data. This primarily happens because destination can also be null-marked in Telugu. Secondly, the information of place does not exist in the database of noun[+place]. Hence, the following error occurs in the test data.

(5.17) *paigā* *mā* *iMṭlō* *aMdarU* *polaM* *pōvāli*
moreover our house-LOC everyone field go-HORT
'Moreover, everyne should go to the fields in our house'

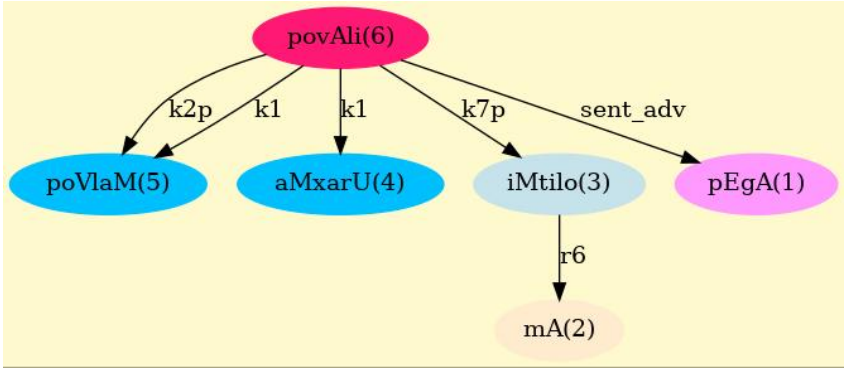


Figure 5.15: k2p vs k1

polaM in ex-(5.17) is null-marked and is apparently absent from the list of noun[+place]. Hence, it is marked as `k1`.

5.6.3 Dependency relation `k7t` wrongly marked as `k1`

The output of the test data also includes mismatches between `k7t` and `k1`. This is also because of the fact that time nouns can also be null-marked sometimes and their absence in the respective lists of database can result in wrong relation marking.

(5.18) *appuḍē* *nēnu* *baruv-aipōy-ā-nu*
already I burden-become-PST-1.SG
'I became a burden already'

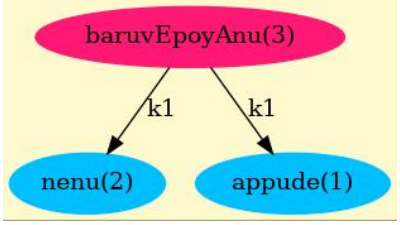


Figure 5.16: k2 vs k1

In ex-(5.18), *appuḍē* 'then only(lit.)' is a time expression which do not contain any overt case-marker. However, it must be missing in the list of time expressions which resulted in the wrong relation marking.

5.6.4 Dependency relation `k7p` wrongly marked as `k7`

Other relation that is often confused is `k7p` and `k7`. It has been discussed in earlier chapters as well that `k7` is used when the location is not a concrete location and refers to an abstract elsewhere location. However, in the following example, though *baḍilō* 'school' is a location/place, it is marked with `k7`.

(5.19) *rātri* *baḍula-lō* *āḍavāḷḷu* *viparītaM-gā* *cēr-ā-ru*
night school-LOC women extreme-ADV join-PST-3.PL
'Women joined night schools in large numbers'

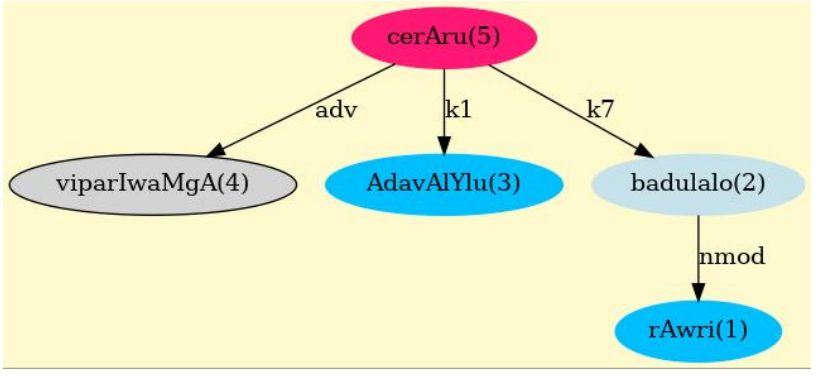


Figure 5.17: k7p vs k7

5.6.5 Dependency relation `k1` wrongly Marked as `k1s`

Subject complement is also null-marked in certain constructions in Telugu. Then, the parser marks it with both the relations viz., `k1` and `k1s`. Consider (5.20) and its respective parsed dependency tree:

(5.20)

nā	*kadha*	*aMdari*	*āḍpillala*
I.POSS	story	everyone-POSS	girls-POSS
kadh-e			
story-EMP			

'My story is similar to the story of all other girls'

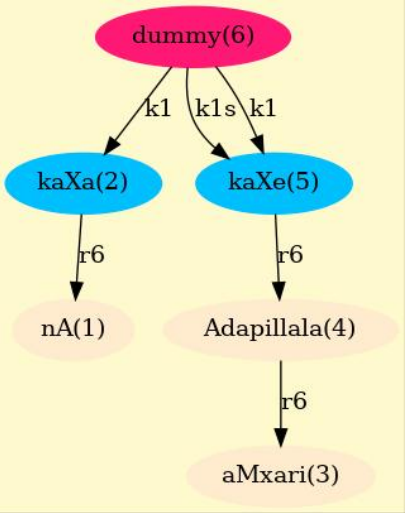


Figure 5.18: k1s vs k1

5.6.6 Dependency Relation `pof` Wrongly Marked as `k1`

pof is marked as `k1/k2` when the database does not have the respective noun-verb compound. In the figure below, *puruḍu* is marked as `k1` because *puruḍu pōsukonu* 'conceive' is absent in the database.

(5.21)

maḷḷi	*mā*	*amma*	*puruḍu*	*pōsukuMdi*
again	our	mother	concieved	

'My mother conceived again'

5.6.7 Dependency Relation `k1` Wrongly Marked as `nmod`

As discussed in error analysis before, when two nouns occur in adjacency, if the first noun does not show any difference in oblique form, system mismatches it as `k1`. Consider ((5.22)):

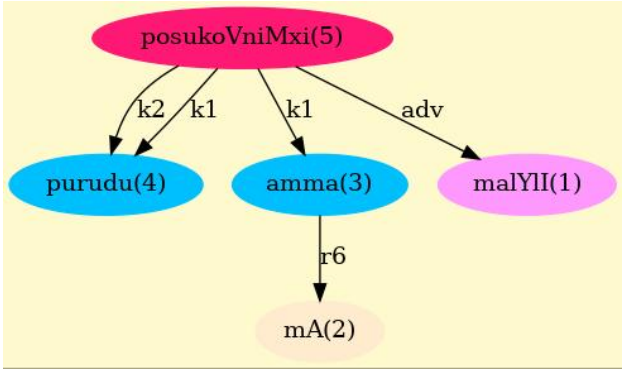


Figure 5.19: pov vs k1

(5.22) *nāku* *peḷḷi* *bhayaM* *paṭṭuk-uM-di*
I-DAT marriage fear catch-PST-3.SG.F
'I am afraid of marriage'

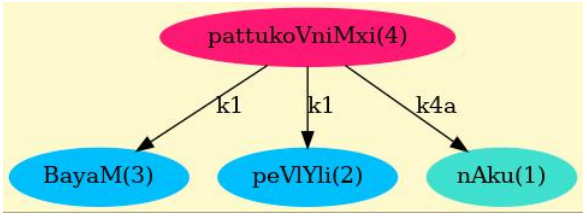


Figure 5.20: k1 vs nmod

5.6.8 Dependency Relation `ras` Wrongly Marked as `adv`

In (5.23), adverb is marked as `ras` or associative relation.

(5.23) *ā* *kōpaM-tō* *mā* *amma* *mīda* *maMḍipaḍ-ē-vāḍu*
that anger-INS our mother on fire-HAB-3.SG.M
'He shouted at my mom in anger'

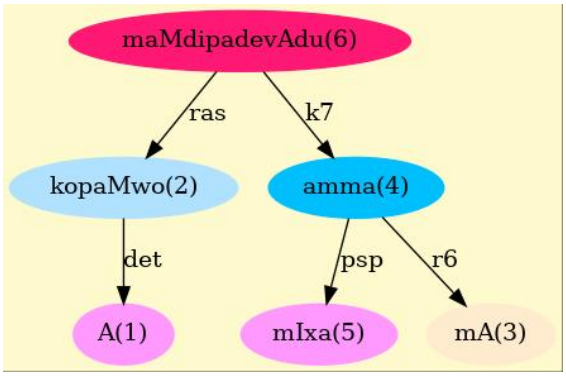


Figure 5.21: ras vs adv

5.7 Sample RBP Graphs

In this section, we present some sample parsed graphs generated by RBP from the test data. Input sentence with gloss and parsed graphs are provided.

(5.24) *vāḷḷa* *ayya-tō* *koMta* *maMdi* *baḍilō* *cērāru*
their father-ASS some people school-LOC cērāru
'Some people joined school accompained by their fathers'

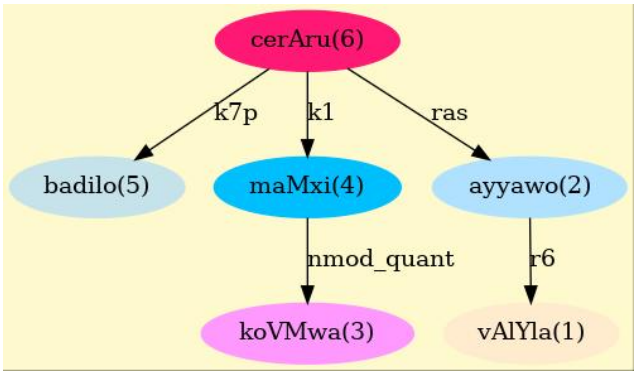


Figure 5.22: RBP generated output-1

(5.25) *naḍirātri* *iMṭiki* *vastunn-ā-ru*
midnight home-DAT come-PROG-3.PL
'They are coming home at midnight'

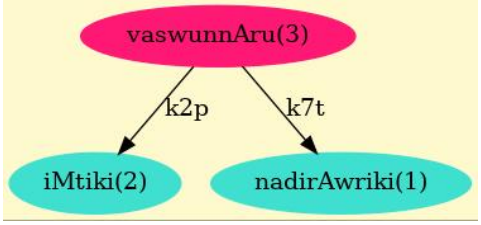


Figure 5.23: RBP generated output-2

(5.26) *nēnu* *pedda-manișini* *ayyānu*
I.NOM elder-human-AGR become-PST.1.SG
'I reached adolescence'

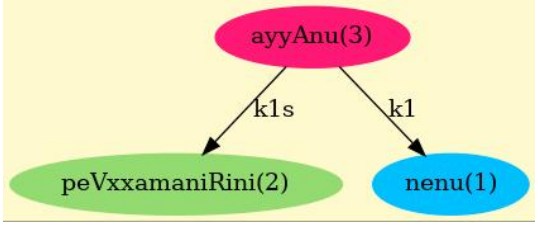


Figure 5.24: RBP generated output-3

(5.27) *mā akkala peḷḷilaki mā nānna ekarā*
our sisters-OBL marriages-DAT acre
polaM ammāḍu
land sell-PST-3.SG.M
'My father sold an acre for my sisters' marriages'

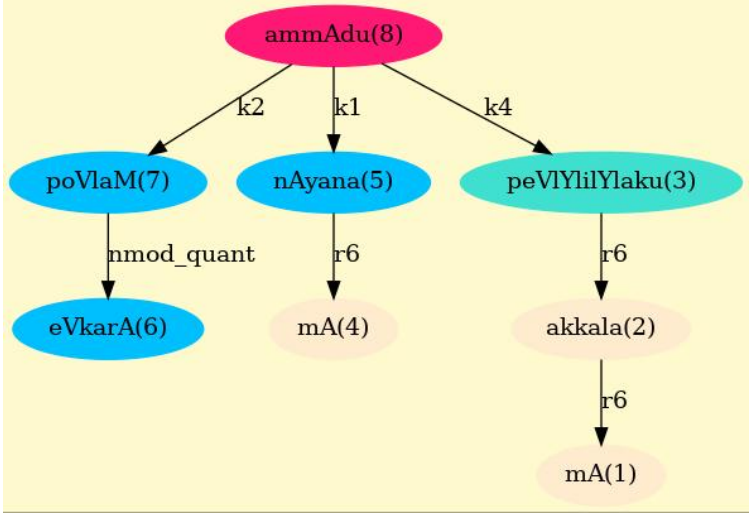


Figure 5.25: RBP generated output-4

5.8 Observations

Based on the evaluation results and error analysis, the following observation were made:

- Pre-processing errors contribute to the decrease in recall rate by 11%. Whereas issues with rules further decreases the recall by another 4%
- Secondly, database issues highly contribute to the wrong relations thereby decreasing the precision rate by 5%. Morphological analyser errors add up to the further reduction of precision by 4%.

5.8.1 Agreement and Ambiguous Relations

Certain verbs in Telugu do not show agreement with the *kartā*. Such verbs include hortative, permissive, probabilitative, etc. In such cases, when the *kartā* is pro-dropped, the system identified the *karma* which is [-animate] and zero-marked as in (5.28). The problem with certain pro-drop constructions is due to the absence of the agreement between the subject and the verb. Consider the following example:

(5.28) *cēpa-lu-∅* *tinn-occu*
fish-PL.NOM/ACC eat-CAP
'can eat fish/fish can eat'

It can be observed that the subject of the (5.28) is missing and there exists no agreement between the subject and the verb. It is highly difficult for the system to resolve such issues.

Consider another example of over-generation (5.29):

(5.29) *vacceṭappuḍu* *kaṭṭelu* *pullalu* *ērukuni* *rāvāli*
while coming sticks pick-CONJP come-HORT
'While coming one have to pick sticks'

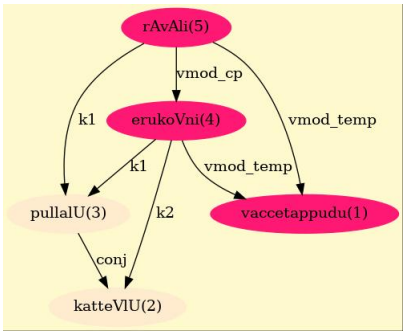


Figure 5.26: Over-generation-2

Relation	Occurrences	Precision(%)	Recall(%)
k1	628	87.1%	84.3%
k1s	50	84%	76%
k2	521	87.5%	84.6%
k2p	89	84.2%	72%
k2g	8	62.5%	62.5%
k3	43	83.8%	72.1%
k4	50	97.9%	94 %
k4a	111	93%	84.7%
k5	64	93.7%	93.7%
k5prk	28	88%	78.5%
k7	67	93.6%	88.5%
k7t	123	94.9%	91.8%
k7p	87	95 %	88.5%
k*u	33	86.2%	75.7%
r6	362	92.9%	91.1%
rt	122	84.6%	81%
jk1	8	75%	75%
pk1	7	85%	85%
rh	83	89.7%	84.3%
ras	22	61.1%	50%
psp	76	92%	90.7%
adv	117	95.6%	94.0%
nmod	108	80.9%	78.7%
nmod_wq	22	95%	86.3%
nmod_quant	78	93.5%	92.3%
nmod_adj	26	85%	65.4%
nummod	39	94.3%	84.6%
vmod	489	87.5%	78.93%
pof	89	81.5%	74.15%
pof_redup	17	81.25%	76.47%
conj	44	93.3%	63.63%
cc	15	100%	93.3%
uh	17	80%	70.5%
mark	65	96.9%	90%
rp	77	94.52%	89.6%
sent_adv	72	95.45%	87.5%
Other tags	28	89.4%	68%

Table 5.6: Relation-based Performance Index

Chapter 6

Conclusion

The present research explores the framework of dependency framework coupled with rule-based parsing as a methodology for efficient parsing of Telugu structures. A rule-based parser that serves the basic requirements of a parser viz. robustness and disambiguation is developed as part of this study. In this chapter, we provide some concluding remarks, major contributions of the study, significance of the current research and some future directions.

Firstly, as part of introduction chapter, basic theoretical understanding of parser including the terminology, theoretical & computational frameworks, types of parsing, kinds of parsers & a review of annotation schemata are presented. In addition to this, structural features of Telugu are outlined for an overview of the language under study. A brief methodology is also presented to set the tone of the thesis.

In chapter-2, the theoretical understanding of various frameworks popularly adopted to parsing are discussed in detail. Conceptual understanding of dependency and constituency framework is provided. A thorough distinction of dependency to constituency framework is attempted. In order to adopt the best-suited framework for Telugu, popularly used dependency frameworks viz Tesniere's dependency and Pāṇinian dependency are compared. Pāṇinian dependency model or the Indian grammatical tradition outweighed the advantages in comparison to other models. Hence, it is chosen as a theoretical framework for this study. It proved to be highly precise for representing Indian language analysis. Through this study, the claim that Indian grammatical theories are mechanical and enable the programming of any natural language is re-emphasized.

Furthermore, selecting an appropriate annotation schema was a challenge. For this study, the commonly used tagsets for Indian languages i.e. universal dependencies and AnnCorra tagsets are explored. Universal Dependencies was observed to be shallow when compared to Anncorra tagset with respect to certain relations. Therefore, AnnCorra tagset is chosen to represent dependency relations in Telugu. Nevertheless, AnnCorra tagset is modified to be adaptable to Telugu. Major modifications include the marking of coordinate constructions and subordinate clauses.

AnnCorra guidelines is quite established for Hindi, but for other major Indian languages, a well-defined guidelines is still a rarity. Each dependency relation pertaining to Telugu syntax is presented as part of chapter-3. Every relation in-terms of its sentence types is explicated. Around 54 relations are discussed in detail including examples and exceptional cases. Guidelines are framed, taking insights from several Telugu grammar books. Several language-specific tags introduced as part of this study, are also discussed.

Implementation of RBP follows immediately after the guidelines. There are three important differences between the data-driven parsers and the present RBP that makes it unique :

1. Treebank annotation is absent.

2. Pre-processing does not include chunking. It only includes morphological analysis and pick-one morph output. The striking difference between RBP and other data-driven approaches is its non-usability of a chunker. In data-driven parsers consisting of a chunker, the head of the chunk is usually given the tag and intra-chunk tags are given to the other words in the chunk. However, in RBP every word in the input sentence will be provided with a specific dependency tag.

3. Produces multiple interpretations/graphs of the same sentence, in-case of ambiguity. All possible analysis of a sentence are provided by RBP.

Implementation of the parser beginning from cleaning, pre-processing, parsing and post-processing phases are described. The process of disambiguation of case-markers for their semantic relations is set out. Parser rules for default kārakā relations are also sketched.

Another prominent aspects of building any computational tool is its evaluation. The parser hence built as part of this study is evaluated for its precision & recall, UAS/LAS/LA, relation-based performance index and confusion matrix. Precision and recall are found to be 90.3% and 84.1% respectively. Whereas LAS & UAS & LA is reported to be 84.1%, 90.3%, 86.2% respectively. The performance of each relation also proves the efficiency of RBP to solve the parsing problem for Telugu. However, like any other computational tool, RBP also fails to provide the correct relation for certain constructions. These cases include relations like `k2`, `k1s`, *k7t*, `r6`, `nmod` etc mismatched as `k1`. A detailed error analysis is also presented. The reasons for failure of parser in certain cases are clearly outlined along with the methods to improve the results further.

6.1 Major Contributions

- This thesis contributes to the theoretical understanding of dependency grammar and its existing frameworks in comparison to the popular constituency framework. This study re-emphasizes the importance of Indian grammatical traditions in programming Indian as well as any other natural language text. This thesis stands as a testament for the robustness of Indian dependency traditions.
- A detailed comparison of Universal Dependency tagset and AnnCorra tagset is provided in addition to guaging the advantages and disadvantages of each schemata.
- This study contributes in enriching the existing version of the AnnCorra tagset for certain relations for Telugu that can be further utilized for major Dravidian languages. An in-detail guidelines for marking dependency relations for Telugu including illustrations and exceptions are presented.
- An open source Rule-based dependency parser exploiting the morphological features of Telugu is developed.
- A rich lexical reserve for Telugu that aligns with the parser modules is created.
- The parser developed as part of this study can also be adapted to other Dravidian languages with minimal effort.

6.2 Significance of the Study

- The rule-based parser developed as part of this study provides a considerably good recall and precision rate performing on par with the data-driven parsers.
- It can be highly useful in generating treebanks and reduces the herculean task of annotating the corpus. Parsing being a challenging task requires ample time to annotate the corpus.
- Using the parse output generated by RBP, we can generate treebanks that can be further post-edited by annotators instead of building a treebank from the scratch.
- RBP provides fine-grained dependency parsing that will be quite relevant in semantic role labelling as well. RBP stands relevant in language teaching modules for Telugu syntax.

- Unlike the data-driven parsers, RBP provides several interpretations of the single sentence leading to multiple analysis. This eases the disambiguation process. For example, Tagging of Shared arguments in participial clauses. In case of shared arguments, RBP typically marks the subject twice, for both subordinate and matrix verb. Like, the shared subject will be marked `k1` on both the verbs. This will be quite useful for further NLP processing like the question-answering systems etc.

- For low-resource languages like Telugu, the availabilty of large annotated corpus is a challenge. Hence, RBP can be integrated to higher NLP and downstream applications involving Telugu for further levels of analysis.

- RBP can be easily adaptable to languages of the Dravidian language family

6.3 Some Challenges

Building an exhaustive list of lexical reserve remains as an on-going process which betters the output of the parser. As RBP heavily relies on pre-processing tools and lexical databases. The failure of which can be the biggest hindrance for RBP. As the language is ever-evolving, database requires regular updation without which RBP cannot be updated.

Another issue with RBP is the problem of over-generation. Often, multiple relations are marked which have to be pruned out in filtering stage. Hence, filtering module requires a special focus. Filtering process is also a continuous task.

Improvement of pre-processing tools stand crucial to the better performance of RBP. Failure in each pre-processing module contributes to the decrease in recall and precision.

6.4 Future Work

- This study stands as a first stage of experimenting with rule-based parsers. Further, a synthesis of the current RBP and data-driven approach can be the future direction.

- Future research involves a full-fledged implementation of RBP for all the complex sentence structures pertaining to Telugu.

- Another important future work is the inclusion of database for Multi-Word Expressions (MWE).

- Dealing with ellipses is another arena for future research. In Telugu, ellipses can be observed in pro-drop constructions, Genitive marker *yokka* is almost always absent, Conjunction *mariyu* is dropped and is expressed morphologically, Complementizer *ani* is omitted with certain verbs, the main verb is absent in copula constructions. Currently, RBP does not supply any ellided elements except for the dummy verb in this guidelines. All the other elements are usually retrieved using the morphological information. Further research can be carried on such ellided constructions.

- Integration of additional linguistic knowledge & ontological information is another crucial aspect that requires further focus.

- Converting the RBP output into other tagsets is also part of the future work. As Universal Dependency is gaining popularity, conversion of the current tags to UD tags can be highly beneficial.

- To gain better understanding of unrestricted natural language text, it is important to expand the current RBP to other Dravidian languages thereby proving to what extent the current methodology is language-independent.

- Evaluation metrics can be further explored to get a wholesome understanding of the machine. The metrics should include semantically inclined evaluation that enables predicate-argument structure. As RBP focuses on fine-grained analysis, semantics is to an extent included as part of the tags. Hence, exploring other evaluation strategies can be beneficial.

Bibliography

Aho, Alfred V. and Jeffrey D. Ullman. 1972. *The Theory of Parsing, Translation and Compiling*, volume 1. Prentice-Hall, Englewood Cliffs, NJ. 1

Aissen, Judith. 2003. Differential object marking: Iconicity vs. economy. *Natural Language & Linguistic Theory*, 21(3):435–483. 72

Ajdukiewicz, Kazimierz. 1935. Die syntaktische konnexitat. *Studia philosophica*, pages 1–27. 9

Ambati, Bharat Ram, Tejaswini Deoskar, and Mark Steedman. 2013. Using ccg categories to improve hindi dependency parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 604–609. 9

Ambati, Bharat Ram, Tejaswini Deoskar, and Mark Steedman. 2014. Improving dependency parsers using combinatory categorial grammar. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics, volume 2: Short Papers*, pages 159–163. 9

Ambati, Bharat Ram, Phani Gadde, and Karan Jindal. 2009. Experiments in indian language dependency parsing. *Proceedings of the ICON09 NLP Tools Contest: Indian Language Dependency Parsing*, pages 32–37.

American Psychological Association. 1983. *Publications Manual*. American Psychological Association, Washington, DC.

Anchiêta, Rafael Torres and Thiago Alexandre Salgueiro Pardo. 2018. A rule-based amr parser for portuguese. In *Ibero-American Conference on Artificial Intelligence*, pages 341–353. Springer. 12

Ando, Rie Kubota and Tong Zhang. 2005. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research*, 6:1817–1853.

Andrew, Galen and Jianfeng Gao. 2007. Scalable training of L1-regularized log-linear models. In *Proceedings of the 24th International Conference on Machine Learning*, pages 33–40.

Antony, PJ, Nandini J Warrier, and KP Soman. 2010. Penn treebank-based syntactic parsers for south dravidian languages using a machine learning approach. *International Journal of Computer Applications*, 7(8):14–21. 22

Bahrani, Mohammad, Hossein Sameti, and Mehdi Hafezi Manshadi. 2011. A computational grammar for persian based on gpsg. *Language Resources and Evaluation*, 45(4):387–408. 10

Baud, Robert H, Anne-Marie Rassinoux, Patrick Ruch, Christian Lovis, and Jean-Raoul Scherrer. 1999. The power and limits of a rule-based morpho-semantic parser. In *Proceedings of the AMIA symposium*, page 22. American Medical Informatics Association. 12

Bès, Gabriel G and Karine Baschung. 1985. *Feasibility of a GPSG French Grammar*. Ph.D. thesis, Université Blaise-Pascal, Clermont-Ferrand. 10

Bharati, Akshar, Vineet Chaitanya, Rajeev Sangal, and K. V. Ramakrishnamacharyulu. 1995. *Natural language processing: a Paninian perspective*. Prentice-Hall of India, New Delhi.

Bharati, Akshar, Mridul Gupta, Vineet Yadav, Karthik Gali, and Dipti Misra Sharma. 2009a. Simple parser for indian languages in a dependency framework. In *Proceedings of the Third Linguistic Annotation Workshop (LAW III)*, pages 162–165. 21

Bharati, Akshar, Samar Husain, Bharat Ambati, Sambhav Jain, Dipti Sharma, and Rajeev Sangal. 2008a. Two semantic features make all the difference in parsing accuracy. *Proc. of ICON*, 8. 21

Bharati, Akshar, Samar Husain, Bharat Ambati, Sambhav Jain, Dipti Sharma, and Rajeev Sangal. 2008b. Two semantic features make all the difference in parsing accuracy. In *Proceedings of ICON 8*. 143

Bharati, Akshar, Rajeev Sangal, and Dipti M Sharma. 2007. Ssf: Shakti standard format guide. *Language Technologies Research Centre, International Institute of Information Technology, Hyderabad, India*, pages 1–25. 134

Bharati, Akshar, Dipti Misra Sharma, Samar Husain, Lakshmi Bai, Rafiya Begum, and Rajeev Sangal. 2009b. Anncorra: Treebanks for indian languages, guidelines for annotating hindi treebank. *LTRC, IIIT Hyderabad, India*. Version 2. 51

Bharati, Akshara, Dipti Misra Sharma, Samar Husain, Lakshmi Bai, Rafiya Begam, and Rajeev Sangal. 2012. Anncorra: Treebanks for indian languages, guidelines for annotating hindi treebank. 18, 45, 54, 58, 83, 108, 118

Bhaskararao, Peri and Karumuri Venkata Subbarao. 2004. *Non-nominative subjects*, volume 1. John Benjamins Publishing. 5, 28, 151

Bhat, Riyaz Ahmad and Dipti Misra Sharma. 2012. Non-projective structures in indian language treebanks. In *Proceedings of the 11th workshop on treebanks and linguistic theories (TLT11)*, pages 25–30. Citeseer. 33

Biagetti, Erica, Salvatore Scarlata, Elia Ackermann, Oliver Hellwig, and Paul Widmer. 2020. Annotation guidelines for the vedic treebank, v. 2. *sn*, (sn). 22

Biberauer, Theresa. 2008. *The limits of syntactic variation*, volume 132. John Benjamins Publishing. 5

Bies, Ann, Mark Ferguson, Karen Katz, Robert MacIntyre, Victoria Tredinnick, Grace Kim, Mary Ann Marcinkiewicz, and Britta Schasberger. 1995. Bracketing guidelines for treebank ii style penn treebank project. *University of Pennsylvania*, 97:100. 15

Bikel, Daniel M and David Chiang. 2000. Two statistical parsing models applied to the chinese treebank. In *Second Chinese Language Processing Workshop*, pages 1–6. 20

Bossong, Georg. 1985. Differentielle objektmarkierung in den neuiranischen sprachen. *Tübingen: Gunter Narr Verlag*. 72

Brdar, Mario et al. 2003. Andrew wilson, paul rayson and tony mcenery, eds.: Corpus linguistics by the lune. a festschrift for geoffrey leech. *Jezikoslovlje*, 4(2):296–303. 3

Bresnan, Joan. 1978. A realistic transformational grammar. *Linguistic theory and psychological reality*. 10

Bresnan, Joan. 1982. Control and complementation. *Linguistic inquiry*, 13(3):343–434.

Bunt, Harry, John Carroll, and Giorgio Satta. 2005. *New developments in parsing technology*, volume 23. Springer Science & Business Media. 1

Carnie, Andrew. 2008. *Constituent structure*. OUP Oxford. 7

Chandra, Ashok K., Dexter C. Kozen, and Larry J. Stockmeyer. 1981. Alternation. *Journal of the Association for Computing Machinery*, 28(1):114–133.

Charniak, Eugene. 1997. Statistical parsing with a context-free grammar and word statistics. *AAAI/IAAI*, 2005(598-603):18. 20

Charniak, Eugene. 2000. A maximum-entropy-inspired parser. In *1st Meeting of the North American Chapter of the Association for Computational Linguistics*. 20

Chen, Danqi and Christopher D Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 740–750. 13, 14

Chinnaya Suri, Paravastu. 1855. Balavyakaranam. *Madras: Vavilla & Sons*. 28, 151

Chomsky, Noam. 1957. *Syntactic Structures*. Mouton and co.: The Hague. 34, 39

Chomsky, Noam. 2013. Topics in the theory of generative grammar. In *Topics in the Theory of Generative Grammar*. De Gruyter Mouton. 37

Clark, Alexander, Chris Fox, and Shalom Lappin. 2013. *The handbook of computational linguistics and natural language processing*. John Wiley & Sons. 6

Collins, Michael, Jan Hajic, Lance Ramshaw, and Christoph Tillmann. 1999. A statistical parser for czech. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics*, pages 505–512. 20

Comrie, Bernard. 1991. Form and function in identifying cases. *Paradigms: The economy of inflection*, pages 41–55. 5

Covington, Michael A. 2001. A fundamental algorithm for dependency parsing. In *Proceedings of the 39th annual ACM southeast conference*, volume 1. Citeseer. 44

De Marneffe, Marie-Catherine, Timothy Dozat, Natalia Silveira, Katri Haverinen, Filip Ginter, Joakim Nivre, and Christopher D Manning. 2014. Universal stanford dependencies: A cross-linguistic typology. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 4585–4592. 48

De Marneffe, Marie-Catherine and Christopher D Manning. 2008. Stanford typed dependencies manual. Technical report, Technical report, Stanford University. 16

De Marneffe, Marie-Catherine, Christopher D Manning, Joakim Nivre, and Daniel Zeman. 2021. Universal dependencies. *Computational linguistics*, 47(2):255–308. 20, 56

De Marneffe, Marie-Catherine and Joakim Nivre. 2019. Dependency grammar. *Annual Review of Linguistics*, 5:197–218.

Eisele, Andreas and Jochen Dorre. 1986. A lexical functional grammar system in prolog. In *Coling 1986 Volume 1: The 11th International Conference on Computational Linguistics*. 10

Falk, Yehuda N. 2001. *Lexical Functional Grammar*. Center for the Study of Language and Information - CSLI Lecture Notes 126. CSLI Publications. 10

Farris, Adam and Aryaman Arora. 2021. For the purpose of curry: A ud treebank for ashokan prakrit. *arXiv preprint arXiv:2111.12783*. 23

Fundel, Katrin, Robert Küffner, and Ralf Zimmer. 2007. Relex—relation extraction using dependency parse trees. *Bioinformatics*, 23(3):365–371. 20

Garapati, Umamaheshwar Rao, Rajyarama Koppaka, and Srinivas Addanki. 2012a. Dative case in telugu: a parsing perspective. In *Proceedings of the Workshop on Machine Translation and Parsing in Indian Languages*, pages 123–132. 23

Garapati, Umamaheshwar Rao, Rajyarama Koppaka, and Srinivas Addanki. 2012b. Dative case in telugu: a parsing perspective. In *Proceedings of the Workshop on Machine Translation and Parsing in Indian Languages*, pages 123–132.

Gatla, Praveen. 2019. Dependency parsing for telugu using data-driven parsers. *Language in India*, 19(1). 24, 169

Gawron, Jean Mark, Jonathan King, John Lamping, Egon Loebner, E Anne Paulson, Geoffrey K Pullum, Ivan A Sag, and Thomas Wasow. 1982. Processing english with a generalized phrase structure grammar. In *20th Annual Meeting of the Association for Computational Linguistics*, pages 74–81. 10

Gazdar, Gerald, Ewan Klein, Geoffrey K Pullum, and Ivan A Sag. 1985a. *Generalized phrase structure grammar*. Harvard University Press. 10

Gazdar, Gerald, Ewan Klein, Geoffrey K. Pullum, and Ivan A. Sag. 1985b. *Generalized phrase structure grammar*. Harvard University Press.

Ghosh, Aniruddha, A Das, P Bhaskar, and Sivaji Bandyopadhyay. 2009. Dependency parser for bengali: the ju system at icon 2009. *NLP tool contest ICON*, 2009:87–91. 22

Goldberg, Yoav and Michael Elhadad. 2010. An efficient algorithm for easy-first non-directional dependency parsing. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 742–750. 20

Goyal, P, Manav R Mital, A Mukerjee, Achla M Raina, D Sharma, P Shukla, and K Vikram. 2003. A bilingual parser for hindi, english and code-switching structures. In *10th Conference of The European Chapter*, page 15. 11

Group, XTAG Research et al. 1998. A lexicalized tree adjoining grammar for english. *arXiv preprint cs/9809024*. 11

Güngördü, Zealal and Kemal Oflazer. 1995. Parsing turkish using the lexical functional grammar formalism. *Machine Translation*, 10(4):293–319. 10

Gusfield, Dan. 1997. *Algorithms on Strings, Trees and Sequences*. Cambridge University Press, Cambridge, UK.

Hall, Johan, Jens Nilsson, and Joakim Nivre. 2010. Single malt or blended? a study in multilingual parser optimization. *Trends in Parsing Technology*, pages 19–33. 20

Haverinen, Katri, Filip Ginter, Veronika Laippala, and Tapio Salakoski. 2009. Parsing clinical finnish: Experiments with rule-based and statistical dependency parsers. In *Proceedings of the 17th Nordic Conference of Computational Linguistics (NODALIDA 2009)*, pages 65–72. 12

Hays, David G. 1964. Dependency theory: A formalism and some observations. *Language*, 40(4):511–525. 7

Hockenmaier, Julia and Mark Steedman. 2007. Ccgbank: a corpus of ccg derivations and dependency structures extracted from the penn treebank. *Computational Linguistics*, 33(3):355–396. 9

Hudson, Richard A. 1980. Constituency and dependency. 42

Hudson, Richard A. 1984. *Word grammar*. Blackwell Oxford. 30, 52, 53, 118

Husain, Samar. 2009. Dependency parsers for indian languages. In *Proceedings of ICON09 NLP Tools Contest: Indian Language Dependency Parsing*.

Hutchins, John W. 2000. Early years in machine translation. *Early Years in Machine Translation*, pages 1–411. 19

India, POMPI. 2011. Census of india 2011, paper1 of 2018, language. *New Delhi: Office of the Registrar General and Census Commissioner*. 4

Jain, Priyanka, Ram Bhavsar, Ajai Kumar, BV Pawar, Hemant Darbari, and Virendrakumar C Bhavsar. 2018. Tree adjoining grammar based parser for a hindi text-to-scene conversion system. In *2018 3rd International Conference for Convergence in Technology (I2CT)*, pages 1–7. IEEE. 11

Jarvinen, Timo and Pasi Tapanainen. 1998. Towards an implementable dependency grammar. *arXiv preprint cmp-lg/9809001*. 43

Jeuring, Johan T and S Doaitse Swierstra. 2001. Grammars and parsing. 1

Joshi, Aravind K, Leon S Levy, and Masako Takahashi. 1975. Tree adjunct grammars. *Journal of computer and system sciences*, 10(1):136–163. 11

Joshi, Nisheeth and Iti Mathur. 2012. Evaluation of computational grammar formalisms for indian languages. *CoRR*, abs/1209.1301. 10

Jurafsky, Dan. 2000. *Speech & language processing*. Pearson Education India. 7, 13, 14, 165

Kanneganti, Silpa, Himani Chaudhry, and Dipti Misra Sharma. 2016. Comparative error analysis of parser outputs on telugu dependency treebank. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 397–408. Springer. 24

Kaplan, Ronald M. 1972. Augmented transition networks as psychological models of sentence comprehension. *Artificial Intelligence*, 3:77–100. 10

Kaplan, Ronald M and Joan Bresnan. 1982. Lexical-functional grammar: A formal system for grammatical representation. *The Mental Representation of Grammatical Relations*, pages 173–281. 10

Kesidi, Sruthilaya Reddy, Prudhvi Kosaraju, Meher Vijay, and Samar Husain. 2011. A constraint based hybrid dependency parser for telugu. *International Journal of Computational Linguistics and Applications*, 2(1-2):53.

Kesidi, Sruthilaya Reddy, Prudhvi Kosaraju, Meher Vijay, and Samar Husain. 2013. *Constraint-based hybrid dependency Parser for Telugu*. Ph.D. thesis, Ph. D. thesis, International Institute of Information Technology Hyderabad. 23

Khan, Naira and Mumit Khan. 2006. Developing a computational grammar for bengali using the hpsg formalism. 11

Kim, Jeong-Ryeol. 1993. Parsing light verb constructions in lexical-functional grammar. . 10

Kiparsky, Paul. 2007. On the architecture of pāṇini's grammar. In *Sanskrit computational linguistics*, pages 33–94. Springer. 44

Klein, Dan and Christopher D Manning. 2002. Fast exact inference with a factored model for natural language parsing. *Advances in neural information processing systems*, 15. 20

Krishnamurthy, Parameswari and Kengatharaiyer Sarveswaran. 2021. Towards building a modern written tamil treebank. In *Proceedings of the 20th International Workshop on Treebanks and Linguistic Theories (TLT, SyntaxFest 2021)*, pages 61–68. 22

Krishnamurti, Bhadriraju. 2003a. *The dravidian languages*. Cambridge University Press. 4

Krishnamurti, Bhadriraju. 2003b. *The dravidian languages*. Cambridge University Press. 28, 151

Krishnamurti, Bhadriraju and John Peter Lucius Gwynn. 1985. *A grammar of modern Telugu*. Oxford University Press, USA. 28, 47, 58, 65, 68, 105, 115, 151

Kroch, Anthony S and Aravind K Joshi. 1985. The linguistic relevance of tree adjoining grammar. *Technical Reports (CIS)*, page 671. 11

Kulkarni, A. 2021a. *Sanskrit Parsing: Based on the Theories of Śābdabodha*. D.K. Printworld. 1, 8, 19, 38, 137

Kulkarni, Amba. 2016. Samsaadhanii: A sanskrit computational toolkit. 18

Kulkarni, Amba. 2021b. *Sanskrit Parsing: Based on the Theories of Śābdabodha*. DK Printworld (P) Ltd. 44, 74, 142

Kulkarni, Amba. April 2021. Sanskrit parsing following indian theories of verbal cognition. *ACM Transactions on Asian and Low-Resource Language Information Processing*, 20:1–38. 129

Kulkarni, Amba and Dipti Misra Sharma. 2019. Pāṇinian syntactico-semantic relation labels. *Depling 2019*, page 198. 58, 61, 86

Kulkarni, Amba, Sanal Vikram, and K Sriram. 2019. Dependency parser for sanskrit verses. In *Proceedings of the 6th International Sanskrit Computational Linguistics Symposium*, pages 14–27.

Kumari, B Venkata Seshu and Ramisetty Rajeshwara Rao. 2015. Improving telugu dependency parsing using combinatory categorial grammar supertags. *ACM Transactions on Asian and Low-Resource Language Information Processing (TALLIP)*, 14(1):1–10. 23

Kumari, B Venkata Seshu and Ramisetty Rajeshwara Rao. 2017. Telugu dependency parsing using different statistical parsers. *Journal of King Saud University-Computer and Information Sciences*, 29(1):134–140. 24

Levin, Beth. 1993. *English verb classes and alternations: A preliminary investigation*. University of Chicago press. 74

Levin, Beth. 2015. 39. verb classes within and across languages. In *Volume 2 Case Studies from Austronesia, the Pacific, the Americas, and Theoretical Outlook*, pages 1627–1670. De Gruyter Mouton. 148

Levine, Robert D and Walt Detmar Meurers. 2006. Head-driven phrase structure grammar: Linguistic approach, formal foundations, and computational realization. *Encyclopedia of language and linguistics*, 2. 11

Lidz, Jeffrey. 2006. The grammar of accusative case in kannada. *Language*, pages 10–32. 72

Liu, Haitao and Wei Huang. 2006. A chinese dependency syntax for treebanking. In *Proceedings of the 20th Pacific Asia Conference on Language, Information and Computation*, pages 126–133. 17

Magerman, David M and Mitchell P Marcus. 1994. Pearl: A probabilistic chart parser. *arXiv preprint cmp-lg/9405005*. 20

de Marneffe, Marie-Catherine and Joakim Nivre. 2019. Dependency grammar. *Annual Review of Linguistics*, 5(1):197–218. 32, 33, 36

McCulloch, Warren S and Walter Pitts. 1943. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133. 13

McDonald, Ryan, Fernando Pereira, Kiril Ribarov, and Jan Hajic. 2005. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of human language technology conference and conference on empirical methods in natural language processing*, pages 523–530. 20

Mel'cuk, Igor Aleksandrovic et al. 1988. *Dependency syntax: theory and practice*. SUNY press. 7, 8, 30, 52, 53, 118

Menon, Vijay Krishna, S Rajendran, M Anand Kumar, and KP Soman. 2016. A new tag formalism for tamil and parser analytics. *arXiv preprint arXiv:1604.01235*. 11

Muralidaran, Vigneshwaran and Dipti Misra Sharma. 2016. Construction grammar based annotation framework for parsing tamil. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 378–396. Springer. 21

Murthy, K Narayana. 1996. Parsing telugu in the ucsg formalism. In *Proc. Indian Congress on Knowledge and Language*, volume 2, pages 1–16.

Nagaraju, G, N Mangathayaru, and B Padmaja Rani. 2016. Dependency parser for telugu language. In *Proceedings of the Second International Conference on Information and Communication Technology for Competitive Strategies*, pages 1–5. 24

Nallani, Sneha, Manish Shrivastava, and Dipti Sharma. 2020a. A fully expanded dependency treebank for Telugu. In *Proceedings of the WILDRE5– 5th Workshop on Indian Language Data: Resources and Evaluation*, pages 39–44, Marseille, France. European Language Resources Association (ELRA). 104

Nallani, Sneha, Manish Shrivastava, and Dipti Misra Sharma. 2020b. A fully expanded dependency treebank for telugu. In *Proceedings of the WILDRE5–5th Workshop on Indian Language Data: Resources and Evaluation*, pages 39–44. 24

Nallani, Sneha, Manish Shrivastava, and Dipti Misra Sharma. 2020c. A simple and effective dependency parser for telugu. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, pages 143–149. 24, 46, 169

Nivre, Joakim. 2006. *Inductive dependency parsing*. Springer. 1, 7, 12, 13, 31

Nivre, Joakim. 2009. Parsing indian languages with maltparser. *Proceedings of the ICON09 NLP Tools Contest: Indian Language Dependency Parsing*, pages 12–18. 21, 165

Nivre, Joakim, Željko Agić, Lars Ahrenberg, Lene Antonsen, Maria Jesus Aranzabe, Masayuki Asahara, Luma Ateyah, Mohammed Attia, Aitziber Atutxa, Liesbeth Augustinus, et al. 2017. Universal dependencies 2.1.

Nivre, Joakim, Marie-Catherine De Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, et al. 2016. Universal dependencies v1: A multilingual treebank collection. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 1659–1666.

Nivre, Joakim, Johan Hall, and Jens Nilsson. 2006. Maltparser: A data-driven parser-generator for dependency parsing. In *LREC*, volume 6, pages 2216–2219. 20

Nivre, Joakim and Mario Scholz. 2004. Deterministic dependency parsing of english text. In *COLING 2004: Proceedings of the 20th International Conference on Computational Linguistics*, pages 64–70.

Ojha, Atul Kr and Daniel Zeman. 2020. Universal dependency treebanks for low-resource indian languages: The case of bhojpuri. In *Proceedings of the WILDRE5–5th workshop on Indian language data: resources and evaluation*, pages 33–38. 22

Osborne, Timothy. 2013. A look at tesnière's éléments through the lens of modern syntactic theory. In *Proceedings of the Second International Conference on Dependency Linguistics (DepLing 2013)*, pages 262–271. 7

Osborne, Timothy. 2019. *A dependency grammar of English: An introduction and beyond*. John Benjamins Publishing Company. 35, 36, 43, 44, 52

Panchal, Sanjeev and Amba Kulkarni. 2019. Co-ordination in sanskrit. *Indian Linguistics*, 80(1-2):59–176. 54

Parida, Shantipriya, Kalyanamalini Sahoo, Atul Kr Ojha, Saraswati Sahoo, Satya Ranjan Dash, and Bijayalaxmi Dash. 2022. Universal dependency treebank for odia language. *arXiv preprint arXiv:2205.11976*. 23

Petrov, Slav, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 433–440. 20

Petrov, Slav and Ryan McDonald. 2012. Overview of the 2012 shared task on parsing the web. 45

Phillips, John D. 1992. A computational representation for generalised phrase-structure grammars. *Linguistics and Philosophy*, 15(3):255–287. 10

Pollard, C. and I. A. Sag. 1994a. *Head-Driven Phrase Structure Grammar*. University of Chicago Press, US.

Pollard, Carl and Ivan Sag. 1987. Information-based syntax and semantics, vol. 1, csli. 11

Pollard, Carl and Ivan A Sag. 1994b. *Head-driven phrase structure grammar*. University of Chicago Press. 11

Preeti K, Patel. 2010. *Vibhakti Divergence between Sanskrit and Hindi*. Unpublished MPhil. dissertation, University of Hyderabad,Hyderabad. 139

Raj, Mohit, Shyam Ratan, Deepak Alok, Ritesh Kumar, and Atul Kr Ojha. 2022. Developing universal dependency treebanks for magahi and braj. *arXiv preprint arXiv:2204.12633*. 23

Rama, Taraka and Sowmya Vajjala. 2018. A dependency treebank for telugu. In *English Conference Papers, Posters and Proceedings*, 8, pages 119–128. 22, 24, 169

Ramarao, C. 1975. *Telugu vākhyam*. A.P. Sahitya Academy. 28, 58, 68, 69, 151

Ramarao, Chekuri. 2017. *A reference grammar of modern Telugu*. EMESCO books Pvt. ltd., Hyderabad. 4, 28, 62, 65, 73, 88, 93, 116, 151

Ramasamy, Loganathan. 2012. Prague dependency style treebank for tamil. 22

Ramasamy, Loganathan and Zdeněk Žabokrtskỳ. 2011. Tamil dependency parsing: results using rule based and corpus based approaches. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 82–95. Springer. 12

Rao, G. Uma Maheshwar. 1999. *A Morphological Analyzer for Telugu*. (electronic form). Hyderabad: University of Hyderabad. Accessible at.

Rasooli, Mohammad Sadegh and Joel R. Tetreault. 2015. Yara parser: A fast and accurate dependency parser. *Computing Research Repository*, arXiv:1503.06733. Version 2.

Ravishankar, Vinit. 2017. A universal dependencies treebank for marathi. In *Proceedings of the 16th international workshop on treebanks and linguistic theories*, pages 190–200. 22

Riezler, Stefan, Tracy Holloway King, Ronald M Kaplan, Richard Crouch, John T Maxwell III, and Mark Johnson. 2002. Parsing the wall street journal using a lexical-functional grammar and discriminative estimation techniques. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 271–278. 10

Rosta, Andrew, Kensei Sugayama, and Richard Hudson. 2005. Structural and distributional heads. *Word Grammar: New perspectives on a theory of language structure*, pages 171–203. 52, 54

Salloum, Said A, Mostafa Al-Emran, and Khaled Shaalan. 2016. A survey of lexical functional grammar in the arabic context. *International Journal of Computing and Network Technology*, 4(03). 10

Sangal, rajeev, Vineet Chaitanya, and Akshar Bharati. 1995. *Natural language processing: a Paninian perspective*. PHI Learning Pvt. Ltd. 44

Sangeetha, P, Parameswari K., and Amba Kulkarni. 2021. A rule-based dependency parser for telugu: An experiment with simple sentences. *Translation Today*, 15(1).

Selvam, M, AM Natarajan, and R Thangarajan. 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*, 2:4. 21

Sgall, Petr and Eva Hajičová. 1971. A „functional" generative description: background and framework. 30

Shieber, Stuart, Susan U Stucky, Hans Uszkoreit, and Jane J Robinson. 1983. Formal constraints on metarules. In *Proceedings of the 21st Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics. 10

Steedman, Mark and Jason Baldridge. 2011. Combinatory categorial grammar. *Non-Transformational Syntax: Formal and Explicit Models of Grammar. Wiley-Blackwell*, pages 181–224. 9

Subbārāo, Kārumūri V. 2012. *South Asian languages: A syntactic typology*. Cambridge University Press. 49, 58, 72, 82

Subbarao, Karumuri Venkata and B Lalitha Murthy. 2000. Lexical anaphors and pronouns in telugu. *Lust et al (ed.), Lexical Anaphors and Pronouns in Selected South Asian Languages: A Principled Typology*, pages 217–276. 76

Sureka, K, KG Srinivasagan, and S Suganthi. 2014. An efficiency dependency parser using hybrid approach for tamil language. *arXiv preprint arXiv:1403.6381*. 21

Suryachandra, Palli and P Venkata Subba Reddy. 2016. Statistical approaches in parsing for telugu language. In *2016 International Conference on Communication and Electronics Systems (ICCES)*, pages 1–5. IEEE.

Taylor, Ann, Mitchell Marcus, and Beatrice Santorini. 2003. The penn treebank: an overview. *Treebanks*, pages 5–22. 15

Tesnière, L. 1959. *èlèments de syntaxe structurale*, editions klincksieck edition. 7, 8, 30, 36, 43, 52, 118

Tse, Daniel and James R Curran. 2012. The challenges of parsing chinese with combinatory categorial grammar. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 295–304. 9

Uma Maheshwar Rao, Garapati. 1999. Morphological analyzer for telugu.(electronic form). *Hyderabad: University of Hyderabad*. 133

Uma Maheshwara, Rao, Amba P. Kulkarni, and M. Christopher. 2011b. A telugu morphological analyzer. *In Proceedings of International Telugu Internet Conference*. 26

Vempaty, Chaitanya, Viswanatha Naidu, Samar Husain, Ravi Kiran, Lakshmi Bai, Dipti M Sharma, and Rajeev Sangal. 2010a. Issues in analyzing telugu sentences towards building a telugu treebank. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 50–59. Springer. 23

Vempaty, Chaitanya, Viswanatha Naidu, Samar Husain, Ravi Kiran, Lakshmi Bai, Dipti M Sharma, and Rajeev Sangal. 2010b. Issues in analyzing telugu sentences towards building a telugu treebank. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 50–59. Springer.

Verma, Mahendra K and Karuvannur Puthanveettil Mohanan. 1990. *Experiencer subjects in South Asian languages*. Center for the Study of Language (CSLI). 28, 49, 152

Vikram, Sanal and Amba Kulkarni. 2020. Free word order in sanskrit and well-nestedness. In *Proceedings of the 17th International Conference on Natural Language Processing (ICON)*, pages 308–316.

Wang, Sun-Chong. 2003. Artificial neural network. In *Interdisciplinary computing in java programming*, pages 81–100. Springer. 13

Yeleti, Meher Vijay and Kalyan Deepak. 2009. Constraint based hindi dependency parsing. *ICON09 NLP Tools Contest: Indian Language Dependency Parsing. Hyderabad, India*. 22

Zeman, Daniel. 2008. Reusable tagset conversion using tagset drivers. In *LREC*, volume 2008, pages 28–30. 45

Zeman, Daniel, Martin Popel, Milan Straka, Jan Hajic, Joakim Nivre, Filip Ginter, Juhani Luotolahti, Sampo Pyysalo, Slav Petrov, Martin Potthast, et al. 2017. Conll 2017 shared task: Multilingual parsing from raw text to universal dependencies. In *CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 1–19. Association for Computational Linguistics. 22

Zhou, Junru and Hai Zhao. 2019. Head-driven phrase structure grammar parsing on penn treebank. *arXiv preprint arXiv:1907.02684*.

A Rule-based Dependency Parser for Telugu: An Experiment with Simple Sentences

SANGEETHA P., PARAMESWARI K.
& AMBA KULKARNI

Abstract

This paper is an attempt in building a rule-based dependency parser for Telugu which can parse simple sentences. This study adopts Pāṇini's Grammatical (PG) tradition i.e., the dependency model to parse sentences. A detailed description of mapping semantic relations to vibhaktis (case suffixes and postpositions) in Telugu using PG is presented. The paper describes the algorithm and the linguistic knowledge employed while developing the parser. The research further provides results, which suggest that enriching the current parser with linguistic inputs can increase the accuracy and tackle ambiguity better than existing data-driven methods.

1. Introduction

Parsing is a challenging task especially when languages under investigation are morphologically rich and have relatively free-word order. A parser is an automated Natural Language Processing (NLP) tool that analyses the input sentences based on the grammar formalism adopted in implementation and provides the output in constructed parse trees. The most frequently adopted grammar formalisms include constituency and dependency models. This study adopts the dependency model that has proved to be an efficient model for Indian languages that are morphologically rich with free-word order (Bharati & Sangal 1993; Kulkarni 2013; Kulkarni & Ramakrishnamacharyulu 2013; Kulkarni 2019).

Telugu is a South-central Dravidian language with agglutinating morphology and with relatively free word order. Hence, dependency grammar formalism was adopted for this

study which proved to be useful for other free-word order languages. Apart from grammar formalism, the technique used for the implementation of a parser also stands as equally important. The implementation techniques majorly include grammar-driven or data-driven. The present study uses a grammar-driven technique that handles a wide range of language ambiguities.

This paper discusses various problematic cases in parsing Telugu simple sentence structures which consist of a clause that includes covering constructions such as copula, imperative, passive, dubitative, interrogative, non-nominative subjects, reflexive, and coordinating noun phrases. This paper is the first attempt (to the authors' best knowledge) in building a rule-based parser for Telugu using a dependency framework.

This paper is organized as follows: Section-2 provide the literature survey of parsing in Telugu; section-3 describes the theoretical background for the study involving a discussion on the mapping from *kāraka* to *vibhakti* in Telugu, taking insights from PG; Section-4 provides a detailed description on building the current parser, algorithm, and constraints (both local and global); Section-5 provides the evaluation of the rule-based parser and Knowledge-based parser, further discussing the error analysis and some observations; finally, Section-6 concludes and explores the future scope of the study.

2. Brief Survey

A few attempts were made in developing a Telugu dependency parser based on data-driven approaches. Some of them include Vempaty Chaitanya, Viswanatha Naidu, Samar Husain, Ravi Kiran, Lakshmi Bai, Dipti Mishra Sharma & Rajeev Sangal (2010) who discussed issues in parsing various linguistic constructions like copula, genitive, implicit and explicit conjunct, and complementizer constructions. Garapati, Uma Maheshwar Rao, Rajyarama Koppaka & Srinivas Addanki

(2012) analysed dative case marker (*-ki)* with various functions in Telugu in parsing perspective. Kesidi, Sruthilaya Reddy, Prudhvi Kosaraju, Meher Vijay & Samar Husain (2013) implemented a constraint-based dependency parser for Telugu which was earlier used for languages like Hindi. This parser deals with relations in two different stages wherein stage-1 handles intra-clausal relations and stage-2 handles inter-clausal relations. Kumari, B. V. S., & Ramisetty Rajeshwara Rao (2015) had developed combinatory categorial grammar supertags using which they claim the enhancement of identification of verbal arguments. Nagaraju, B, N. Mangathayaru & B. Padmaja Rani 2016), Kumari B. V. S. & Ramisetty Rajeshwara Rao 2017, Kanneganti S., Himani Chaudhry & Dipti Misra Sharma (2018) worked on various statistical approaches of parsers. Rama, Taraka & Sowmya, Vajjala (2018) developed a Telugu treebank using Universal Dependency (UD) tagset with an addition of language-specific tags to handle compound and conjunct verb phrases for Telugu. Gatla (2019) developed a treebank for Telugu which was trained using data-driven parsers, namely, Minimum-Spanning Tree (MST) parser and Models and Algorithms for Language Technology (MALT) parser. Nallani, Sneha, Manish Shrivastava & Dipti Mishra Sharma (2020) expanded treebank by adding language-specific intra-chunk tags to the existing annotation guidelines based on the Pāṇinian framework. In addition to improving the existing tagset, Nallani, Sneha, Manish Shrivastava & Dipti Mishra Sharma (2020b), also developed a Telugu parser using a minimal feature Bidirectional Encoder Representations from Transformers (BERT) model providing considerable results. The highest Label Attachment Score (LAS) reported so far has been 93.7% (Nallani, Sneha, Manish Shrivastava & Dipti Mishra Sharma 2020) and the approaches have been data-driven. However, the results of the above-mentioned systems prove that there

should be continuous improvement in the annotated corpus size to improve the results further in data-driven approaches. Hence, the effort in building the parser for Telugu using grammar-driven approaches is attempted in this paper to study its feasibility and advantages.

3. Theoretical Background

The dependency model follows the grammatical tradition of dependency, tracing back to Pāṇini`s grammar. The dependency grammatical model represents the relation between the head and its dependents through directed arcs and arc labels. The relation between content words is marked by dependency relations; functional words are attached to the content words they modify. The parse thus generated is a tree, where the nodes of the parse tree stand for words in an utterance and the link between words represents the relation between pairs of words. All such dependencies in a sentence can either be argument dependencies (subject, object, indirect object, etc.) or modifier dependencies (determiner, noun modifier, verb modifier, etc.). The peculiar feature of the dependency model is to provide syntactico-semantic relations, unlike the other grammar formalisms, which are purely syntactic (Bresnan 1982; Gazdar Gerald, Ewan Klein, Geoffrey k. Pullum, & Ivan A. Sag, 1985). Based on these syntactico-semantic relations, Bharati Akshar, Dipti Misra Sharma, Samar Husain, Lakshmi Bai, Rafiya Begum & Rajeev Sangal (2009) have developed a dependency tagset known as Anncora tagset which can be used for almost all major Indian languages. This tagset consists of around 19 fine-grained tags for *karaka* (K) relations and 25 fine-grained tags for non-*kāraka* (r) relations. This study adopts the Anncora tagset in order to label dependency relations.

The most common dependency relation in a simple sentence structure includes the dependency between a noun and a verb

or a noun and a noun. PG uses syntactico-semantic relations called *kāraka* relations expressed through *vibhakti*s to capture dependencies between noun-verb and non-*kāraka* relations to capture noun-noun dependencies. The pāṇinian treatment of *kāraka* relations considers a system of default *vibhakti* for each relation. This *vibhakti* assignment is independent of verb semantics. Table-1 provides the default *vibhakti* for *kāraka* relations in Telugu. In addition to this, the other tags used for the current parser are listed as part of the Appendix.

Sl.No	***kāraka* Relation**	***Vibhakti***
1	*kartā* (k1)	-0
2	*karma* (k2)	*-ni/-nu*
3	*karaṇa* (k3)	*-tō*
4	*sampradāna* (k4)	*-ki/ku*
5	*apādāna* (k5)	*nuMḍi/nuMci/niMci*
6	*ṣaṣṭhī* (r6)	*Yokka*
7	*viṣaya-adhikaraṇam (k7)*	*-lō*

Table-1: *kāraka* relations and default *vibhakti*s in Telugu

Apart from these default *vibhakti*s, there exist cases of deviation in Telugu in which there is no one-one mapping between the *vibhakti* and *kāraka* relation. These deviations arise when the verbs do not follow linguistic generalizations or when a structure is out of the scope of linguistic generalisation. In order to handle these deviations, Panini employs a model wherein he proposes two methods (Preeti 2010) viz.

1. Assigning a different *vibhakti*
2. Imposing a new *kāraka* relation

Preeti (2010) summarizes the ways of mapping semantic relations to *vibhakti*s through *kāraka*s in PG. Consider the following figure:

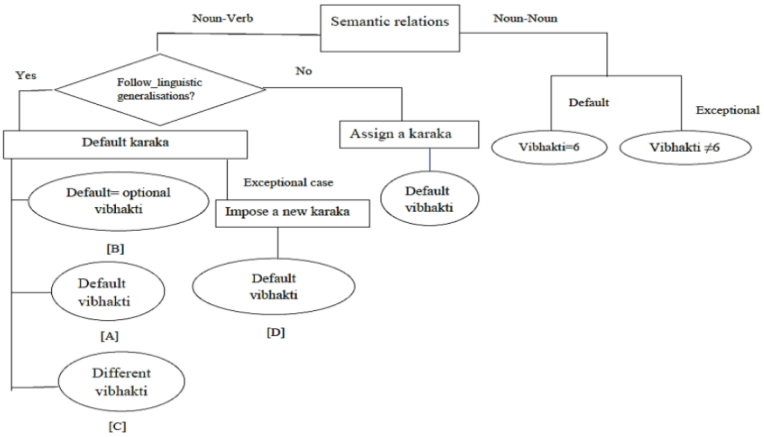


Figure-1 Semantic Relations

Based on fig-1, the semantic relations between noun-verb are divided into the following types:

3.1. Type-A

The first type of semantic relation is when the language follows the linguistic generalisation and takes a default *kāraka* as listed in Table-1. In example (1) as explicated, *kartā* (k1) and *karma* (k2) are marked with the default *vibhakti* i.e ∅ and *-ni* respectively.

1.	*nēnu.*∅	*ravi-ni*	*cūs-ā-nu.*
	I.NOM	Ravi-ACC	do-PST-1.SG.
	‘I saw Ravi’		

3.2. Type-B

In certain relations, there exist instances of verbs in addition to the default case marking which deviate from the default case marking and assign optionally other case-suffixes as in (2) and (3). The verb *ceppu* ‘to tell’ assigns either *vibhakti -ki* or *–tō* to

express the relation *sampradāna* (k4) i.e the recipient of an action as in (2).

2.	*nēnu.∅*	*prakās-**ki / -tō***	*ā viSayaM*	*cepp-ā-nu*
	I.NOM	Prakash-DAT/ASS	that matter	tell-PST-1.SG
	'I told that matter to Prakash'			

Similarly, the verb *ekku* 'to climb' in Telugu, has an expectancy of a noun expressing the location 'to climb'. In this case, the noun is marked either with the *vibhakti -nu* or *mīda* as in (3).

3.	*nēnu*	*ēnugu-*	***nu / mīda***	*ekk-ā-nu.*
	I.NOM	elephant-	ACC/on	climb up-PST-1.SG.
	'I climbed an elephant'			

3.3. Type-C

In certain cases, it is found that a different *vibhakti* is assigned instead of the default one to indicate a particular semantic relation. For instance, the default *vibhakti* indicates the source of separation, *apādānā* i.e. the ablative case as in example (4). However, in the case of mental separation as in (5) where the *kartā,vāḍu* 'he' separates himself mentally due to the fear of *siMhaM* 'lion' which is considered as *apadānā* in PG but it is realized by the different *vibhakti* i.e. *-ki*, not by *–nuMḍi*

4.	*Cettu*	*nuMḍi*	*ākulu*	*rālā-yi*
	Tree	From	Leaves	fall-3.PL
	'Leaves fell from the tree'			

5.	*vāḍu*	*siMhāni-ki*	*bhayapaḍatā-ḍu*
	He	lion-	scare-3.SG.M
	'He is scared of a lion'		

3.4. Type-D

In certain exceptional cases, it is found that a new *kāraka* is imposed using a default *vibhakti.* This can be due to the extension of the case relation as explicated in (6) where *iḷḷu* 'home' is the *karma* to the verb *veḷḷu* as per PG, however it is marked with the *vibhakti -ki.*

6.	*nēnu*	*iMti-ki*	*veḷḷ-ā-nu*
	I.NOM	house-DAT	go-PST-1.SG.
	'I went home'		

The other case as shown in Figure-1 is when the sentence does not follow linguistic generalizations and a new *kāraka* is assigned. We have not come across such cases so far in Telugu; hence no explanation is provided in this paper.

When the semantic relationship is found between noun-noun, non-*kāraka* relation i.e. *ṣaṣṭhī* (the tag 'r6') is expressed by *yokka* or the default oblique marker or by the *vibhakti-ki* in certain cases in Telugu as in (7) i.e *vādi-**ki** 'his'*.

7.	*vādi-**ki***	*kāli-ki*	*debba*	*tagil-iM-di*
	He-**DAT**	Leg-DAT	Wound-NOM	Hit-PST-3.SG.N
	'He got a wound on his leg"			

4. Parser and Algorithm

The parser takes input from sentences that are morphologically analysed and Parts of Speech (POS) tagged. Telugu morphological analyzer and POS tagger (Garapati 1999) are used as pre-processing tools. POS tagger helps in selecting the best possible morphological analysis of each word. The parser is built following the Indian theories of verbal cognition where three factors viz. *ākānksā* (expectancy), *yōgyatā* (meaning compatibility), and *sannidhi* (proximity) are used. We model the parser as a tree where the nodes of a tree correspond to a

word and the edges between nodes correspond to a relation between the corresponding words. For instance, the parsed tree of the example (1) is provided as below:

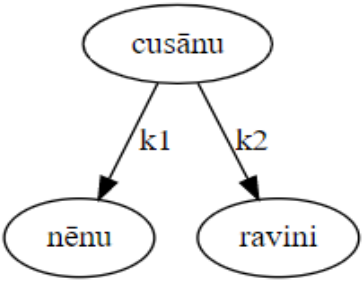


Fig-2 Parsed tree for example (1)

The basic algorithm for parsing which is followed is given below (Kulkarni 2019)

1. Define one node each corresponding to every word in a sentence
2. Establish directed edges between the nodes, if there is either a mutual or unilateral expectancy (*ākānksā*) between the corresponding words. In order to hypothesize a possible edge between two words, we refer to the expectancies of the verbs and the corresponding *vibhakti*s and then postulate a possible relation
3. Define constraints, both local on each node as well as global on the graph as a whole. One of these constraints corresponds to *sannidhi* (Proximity)
4. Use semantic constraints to filter out the meaning-wise non-congruent solutions
5. Extract all possible trees from this graph that satisfy both local and global constraints

6. Produce the most probable solution as the first solution by defining an appropriate cost function. The cost C associated with a solution tree is defined as $\boldsymbol{C = \sum_e d_e \times r_k}$ an edge from a word w_j to a word w_i with label k, $\boldsymbol{d_e = |j-i|}$, r_k rank of the role with label k. Then the problem of parsing a sentence may be modelled as the task of finding a sub-graph T of G such that T is a Directed Tree (or a Directed Acyclic Graph).

4.1 Algorithm: An Elaboration

In this section, we explain steps 2, 3, and 4 of the algorithms in detail. The step-2 corresponds to the use of lexical semantics of nouns and verbs, step-3 is the use of constraints, and step-4 is the use of selectional restriction or mutual congruity.

The **step-2** of the algorithm deals with the expectancies of verbs and the corresponding *vibhakti*s which enable the parser to postulate a possible relation. We notice that the mapping of semantic relations to *vibhakti*s is one-one except for the optional case marking (see Section 2.2), however the reverse mapping viz. *vibhakti* to semantic relation is not one-one. Case-suffixes as small as 7 (see table-1 and *ṣaṣṭhī)* in number are used to express around 40 case relations which lead to ambiguity. Ambiguities hence occurred are resolved by augmenting linguistic information such as the lexical semantics of verbs and nouns. (i) Lexical semantics of verbs. The lexical semantics of verbs provides cues in certain cases to disambiguate *vibhakti*s with their corresponding semantic relation. Consider the examples (8) & (9)

8.	*nēnu-Ø*	*vāḍi-ki*	*pustakaṁ*	*icc-ā-nu*
	I.NOM	He-DAT	Book-ACC	Give-PST-1.SG
	“I gave a book to him”.			

9.	*nēnu-Ø*	*baḍi-ki*	*veḷl-ā-nu*
	I-NOM	School-DAT	Go-PST-1.SG.
	"I went to the school"		

The *vibhakti -ki* is used to express two different relations viz. *sampradānā* (k4) as in (8) and goal/destination (k2p) as in (9). In such cases, the semantics of the verb is considered to disambiguate the *vibhakti*. In example (9), the verb belongs to the class of [+motion] hence it has a requirement of k2p unlike the example (8). This semantic information is augmented with syntactic rules in order to mark the appropriate relation.

(ii) Lexical Semantics of Nouns

In some cases, it is the lexical choice of nouns that helps in resolving the ambiguity. For instance, when the *vibhakti-ki/-ku* is marked with *kāla-adhikaraṇam* (k7t) or *deśa-adhikaraṇam* (k7p) relation, corresponding nouns should be either place or time denoting terms as in example (10).

10.	*ravi-Ø*	*padi-Ø*	*gaMṭalaku*	*haidarabādu-ku*	*cērukuṇ-ṭā-ḍu*
	Ravi-NOM	10	Hour-DAT	Hyderabad-DAT	Reach-FUT-3.SG.M
	"Ravi will reach Hyderabad at 10'o'clock"				

Here, the noun expressing time i.e. *padi gaMtalu* '10 'o clock', and the place i.e. *haidarabādu* 'Hyderabad' are marked with *-ku*, however, they are marked as k7p and k7t respectively based on their semantics. In such cases, a list of these terms is maintained as linguistic cues to access the information.

The step-3 of the algorithm is to define local and global constraints. The local constraints used in the parser to postulate the best possible result are given below (Kulkarni 2019):

1. A node can have one and only one incoming edge.

2. There cannot be more than one outgoing edge with the same label from the same node if the relation corresponds to a *kāraka* relation.

3. There cannot be self-loops in a graph. In addition to the local constraints, we also use global constraints like *sannidhi* 'proximity' which is a constraint that restricts crossing of edges. The sample graph satisfying all the above local and global constraints is provided below:

11.	*nēnu-Ø*	*prasādu -tō*	*rēpu*	*madrāsu -lō*	*telugu*	*sinimā -ki*	*veḷ-tā-nu*
	I-NOM	Prasad-ASS	tomorrow	Madras-LOC	Telugu	Movie-DAT	Go-FUT-1.SG
	'I will go to a Telugu movie with Prasad in Madras tomorrow.'						

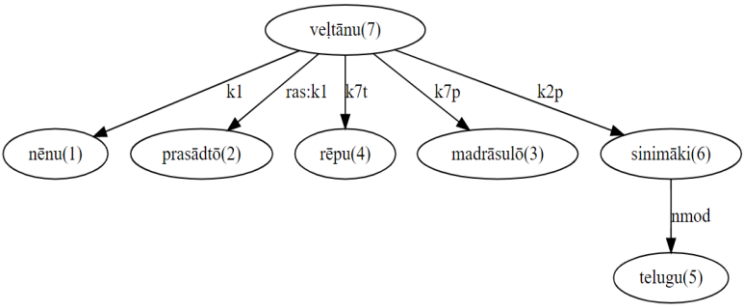


Figure-2 Sample graph for the example (11)

The use of semantic constraints is dealt with in step4 of the algorithm. It is quite important to include semantic constraints in a parser to arrive at the correct solution. For instance, the sentence *colourless green ideas sleep furiously* (Chomsky 1957) is a syntactically well-formed sentence but semantically ill-formed. The natural language feature which enables the use

of semantically well-formed constructions is termed as *yōgyatā* in PG or the selectional restriction in western terminology. The selectional restriction is defined as the semantic constraint imposed on the arguments of verbs. We use selectional restriction of arguments of the verb to prune out the non-congruent solutions and arrive at a single parse. Let us consider the following examples:

12.	*tūphānu-Ø*	*illu-Ø*	*kūlc-iM-di*
	Storm-NOM	House-ACC	destroy-PST-3.SG.N
	“The storm destroyed the houses”		

***13.**	*illu-Ø*	*tūphānu-Ø*	*kūlc-iM-di*
	house-NOM	storm-ACC	destroy-PST-3.SG.N
	“Houses destroyed the storm”		

Both examples (12) and (13) are syntactically well-formed sentences, when *yōgyatā* is applied, the example (13) stands semantically ill-formed because ‘Houses destroying the storm’ is a semantically unacceptable sentence. In order to solve such issues, the canonical word order of a language is used as a cue.

The other instance in which we use selectional restriction is to disambiguate *kartā* and karma in Telugu. When karma is [-animate], the *vibhakti* ∅ is used which is synonymous with the marker for *kartā*. In such cases, two ontological features [+/- animate] and [+/- human] could resolve the ambiguity in Telugu as well as in other Indian languages as examined by (Bharati, Akshar; Samar, Husain; Bharat, Ambati; Sambhav, Jain; Dipti, Sharma; & Rajeev, Sangal 2008). *kartā* is considered to be higher in its animacy hierarchical order in comparison with *karma*. Consider the following example:

14.	*nēnu-Ø*	*pāta-Ø*	*pāḍ-ā-nu*
	I.NOM	Song. ACC	sing-PST-3.SG.N
	“I sang a song”		

Here, the verb *pāḍu* ‘sing’ expects *kartā* with a semantic feature of [+human] thus, (n*ēnu*) ‘I’ is prioritized over a [-animate] entity (i.e. *patā*) ‘song’. These two semantic features proved to be quite helpful in resolving the most ambiguous relation of *kartā* and *karma.* As seen earlier, this parser exploits various linguistic information which stands crucial in disambiguating certain cases. In the next section, we present the results, which show the impact of linguistic information used in the parser.

5. Evaluation of the System

The parser is evaluated for its Labelled Attachment Score (LAS) and Unlabelled Attachment Score (UAS). In this section, the data used for evaluating parsers is presented followed by the results. Finally, we also present the error analysis and some observations.

5.1. Data

The present study selects 453 sentences to test parsers which are extracted from various sources such as (i) Telugu Grammar books viz. *telugu vākhyam* (Ramarao 1885) and A grammar of modern Telugu (Krishnamurti & Gwynn 1985) (ii) Random sentences from Telugu corpus (3 million words (CALTS[1]) corpus). The corpus contains sentences with intransitive verbs (223 sentences), transitive verbs (197 sentences), and ditransitive verbs (33 sentences). The sentences covering constructions such as copula, imperative, passive, dubitative,

1 Centre for Applied Linguistics and Translation Studies

interrogative, non-nominative subjects, reflexive and coordinating noun phrases are noticed.

5.2. Results

The results consist of the Unlabelled Attachment Score (UAS) where the dependency tree produced by the parser matches exactly with the tree from the gold data without considering the labels and the Labelled Attachment Score (LAS) which checks if the two relations and labels are correctly matched. Out of 453 sentences, 1043 relations are manually identified and annotated for the evaluation. MALT parser is developed with the data annotated. The rule-based parser produces correct dependency trees for 1001 relations and 969 correct labelled trees. Whereas MALT parser produces 928 relations, out of which 739 relations are correctly labelled. The results are provided in the table-2.

Parser type	**UAS**	**LAS**
Rule-Based Parser	96.5%	92.9%
MALT parser	89%	70.85%

Table 2: Results

Further, the rule-based parser output is analysed with different sentence structures as given in Table-3. The exact match and partial match of sentences are also identified.

Sentence Type	**No. of sentences**	**exact match**	**partial match**	**UAS**	**LAS**
Intransitive	223	208	18	97.6%	95.5%
Transitive	197	152	40	97%	92.4%
Ditransitive	33	20	11	86.6%	80%
Copula constructions	87	68	16	92.5%	80%

Imperative constructions	25	15	8	68%	52%
Dubitative constructions	56	36	18	64%	56%
Passive constructions	33	28	3q	90%	81%
Non-nominative subject constructions	66	38	25	48%	41%
Reflexive constructions	17	8	7	46%	33%
Interrogative constructions	62	48	10	85%	77%

Table-3 Simple sentence structures and results

The parsing errors in these simple sentence structures are studied which help in improving further the rules in the rule-based parser for Telugu.

5.2. Error Analysis and Observations

In this section, we discuss certain cases where the rule-based parser fails to provide the appropriate results. The current rule-based parser has a difficulty in dealing with the coordinating noun phrases and with certain pro-drop constructions. As seen in the example (15), the noun phrases *gāli nīru* 'air and water' are co-ordinating noun phrases, but the linguistic cue to express them as coordination such as either comma (,) (i.e., *gāli nīru*) or the vowel-length in the end (*gālī nīrū*) are not present. This makes the system identify them wrongly as separate relations.

15.	*ā*	*prāMtaM-lō*	*gāli nīru*	*lēvaṭa*
	that	place-LOC	water air	be-NEG-QUO
	"There is no water or air in that place"			

Certain verbs in Telugu do not show agreement with the *kartā*. In example (16), when the verb expresses the mood of possibility with the auxiliary verb *vaccu*, it does not show agreement with the verb. When the *kartā* is pro-dropped, the system identifies the *karma* (i.e. *cēpa* 'fish'), the zero-marked as *kartā*. Consider the example below:

16.	*cēpa.Ø*	*tin-a-vaccu*
	Fish.ACC	eat-INF-POSS
	"(subject) can eat fish"	

The other two reasons for the failure of the parser in certain cases are due to the wrong output from the pre-processing tools and the lack of a database for the parser. These are handled by correcting the pre-processing output and improving the database (vocabulary). Whereas, in data-driven parsers like MALT, it is difficult to improve the accuracy unless a huge annotated corpus is trained again.

6. Conclusion

This paper deals with building a rule-based parser for Telugu experimenting with simple sentences. A discussion on the application of the Pāṇinian grammatical model to Telugu and the algorithm is provided. This paper explains how the use of two semantic features viz. animacy and humanity enables the unambiguous marking of *kartā* and *karma* relations. The

results show that the rule-based parser proves to be better than the data-driven parser due to the inclusion of linguistic information. Further, the study aims to improve the accuracy of the pre-processing tools and also build the required database for Telugu parsing. The next phase of the study will focus on implementing the rule-based parser for all the sentence structures in Telugu and extending this algorithm to other Indian languages.

Appendix - List of tags used in the Telugu Parser

k1 (*kartā* 'Agent') **k2** (*karma* 'patient/goal') **k3** (*karaṇa* 'instrument') **k4**(*sampradāna* 'beneficiary') **k4a** (*anubhavāı kartā* 'Experiencer') **k5** (*apādāna* 'Source') **k7** (*viṣaya-adhikaraṇam* 'location elsewhere') **k7t** (*kāla-adhikaraṇam* location in time) **k7p** (*deśa-adhikaraṇam* 'location in space') **k2g**(*gounakarma*'secondary karma')	**r6***(ṣaṣṭhī karma* 'genitive') **rh** *(hetuḥ* 'reason') **rt** (*tātparya* 'purpose') **k1s**(*kartṛsamānādhikaraṇam'*complement of a *kartā*') **k2s** *(karmasamānādhikaraṇam* 'complement of a *karma'*) **adv (***kriyāviśeṣaṇnam* adverbs) **k*u**(*sādrishya* 'similarity') **rd** ('direction')	**ras-k*** (*upapada sahakārakatwa* 'associative') **case** ('for postpositions') **det** ('determiner') **enm**(enumerator (number words)) **jjmod**('adjective modifier' **lwg**('local word grouping) **nmod** ('noun modifier') **r6v** ('verb and noun relation') **rsym** ('symbols) **title** ('titles of names') **vmod** ('verb modifier')

References

AMBATI, BHARAT RAM; PHANI GADDE & KARAN JINDAL. 2009. Experiments in Indian Language Dependency Parsing. *Proceedings of the ICON09 NLP Tools Contest: Indian Language Dependency Parsing*. 32-37.

BHARATI, AKSHAR & RAJEEV SANGAL. 1993. Parsing Free Word Order Languages in the Paninian Framework. *31st Annual Meeting of the Association for Computational Linguistics.* 105-111.

BHARATI, AKSHAR, VINEET CHAITANYA, RAJEEV SANGAL & K. V. RAMAKRISHNAMACHARYULU. 1995. *Natural Language Processing: A Paninian Perspective*. New Delhi: Prentice Hall of India.

BHARATI, AKSHAR, SAMAR HUSAIN, BHARAT AMBATI, SAMBHAV JAIN, DIPTI SHARMA & RAJEEV SANGAL. 2008. Two Semantic Features Make All the Difference in Parsing Accuracy. *Proceedings of ICON* 8.

BHARATI, AMBATI, DIPTI MISRA SHARMA, SAMAR HUSAIN, LAKSHMI BAI, RAFIYA BEGUM & RAJEEV SANGAL. 2009. AnnCorra: TreeBanks for Indian Languages, Guidelines for Annotating Hindi TreeBank (version–2.0). *LTRC. Hyderabad:* IIIT Hyderabad.

BRESNAN, JOAN. 1982. Control and Complementation. *Linguistic Inquiry* 13(3). 343-434. http://www.jstor.org/stable/4178286 (Accessed 2nd June 2021).

CHOMSKY, NOAM. 1957. *Syntactic Structures* (2nd edition). Mouton de Gruyter: The Hague.

GARAPATI, UMA MAHESHWAR RAO. 1999. *Morphological Analyzer for Telugu*. (Electronic form). Hyderabad: University of Hyderabad.

GARAPATI, UMA MAHESHWAR RAO., RAJYARAMA KOPPAKA & SRINIVAS ADDANKI. 2012. Dative case in Telugu: A Parsing

Perspective. *Proceedings of the Workshop on Machine Translation and Parsing in Indian Languages.* 123-132.

GATLA, PRAVEEN. 2019. Dependency Parsing for Telugu Using Data-driven Parsers. *Language in India* 19(1). 185-197.

GAZDAR, GERALD, EWAN KLEIN, GEOFFREY K. PULLUM, & IVAN A. SAG. 1985. *Generalized Phrase Structure Grammar*. Cambridge, MA: Harvard University Press.

HUSAIN, SAMAR. 2009. Dependency Parsers for Indian Languages. *Proceedings of ICON09 NLP Tools Contest: Indian Language Dependency Parsing*.

KRISHNAMURTI, BH. & J. P. L. GWYNN. 1985. A Grammar of Modern Telugu. New Delhi: Oxford University Press.

KANNEGANTI, SILPA, HIMANI CHAUDHRY & DIPTI MISRA SHARMA. 2018. Comparative Error Analysis of Parser Outputs on Telugu Dependency Treebank. In Gelbukh A. (ed.), *Computational Linguistics and Intelligent Text Processing. CICLing 2016*. Lecture Notes in Computer Science 9623. Springer Cham. https://doi.org/10.1007/978-3-319-75477-2_28.

KESIDI, SRUTHILAYA REDDY, PRUDHVI KOSARAJU, MEHER VIJAY & SAMAR HUSAIN. 2013. Constraint based Hybrid Dependency Parser for Telugu. Hyderabad: International Institute of Information Technology Hyderabad doctoral dissertation.

KUMARI, B. VENKATA SESHU & RAMISETTY RAJESHWARA RAO. 2015. Improving Telugu Dependency Parsing Using Combinatory Categorial Grammar Supertags. *ACM Transactions on Asian and Low-Resource Language Information Processing (TALLIP)* 14(1), 1-10.

KUMARI, B. VENKATA SESHU & RAMISETTY RAJESHWARA RAO. 2017. Telugu Dependency Parsing Using Different Statistical Parsers. *Journal of King Saud University-Computer and Information Sciences* 29(1). 134-140.

KULKARNI, AMBA SHEETAL POKAR & DEVANAND SHUKL. 2010. Designing a Constraint-based Parser for Sanskrit. *International Sanskrit Computational Linguistics Symposium.* 70-90.

KULKARNI, AMBA., & K. V. RAMAKRISHNAMACHARYULU. 2013. Parsing Sanskrit Texts: Some Relation Specific Issues. *Proceedings of the 5th International Sanskrit Computational Linguistics Symposium.*

KULKARNI, AMBA. 2013b. A Deterministic Dependency Parser with Dynamic Programming for Sanskrit. *Proceedings of the Second International Conference on Dependency Linguistics (DepLing 2013).* 157-166.

KULKARNI, AMBA. 2019. *Sanskrit Parsing Based on the Theories of Sabdabodha.* Indian Institute of Advanced Study, Shimla and DK Publishers (P) Ltd.

NAGARAJU, B., N. MANGATHAYARU & B PADMAJA RANI. 2016. Dependency Parser for Telugu Language. *ICTCS '16: Proceedings of the Second International Conference on Information and Communication Technology for Competitive Strategies.* 1-5.

NALLANI, SNEHA, MANISH SHRIVASTAVA, M., & DIPTI MISHRA SHARMA. 2020. A Fully Expanded Dependency Treebank for Telugu. *Proceedings of the WILDRE5–5th Workshop on Indian Language Data: Resources and Evaluation.* 39-44.

NALLANI, SNEHA, MANISH SHRIVASTAVA, M., & DIPTI MISHRA SHARMA. 2020b. A Simple and Effective Dependency Parser for Telugu. *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop.* 143-149.

NIVRE, JOAKIM .2000. Statistical Parsing. *Handbook of Natural Language Processing* 2. 525–543.

PATEL PREETI K. 2010. *Vibhakti Divergence Between Sanskrit and Hindi.* Hyderabad: University of Hyderabad dissertation.

POLLARD, CARL JESSE & IVAN A. SAG. 1994. *Head-driven Phrase Structure Grammar*. Chicago: University of Chicago Press.

RAMA, TARAKA & SOWMYA, VAJJALA. 2018. A Dependency Treebank for Telugu. *English Conference Papers, Posters and Proceedings* 8. https://lib.dr.iastate.edu/engl_conf/8.

RAMARAO, CHEKURI. 1885. *Telugu Vākyaṁ*. Hyderabad: AP Sahitya Academy.

VEMPATY CHAITANYA, VISWANATHA NAIDU, SAMAR HUSAIN, RAVI KIRAN, LAKSHMI BAI, DIPTI MISHRA SHARMA & RAJEEV SANGAL. 2010. Issues in Analyzing Telugu Sentences towards Building a Telugu Treebank. In Gelbukh A. (ed.), Computational Linguistics and Intelligent Text Processing. *CICLing* 2010. Lecture Notes in Computer Science 6008. Berlin, Heidelberg: Springer. https://doi.org/10.1007/978-3-642-12116-6_5.

***

Cite This Work:

P., SANGEETHA, PARAMESWARI K. & AMBA KULKARNI. A Rule-based Dependency Parser for Telugu: An Experiment with Simple Sentences. *Translation Today*, Vol. 15(1). 123-144. DOI:10.46623/tt/2021.15.1.ar5

Parsing Subordinate Clauses in Telugu using Rule-based Dependency Parser

[1]Sangeetha Perugu, [2]Parameswari Krishnamurthy, [3]Amba Kulkarni
[1,2]Centre for Applied Linguistics and Translation Studies, [3]Centre for Sanskrit Studies
University of Hyderabad
[1]geethanjali.sheldon@gmail.com, {[2]pksh, [3]ambakulkarni}@uohyd.ac.in

Abstract

Parsing has been gaining popularity in recent years and attracted the interest of NLP researchers around the world. It is challenging when language under study is a free-word order language which allows ellipsis like Telugu. In this paper, an attempt is made to parse subordinate clauses especially, non-finite verb clauses and relative clauses in Telugu which are highly productive and constitute a large chunk in parsing task. This study adopts a knowledge-driven approach to parse subordinate structures using linguistic cues as rules. Challenges faced in parsing ambiguous structures are elaborated alongside providing enhanced tags to handle them. Results are encouraging and this parser proves to be efficient for Telugu.

1 Introduction

Parsing, the word derived from Latin (*pars orationis*), was originally used in elementary schools for grammatical explication of sentences (Nivre, 2006). Currently, parsing is a well-known and well-researched area in natural language processing (NLP) which involves analyzing sentences syntactically or syntactico-semantically. Building parsers and treebanks have attracted several researchers for its utility in various larger NLP applications. An efficient and ready-to-use parser for languages like Telugu, one of the most widely spoken Dravidian languages is still under development, though a handful of resources are traced.

Telugu is a south-central Dravidian language with free-word order and well-known for its agglutinating morphology. Agglutination allows carrying multiple grammatical information on words in Telugu. This grammatical information is quite helpful in parsing and stands as a rationale behind building the rule-based parser, despite multiple challenges. Parsing free-word order and agglutinating languages like Telugu is particularly challenging as they allow pro-drops, ellipsis and complex constructions. Earlier attempts in developing Telugu dependency parsers include mostly data-driven approaches (Ambati et al., 2009; Husain, 2009; Bharati et al., 2009; Kesidi et al., 2013; Kanneganti et al., 2016; Gatla, 2019; Nallani et al., 2020; Rama and Vajjala, 2018). Among the attempts made, UDPipe for Telugu[1] which is trained using Telugu-MTG UD treebank (Rama and Vajjala, 2018) is the only publicly accessible parser. There is an attempt in developing a rule-based parser with linguistic knowledge-driven approach (Sangeetha et al., 2021) for simple sentences. In this paper, we present our experiment in parsing subordinate clauses, particularly, non-finite verb clauses and relative participle clauses in Telugu using rule-based dependency parser.

2 A Rule-Based Dependency Parser

This study uses a rule-based parser (RBP) which takes input from sentences that are morphologically analysed. Telugu POS tagger, pruning and pick-one-morph modules are used to select one analysis per token (Rao, 1999). The RBP follows dependency approach based on the Indian theories of verbal cognition where three factors viz. *ākānksā* (expectancy), *yōgyata* (meaning compatibility), and *sannidhi* (proximity) are used and implemented initially for Sanskrit (Kulkarni, 2019). Telugu RBP is adopted from Sanskrit RBP and modified for Telugu parsing (Sangeetha et al., 2021). We model the parser as a tree where the nodes of a tree correspond to a word and the edges between nodes correspond to a relation between the corresponding words. Parser is implemented using the functional programming language Ocaml [2] to write rules and

[1] http://lindat.mff.cuni.cz/services/udpipe/
[2] https://ocaml.org/

Proceedings of the ICON 2021 Workshop on Parsing and its Applications for Indian Languages (PAIL) , pages 12–19
December 16, 2021. Silchar, India.

Perl to generate dependency trees as graphs. The figure 1 explains the architecture of the RBP.

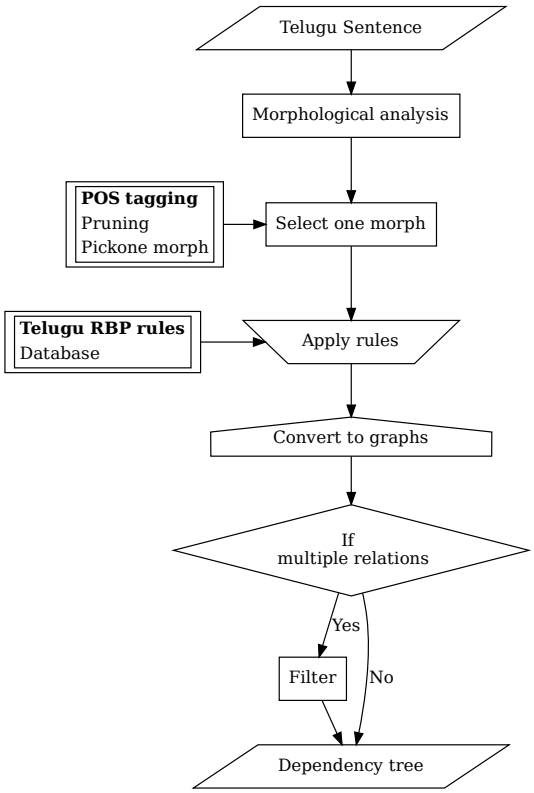


Figure 1: Architecture

In parsing simple sentences, 29 dependency labels are used and they are divided into kāraka(K) relations (for example, kartā (roughly equivalant to subject) (k1), karmā (object) (k2) etc.) and non-kāraka (for example, genitive (r6), associative(ras) etc.) labels. The dependency tree for the sentence (1) is seen in figure 2.

(1) mā nānna rēpu ūri nuMci
our father tomorrow village from
vas-tā-ru
come-FUT-3.SG.HON.
'My father will come from village tomorrow'

3 Subordinate Clauses in Telugu

Subordinate clauses in Telugu include non-finite verb clauses, relative participle clauses and complementizer clauses. Subordinate clauses in Telugu do express ambiguity with different syntactico-semantic relations.

Non-finite verb clauses are highly productive

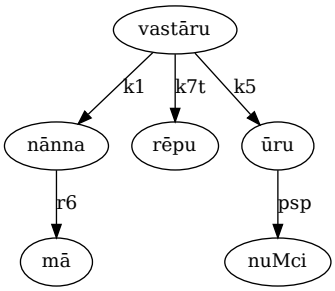


Figure 2: Dependency tree for sentence(1)

in the formation of sentences in Telugu and they constitute a large chunk in parsing task. They are dependent clauses which cannot stand alone in a sentence. They are realised as subordinate clauses which are derived from simple sentences with certain structural changes and precede the matrix clause by occurring to their left side. The verb of subordinate clause is syntactically the head of the clause but does not exhibit person-number-gender agreement with respective subjects, however it is marked for appropriate tense, aspect and mood. They are classified into conjunctive participles, conditionals, concessives and infinitives in Telugu (Krishnamurti and Gwynn, 1985). Conjunctive participles are divided into past, durative and negative. Conditionals and concessives clauses can have both affirmative and negative forms whereas infinitives can have only affirmative form.

Relative participle clauses are primarily noun phrases which are further divided into past, durative, future/habitual and negative participles. Negative participles do not differentiate for tense. Complementizer clauses are formed by the quotative form i.e. *ani* 'that' which links both finite clauses. Figure 3 provides the classification of subordinate clauses in Telugu. Examples of various types of subordinate clauses are provided in the table 1.

In this paper, we present challenges in parsing non-finite verb clauses and relative participle clauses using rule-based parsing. We use the anncora tagset for tagging the dependency relations (Version 2.5) (Bharati et al., 2009). There is a great requirement for the enhancement of tags for Telugu to disambiguate various functions of subordinate clauses. An attempt is made to build enhanced tags and implemented using linguistic cues as rules in RBP.

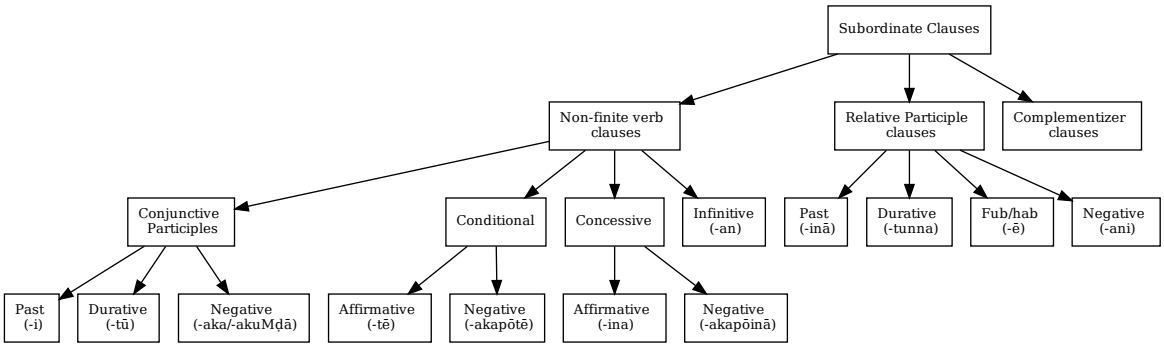


Figure 3: Types of non-finite clauses in Telugu

Type of subordinate clause	Example
I. Non-finite verb clauses	
Conjunctive Participle	
Past	*tin-**i*** 'having eaten'
Durative	*tin-**ṭū*** 'along with eating'
Negative 1 (*-akuMḍa*)	*tina-**kuMḍā*** 'not having eaten'
Negative 2 (*-aka*)	*tin-**aka*** 'due to not having eaten'
Conditional	
Affirmative	*tin-**tē*** 'if one eats'
Negative	*tin-**akapōtē*** 'if one does not eat'
Concessive	
Affirmative	*tin-**inā*** 'inspite of having eaten'
Negative	*tin-**akapōinā*** 'inspite of not having eaten'
Infinitive	*tin-**(an)*** 'to eat'
II. Relative Participle	
Past	*tin-**ina** abbāyi* 'the boy who ate'
Durative	*tin-**tunna** abbāyi* 'the boy who is eating'
Future-habitual	*tin-**ē** abbāyi*'the boy who will eat'
Negative	*tin-**ani** abbāyi* 'the boy who did not eat'

Table 1: Examples of subordinate clauses

.

4 Challenges in Parsing Subordinate Clauses

Subordinate clauses in Telugu are ambiguous across certain sub-types. These ambiguous constructions pose various parsing challenges mainly due to multiple functions or interpretations of a non-finite marker which causes ambiguity. Certain ambiguous constructions with non-finite verb clauses and relative participle clauses in Telugu are discussed in this section.

4.1 Conjunctive participle clause

The conjunctive participle clause occurs as a subordinate clause and modifies the matrix clause. This conjunctive participle clause can be used to express verbal modifier (`vmod`) functions such as serial action, manner and simultaneous action in Telugu. Example (2) explicates conjunctive participle as a serial verb. The figure 4 is shown with the tag `vmod:cp_serial` for the sentence (2) with conjunctive participle expressing serial action.

(2) rāmuḍu.∅ annaM.∅ **tin-i**
Ram.NOM food.ACC eat-CP.PST
paḍukunn-ā-ḍu
sleep-PST-3.SG.M
'Ram ate food and slept'

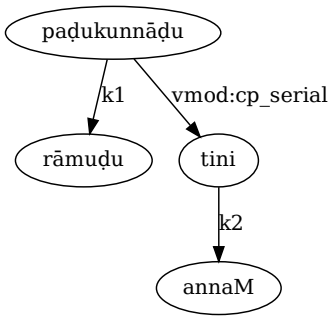


Figure 4: Dependency tree for (2)

The conjunctive participle can express manner as explicated in the sentence (3) with the Figure 5. Here, the verb class i.e. *motion verbs* is used as a cue to identify the manner in the verb modification with the tag `vmod:cp_manner`.

(3) vimala.∅ āphīsu-ku nadic-i
vimala.NOM office-DAT walk-CP.PST
veḷt-uM-di
go-HAB-3.SG.F
'Vimala goes to office by walk'

The conjunctive participles express simultaneous action when the participle is durative as in the sentence (4).

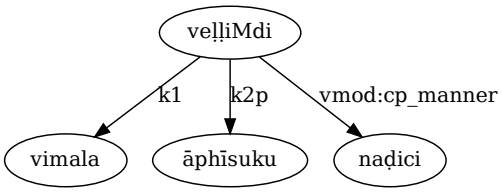


Figure 5: Dependency tree for (3)

(4) prakāsh.∅ sinimā **cūs-tū**
prakash.NOM cinema watch-CP.DUR
cūldriMk tāg-ā-ḍu
cool-drink drink-PST-3.SG.M
'Prakash drank cool drink while watching a cinema'

Figure 6 shows a dependency tree of the sentence (4) adding a new tag `vmod:cp_simul`.

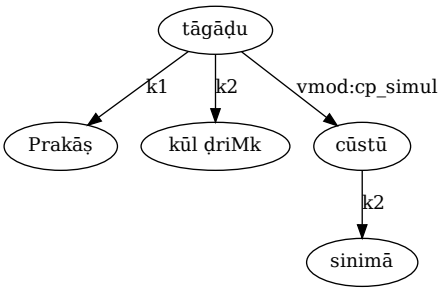


Figure 6: Dependency tree for (4)

However, when the active form of conjunctive participle verb is followed by the passive matrix verb, it renders an ambiguous interpretation. Consider example (5) from (Ramarao, 2017, pg. 116) and its dependency tree in the Figure 7.

(5) sujāta **tiraskariMc-i**
sujata.NOM reject-CP.PST
avamāniMc-a-baḍ-iM-di
insult-PASS-PST-3.SG.F
'Sujata rejected (someone) and was insulted' or 'Sujata got rejected and was insulted'.

Example (5) is ambiguous due to argument ellipsis. This can be interpreted in two different ways by supplying either a passive subject (as in (6)) or the object (as in (7)) in the non-finite clause. This ambiguity is represented in Figure 7.

(6) sujāta vāḍi cēta **tiraskariMc-(abaḍ̄)i**
sujata.NOM he by reject-(PASS).CP.PST
avamāniMc-abaḍ-iM-di
insult-PASS-PST-3.SG.F
'Sujata got rejected by him and was insulted'

(7) sujāta vāḍi-ni **tiraskariMc-i**
sujata.NOM he-ACC reject-CP.PST
avamāniMc-abaḍ-iM-di
insult-PASS-PST-3.SG.F
'Sujata rejected him and was insulted'

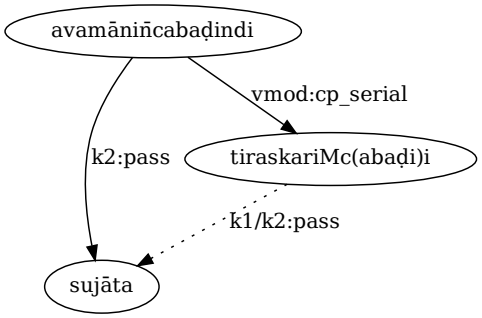


Figure 7: Dependency tree for (5)

Other cases include constructions with negative matrix verb percolating its features to the conjunctive participle resulting in ambiguity as in the sentence (8).

(8) ravi.∅ kāphī.∅ **tāgi**
Ravi.NOM coffee.ACC drink-CP.PST
skūl-ki veḷḷ-a-lēdu
school-DAT go-PST-NEG
'Ravi drank coffee but he did not go to school/ It is not coffee that Ravi drank (but something else) and went to school'

Since disambiguating senses in (8) is not in the scope of parsing and it requires deep semantic analysis, the dependency tree does not show the difference in meaning as in the figure 8.

However, the occurrence of the particle *kūḍa* 'also' after the participle form helps in disambiguating and the negative percolation from the matrix to subordinate clause is prevented.

(9) ravi.∅ kāphī.∅ **tāg-i** **kūḍa**
Ravi.NOM coffee.ACC drink-CP.PST also
skūl-ki veḷḷ-a-lēdu
school-DAT go-PST-NEG
'Ravi drank coffee but he did not go to school'

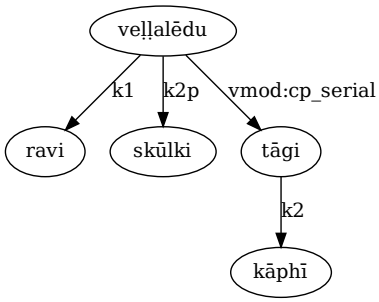


Figure 8: Dependency tree for (8)

4.2 Conditional clauses

Conditional clauses in Telugu not only express conditional sense but also show other interpretations leading to several parsing analyses. Such constructions are identified and tagged differently in the RBP.

Sentences (10) and (11) differ with the use of tense in finite verb and render different senses. If the finite verb of a complex sentence is in non-past tense, it is considered as a conditional clause and will be tagged with `vmod:cond`. Whereas, if the matrix verb is in the past tense, the conditional verb expresses the serial action and is given the tag `vmod:cond_serial` as the sentence (11).

(10) rāyi-tō **koḍi-tē** kāya kiMda
stone-INST hit-COND fruit-NOM down
padu-tuM-di
fall-NON.PST-3.N.SG
'If you hit with a stone, the fruit falls'

(11) rāyi-tō **koḍi-tē** kāya kiMda
stone-INST hit-COND fruit-NOM down
pad-iM-di
fall-PST-3.N.SG
'The fruit fell when hit with a stone'

Other exceptional case of conditional suffix rendering non-conditional sense include the causal meaning. In the sentence (12) (Ramarao, 2017, pg. 129), the verb of non-finite clause *tiM-tē* expresses the cause for the main action and can be alternated with conjuctive participle form *tini* 'having eaten'. The subject *subbārāvu* 'Subbarao' is shared with both non-finite and matrix clauses. Shared subject constraint is used as a syntactic cue in order to parse these constructions and tag `vmod:cond_cause` is attached in the dependency tree as in 9.

(12) subbārāvu guḍlu **tiMṭe**
Subbarao-NOM eggs eat-NF-COND
baliṣ-ā-ḍu
fat-become-PST-3.SG.M
'Subbarao became strong by eating eggs'

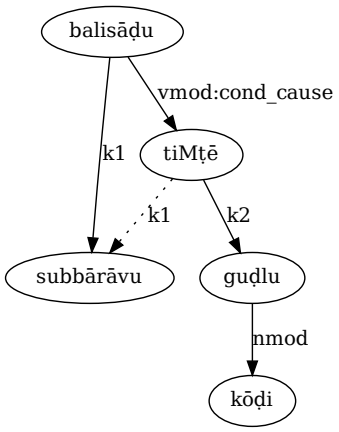


Figure 9: Dependency tree for (12)

4.3 Concessive clauses

Concessive clauses in Telugu are formed by adding the suffix *-inā* to the verb stem and express the meaning 'even if/even though'. It functions as adverbial modifiers to the matrix verb. The negative concessive form is formed by the suffix 'akapoyinā'. This clause is tagged as `vmod:conc` in the rule-based parser.

(13) nēnu cadiv-inā pāsu
I-NOM study-NF-CONC
avva-lēdu
become-NEG
'Even after studying, I did not pass (the examination)'

4.4 Infinitive clauses

Infinitive clauses are not very common in Telugu. The infinitive suffix in Telugu is *-an* and the tag `vinf:k1` is used in tagging infinite clauses when they occur in the subject position as in the sentence (14) and the respective dependency tree in Figure 11.

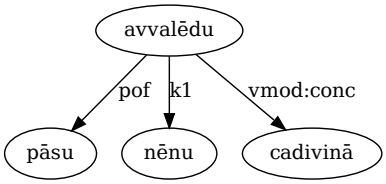


Figure 10: Dependency tree for (13)

(14) mīru nā-tō ā viṣayaM cepp-an
I-HON I-INST that matter tell-INF
akkar-lēdu
need-NEG
'You need not tell me that matter'

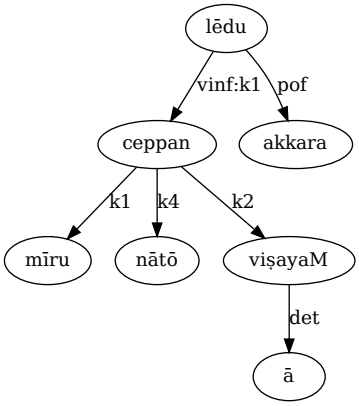


Figure 11: Dependency tree for (14)

4.5 Relative Participle Clauses

A simple sentence can be changed into a relative clause by replacing its finite verb by a relative participle (or verbal adjective) in the corresponding tense-mode and shifting the noun that it qualifies as head of the construction (Krishnamurti and Gwynn, 1985). Relative participle clauses occur immediately before nouns which they qualify. In Telugu, they show the distinction in tense in affirmative construction whereas in negative they do not show the tense.Relative participles are tagged as `nmod:relcl` in RBP. `nmod:relcl` is added with the argument relation of the noun which is relativized. In the sentence (15), the relativized nouns holds the object (k2) relation with the relative participle whereas the sentence (16) with the subject (k1) relation. There are tagged as `nmod:relcl_k2` and `nmod:relcl_k1` respectively in Figures 12 and 13.

(15) nēnu cūs-ina maniṣi iMṭi-ki
I.NOM see-RP.PST man home-DAT
vacc-ā-ḍu
came-PST-3.SG.M
'The man whom I saw came home'

(16) nan-nu cūsina maniṣi iMṭi-ki
I-ACC see-RP.PST man home-DAT
vacc-āḍu
come-PST-3.SG.M
'The man who saw me came home'

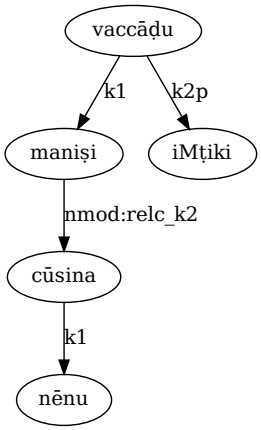


Figure 12: Dependency tree for (15)

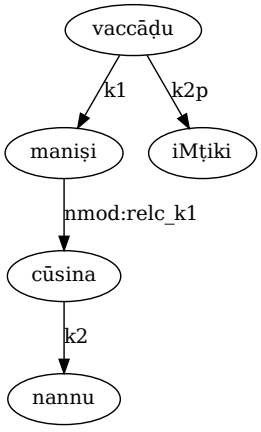


Figure 13: Dependency tree for (16)

Relative participle clause constructions are ambiguous when the noun in the relative clause has the potential to be an agent followed by the relative

participle form of the verb which is transitive.

(17) nēnu tin-ina kaMcaM pāta-di
I.NOM eat-RP.PST plate old-3.SG.N
'The plate in which I ate is old'/'The plate which I ate is old'

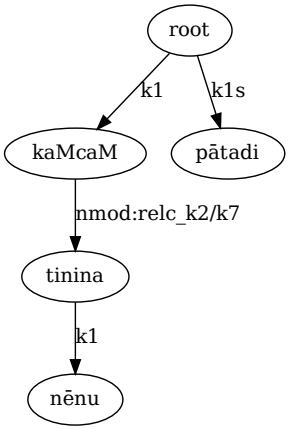


Figure 14: Dependency tree for (17)

The token *kaMcaM* 'plate' can be interpreted with the tag `k7` (location) as well as `nmod:relc_k2` as in figure 14. However, we use selectional restriction rules to rule out one of the analysis as eating *kaMcaM* 'plate' with the tag `nmod:relc_k2` is semantically not possible.

5 Enhanced Anncora Tagset

Anncora guidelines (Bharati et al., 2009) suggest the tag `vmod` for conjunctive participles, concessives, conditionals and *nmod* for relative participles. In this study, we have used multiple linguistic cues and enhanced subordinate clause tags as shown in the table 2. Around 41 rules with linguistic cues have been used to parse both simple and subordinate clauses in Telugu.

6 Evaluation

Rules of RBP are framed based on the model sentences collected from various Telugu grammar books Krishnamurti and Gwynn (1985), Ramarao (1975), Krishnamurti (2003) & (Ramarao, 2017). The purpose of choosing grammar texts for building rules is due to the wide-range of exceptions that are covered. These exceptions enabled us to segregate several cases of subordinate clause occurrences and providing fine-grain tags. Around

Subordinate clause	Enhanced Tag for Telugu
conjunctive participle	vmod
serial action	vmod:cp_serial
simultaneous action	vmod:cp_simul
Manner	vmod:cp_manner
conditional clauses	
condition	vmod:cond
serial action	vmod:cond_serial
cause	vmod:cond_cause
concessive clause	vmod:conc
infinitive clause	vinf:k1
Relative participle clause	
relativization of subject	nmod:relcl_k1
relativization of object	nmod:relcl_k2
relativization of location	nmod:relcl_k7

Table 2: Dependency Tags for Subordinate Clauses in Telugu

250 sentences were collected from news paper data for testing subordinate clauses. The labelled attachment score (LAS) is **72%** and unlabelled attachment score is **81%**. The Table 3 shows the LAS and UAS various sub-type of subordinate clauses.

Type of clauses	LAS	UAS
Conjunctive participle clauses	77.7%	86.2%
Conditional clauses	70.5%	82%
Concessive clauses	69.6%	80%
Infinitive clauses	64%	64%
Relative participle clauses	66.7%	73.2%

Table 3: Results of various subordinate clauses

RBP works on the linguistic cues (verbal/nominal databases, grammatical information) provided to it. RBP fails when these linguistic cues are not included as part of database or when it encounters an exception. But these cues can be updated as and when RBP encounters a new corpus. Another case in which RBP fails to deliver a correct parse is when pre-processing tools like morphological analyser, POS, pruning, pick-one morph provide an erroneous output.

7 Conclusion

Parsing of non-finite verb clauses and relative participle constructions in Telugu is attempted in this paper using a rule-based parser. It is observed that knowledge-driven parser works better for agglutinating languages like Telugu as many linguistic cues can be seen in the structure. Parsing of subordinate clauses is challenging due to its diverse interpretations and usage. Various ambiguous constructions are considered in this paper alongside

adding enhanced/fine-grain tags to the existing Anncora tagset. These tags are beneficial as the tag `vmod` is quite under-specified. Results prove that RBP serves as an efficient parser for Telugu and addition of linguistic cues can improve the performance further. Parsing of other complex structures will be carried out in the future work.

Acknowledgments

We thank the reviewers for their critical comments which immensely helped us in improving this paper.

References

Bharat Ram Ambati, Phani Gadde, and Karan Jindal. 2009. Experiments in indian language dependency parsing. *Proceedings of the ICON09 NLP Tools Contest: Indian Language Dependency Parsing*, pages 32–37.

Akshar Bharati, Dipti Misra Sharma, Samar Husain, Lakshmi Bai, Rafiya Begum, and Rajeev Sangal. 2009. Anncorra: Treebanks for indian languages, guidelines for annotating hindi treebank. *LTRC, IIIT Hyderabad, India*. Version 2.

Praveen Gatla. 2019. Dependency parsing for telugu using data-driven parsers. *Language in India*, 19(1).

Samar Husain. 2009. Dependency parsers for indian languages. In *Proceedings of ICON09 NLP Tools Contest: Indian Language Dependency Parsing*.

Silpa Kanneganti, Himani Chaudhry, and Dipti Misra Sharma. 2016. Comparative error analysis of parser outputs on telugu dependency treebank. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 397–408. Springer.

Sruthilaya Reddy Kesidi, Prudhvi Kosaraju, Meher Vijay, and Samar Husain. 2013. *Constraint-based hybrid dependency Parser for Telugu*. Ph.D. thesis, Ph. D. thesis, International Institute of Information Technology Hyderabad.

Bhadriraju Krishnamurti. 2003. *The dravidian languages*. Cambridge University Press.

Bhadriraju Krishnamurti and John Peter Lucius Gwynn. 1985. *A grammar of modern Telugu*. Oxford University Press, USA.

Amba Kulkarni. 2019. *Sanskrit Parsing: Based on the Theories of Śābdabodha*. DK Printworld (P) Ltd.

Sneha Nallani, Manish Shrivastava, and Dipti Misra Sharma. 2020. A simple and effective dependency parser for telugu. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, pages 143–149.

Joakim Nivre. 2006. *Inductive dependency parsing*. Springer.

Taraka Rama and Sowmya Vajjala. 2018. A dependency treebank for telugu. In *English Conference Papers, Posters and Proceedings*, 8, pages 119–128.

C Ramarao. 1975. *Telugu vākhyam*. A.P. Sahitya Academy.

Chekuri Ramarao. 2017. *A reference grammar of modern Telugu*. EMESCO books Pvt. ltd., Hyderabad.

G. Uma Maheshwar Rao. 1999. *A Morphological Analyzer for Telugu*. (electronic form). Hyderabad: University of Hyderabad. Accessible at.

P Sangeetha, Parameswari K., and Amba Kulkarni. 2021. A rule-based dependency parser for telugu: An experiment with simple sentences. *Translation Today*, 15(1).

పి. సంగీత
98480 84364

యంత్రానువాదంలో వాక్యవిశ్లేషకం అక్కర

1. పరిచయం

గత దశాబ్ద కాలంలో, సంగణక భాషాశాస్త్రం ప్రపంచవ్యాప్తంగా పరిశోధకుల దృష్టిని ఆకర్షించింది. ఆంగ్లంలోనేగాక ఇతర భాషలలో సంగణక సహాయక భాషా సాధనాలు రూపొందించేందుకు పరిశోధకులు మొగ్గుచూపుతున్నారు. ఈ భాషా సాధనాలలో యంత్రానువాదం అత్యంత ఆసక్తికరమైన మరియు కష్టసాధ్యమైనదిగా పేర్కొనబడింది. ప్రపంచ భాషలలో ఏ భాష నుండైనా మనకు కావలసిన భాషకు సులువుగా, తక్కువ సమయంలో అనువాదం చేసుకునేందుకు యంత్రానువాదం ఎంతో దోహదపడుతుంది. యంత్రానువాదంలో వాక్యవిశ్లేషకం (పార్సర్) అత్యంత ఆవశ్యకం. పార్సర్ అనేది వాక్యాన్ని వ్యాకరణపరంగానూ అర్థపరంగానూ విశ్లేషించేందుకు ఉపయోగించే సాధనం. ఈ వ్యాసంలో వాక్యవిశ్లేషకం గురించి దాని ఆవశ్యకత గురించి మేము రూపొందించిన తెలుగు వాక్యవిశ్లేషకం గురించి క్లుప్తంగా వివరిస్తున్నాను.

2. వాక్యవిశ్లేషణ – పరిచయం

వాక్యంలో ఉన్న ప్రతి పదానికీ అదే వాక్యంలో ఉన్న ఇతర పదాలకూ మధ్యగల అంతర్గత సంబంధాన్ని వ్యాకరణపరంగా/వాక్యనిర్మాణపరంగా (syntactically) లేదా అర్థపరంగా (semantically) విశ్లేషించడాన్ని పార్సింగ్ అంటారు. పదాల మధ్య ఉన్న సంబంధాలు కనుగొని ప్రతి సంబంధానికీ ఒక ప్రత్యేక టాగ్‌ను ఇవ్వడం పార్సింగ్ ప్రక్రియ ప్రధాన ఉద్దేశ్యం. ఒక వాక్యం యొక్క విశ్లేషణలో (parse), నాడికలు (nodes), వాక్యంలోని పదాలకు టీకలూ (సంవర్గ చిహ్నాలూ) పదాల మధ్య రేఖలూ వాటి మధ్యన ఉన్న సంబంధాన్ని తెలియజేస్తాయి (క్రింది బొమ్మలను చూడండి). ఈ పార్సింగ్ ప్రక్రియకు ఉపకరించే సాధనాన్ని పార్సర్/వాక్యవిశ్లేషకం అంటారు. ఏ వాక్యంలోనైనా ముఖ్యమైనది క్రియ. అయితే క్రియ, వాక్య నిర్మాణంలో కీలకమైన పాత్రను పోషించడానికి చాలా కారణాలు ఉన్నప్పటికీ ఆ అంశాన్ని మనం ఇక్కడ విపులంగా చర్చించడం లేదు. వాక్యంలో ఉన్న పదసంబంధాలు అన్నీ కూడ క్రియను అధారంగా చేసుకుని ఏర్పడినవే కనుక, వాక్యవిశ్లేషకం నిర్మాణంలో క్రియ తప్పనిసరిగా ప్రధానపాత్ర పోషిస్తుంది. అయితే ప్రతిసారీ క్రియాసహిత వాక్యాలే కాక కొన్నిసార్లు క్రియారహిత వాక్యాలు సాధారణంగా కూడా ఉపయోగించడం జరుగుతుంది (ఉ.దా. "నేను వైద్యుడిని"). ఇలాంటి క్రియారహిత వాక్యాలకు మాత్రం మేము రూపొందిస్తున్న వాక్యవిశ్లేషకంలో అంతరంగనిర్మాణంలో సహాయక క్రియను చేర్చి బాహ్యనిర్మాణంలో తొలగించడం జరుగుతుందని భావించడం జరిగింది.

సాధారణంగా వాక్యంలో ఏర్పడిన నిర్మాణాత్మకమైన సందిగ్ధత(structural ambiguity)ని తొలగించటానికి వాక్యవిశ్లేషకాన్ని ఉపయోగించడం జరుగుతుంది. వాక్యవిశ్లేషణ చేసేందుకు అనేక వ్యాకరణ సిద్ధాంతాలు అందుబాటులో ఉన్నా కూడా వాటిలో సంగణక ప్రక్రియలకు ఎక్కువగా ఉపయోగించేవి పద నిర్మాణ ఆధారిత (dependency) మరియు పద వర్గ (constituency) వ్యాకరణ సిద్ధాంతాలు. పద నిర్మాణ ఆధారిత (Dependency) వ్యాకరణ సిద్ధాంతం పాణినీయ వ్యాకరణ సాంప్రదాయం నుండి తీసుకోబడినది. ఈ సిద్ధాంతం భారతీయ భాషలకు అనువైనది అని ప్రముఖ భాషాశాస్త్ర పరిశోధకుల అభిప్రాయం. ఈ పత్రంలో పద నిర్మాణ ఆధారిత వ్యాకరణ సిద్ధాంతాన్ని ఉపయోగించడం జరిగింది.

ఆధునిక పద నిర్మాణ ఆధారిత సిద్ధాంతం ఫ్రెంచ్ భాషాశాస్త్రవేత్త లూసియన్ తెస్నేర్ (1959) ప్రతిపాదించారు. ఆయన రాసిన "ఎలెమెంట్స్ అఫ్ స్ట్రక్చరల్ సింటాక్స్ (Elements of structural syntax)" అనే పుస్తకంలో ఈ వ్యాకరణ సిద్ధాంతం గురించి స్పష్టమైన రీతిలో వివరించి, ఇది ఏ భాషను విశ్లేషించేందుకైనా అనువైనది అని పేర్కొన్నారు.

వాక్యవిశ్లేషకం రూపొందించే క్రమంలో టీకాసమితి (tagset) ఎంపిక అనేది చాలా ముఖ్యమైన అంశం. పదాల మధ్య గల సంబంధాన్ని తెలియజేయడానికి ప్రతి సంబంధానికి ప్రత్యేకమైన పేరు గల టీకా సమితి అవసరం అవుతుంది. ప్రస్తుతం అనేక రకాల టీకా సమితులు వాడుకలో ఉన్నప్పటికీ వాటిలో పెన్ (penn) టీకాసమితి, ప్రేగ్ టీకాసమితి, స్టాన్ఫోర్డ్ టీకాసమితి, UCREL టీకాసమితి, ఆన్‌కోరా (anncora) టీకాసమితులు ప్రధానమైనవి. ఈ వాక్యవిశ్లేషకానికి ఆన్‌కోరా (anncora) టీకానమితి ఉపయోగించడం జరిగింది. ఆన్‌కోరా (anncora) టీకాసమితి పాణిని వ్యాకరణం ఆధారంగా రూపొందించబడింది. ఈ టీకాసమితి వాక్యంలోని పదాల మధ్య ఉన్న కారక మరియు ఆకారక సంబంధాల గురించి వివరిస్తుంది. అయితే కారక సంబంధాలు క్రియకు మరియు ఇతర నామవాచకాలకూ మధ్య ఏర్పడే సంబంధాల గురించి తెలియజేస్తే, అకారక సంబంధాలు మాత్రం క్రియతో కాకుండా వాక్యంలోని ఇతర భాగాలతో ఏర్పడే సంబంధాల (ఉదా. రవి యొక్క ఇల్లు – ఈ పదబంధంలో "రవి"కి మరియు "ఇల్లు"కి మధ్య గల సంబంధం అకారక సంబంధం) గురించి తెలియజేస్తాయి. ఆన్‌కోరా (anncora) టీకాసమితిలో ప్రస్తుతం 19 రకాల కారక సంబంధాలు అందుబాటులో ఉన్నాయి. అవి: కర్త (k1), కర్మ (k2), ప్రయోజక కర్త (pk1), కర్త సమానాధికారణ (k1s) మొదలగు సంబంధాలు ఉన్నాయి. ఈ టాగ్ సెట్ లో ఉన్న సంబంధాలేగాక కొత్త వాక్య సంబంధాలు కావలసివచ్చినప్పుడు కొత్త పేర్లు ఈ టీకాసమితిలో చేర్చడం జరుగుతుంది.

ఈ క్రింది విభాగంలో తెలుగు వాక్యవిశ్లేషణ ఎందుకు కష్టం, ఎలాంటి సమస్యలు తలెత్తుతాయో అన్నఅంశాలు చర్చించబడ్డాయి.

3. విభక్తులు – వాక్యవిశ్లేషణ

తెలుగులో విభక్తులు వివిధ రకాల ప్రత్యయాల ద్వారా సూచింపబడతాయి. అయితే ఒక విభక్తికి ఒకే ప్రత్యయం కనుక వాడినట్టయితే వాక్య విశ్లేషణం చాలా సులువుగా ఉండేది. కానీ తెలుగులో ఒకే విభక్తికి అనేక ప్రత్యయాలు, ఒకే ప్రత్యయానికి అనేక విభక్తులు వాడే సదుపాయం భాషలో ఉండటం వల్ల వాక్య విశ్లేషణ కష్టతరమైన ప్రక్రియగా మారుతుంది.

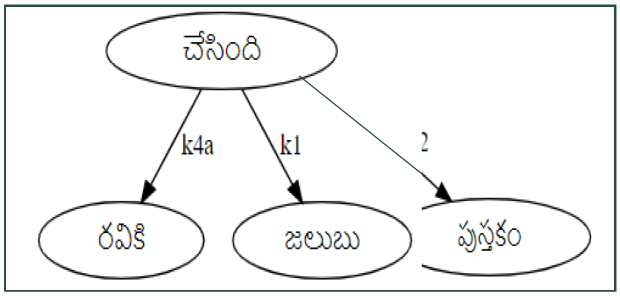
3.1. ఒకే ప్రత్యయం – అనేక విభక్తులు

తెలుగు వాక్యంలోని నామవాచక పదాలు చాలావరకు విభక్తి

ప్రత్యయాలు కలిగి ఉంటాయి. ప్రథమావిభక్తికి-ϕ (ప్రత్యయం ఏమీ లేకపోవడం), ద్వితీయావిభక్తికి-ని/ను, షష్ఠీవిభక్తి -యొక్క-ϕ ప్రత్యయాలు తరచుగా ఉపయోగిస్తారు. అయితే చాలా సందర్భాల్లో ఒకే ప్రత్యయం అనేక విభక్తులను వ్యక్తపరుస్తుంది. ఉదాహరణకు:

1. రవికి జలుబు చేసింది.
2. నేను రవికి పుస్తకం ఇచ్చాను.

పై రెండు వాక్యాల్లో "రవి" నామవాచకానికి "-కు/కి " ప్రత్యయం చేర్చబడింది. కానీ కింది రెండు వాక్యాల్లో వాడిన "-కి/కు" ప్రత్యయానికి వేర్వేరు అర్థాలు ఉన్నాయి. ఇలాంటి విభక్తి సందిగ్ధత తొలగించేందుకు వాక్యవిశ్లేషణ చాలా ఉపయోగపడుతుంది.

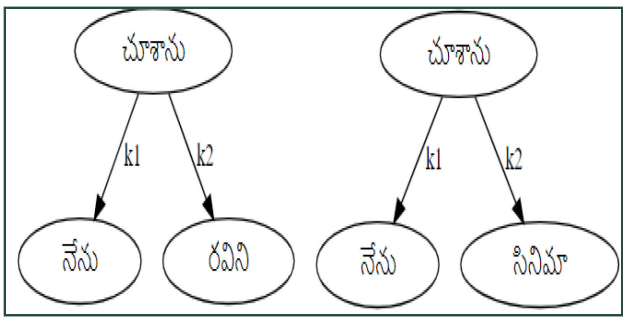


3.2. ఒకే విభక్తి - అనేక ప్రత్యయాలు

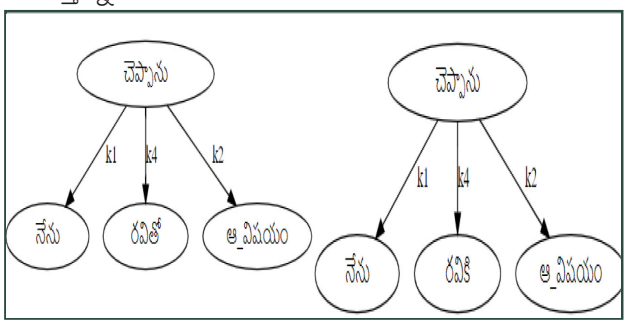
అంతేకాకుండా ఒకే విభక్తి, వేర్వేరు ప్రత్యయాల ద్వారా కూడా వ్యక్తం కావడం భాషలో చాలా తరచుగా జరుగుతుంది. ఉదాహరణకు,

3. నేను రవిని చూశాను
4. నేను సినిమా(ϕ) చూశాను

అనే రెండు వాక్యాలను తీసుకుందాం. మొదటి వాక్యంలో "నేను" అనే పదం "చూశాను" అన్న క్రియాపదానికి కర్తగా వ్యవహరిస్తే, "రవిని" అనే పదం వాక్యంలో కర్మగా వ్యవహరిస్తోంది. అలాగే రెండవ వాక్యంలో "నేను" కర్తగా వ్యవహరిస్తే, "సినిమా" కర్మగా వ్యవహరిస్తోంది. ఈ రెండు వాక్యాలనూ పరిశీలిస్తే మొదటి వాక్యంలో కర్మకు "-ని" అనే ప్రత్యయం చేర్చబడింది. కానీ రెండవ వాక్యంలో కర్మకు "-ని" ప్రత్యయం చేర్చబడలేదు. రెండు వాక్యాల్లో ఒకే రకమైన క్రియను ఉపయోగించినప్పటికీ విభక్తులలో తేడాలు ఉన్నాయి. ఇటువంటి తేడాలు సహజంగా కర్మ యొక్క చేతన (animate), అచేతన (inanimate) లక్షణాల కారణంగా ఏర్పడుతాయి. అయితే వాక్యవిశ్లేషకం ఈ రెండు వాక్యాలను క్రింద బొమ్మల్లో చూపించిన విధంగా విశ్లేషిస్తుంది.



వాక్యవిశ్లేషణని ఇంకొన్ని ఉదాహరణలతో అర్థం చేసుకునే ప్రయత్నం చేద్దాం. "నేను రవి-కి/తో ఆ విషయం చెప్పాను" అనే వాక్యంలో "-కి/తో" ప్రత్యయాలు రెండూ తృతీయా విభక్తిని సూచిస్తున్నాయి.



పైన గీసిన చెట్ల పటాలను పరిశీలిస్తే "కి" / "తో" ప్రత్యయాలు రెండూ ఒకే సంబంధాన్ని సూచిస్తున్నాయి. కానీ ఉదాహరణకు "రవి కత్తితో మామిడిపండును కోశాడు" అనే వాక్యాన్ని తీసుకుంటే "తో" ప్రత్యయం "కత్తి"తో ఉపయోగించినప్పటికీ "రవితో/కత్తితో" అనే రెండు నామవాచకాలు ఒకే సంబంధాన్ని చూపించడం లేదు. ఇలాంటి వ్యాకరణ సందిగ్ధతలను తీర్చుతూ వాక్యాలను స్వయంచాలకంగా విశ్లేషించడంలో సంగణక విశ్లేషకం పాత్ర ఎంతైనా అవసరం అవుతుంది.

ఈ పత్రంలో కేవలం విశ్లేషకాన్ని పరిచయం చేసే ప్రయత్నం మాత్రమే జరిగింది. అందుకోసం దానికి తగిన చాలా సులువైన వాక్యాలను పైన ఉదాహరణలుగా వివరించడం జరిగింది. అందువల్ల విశ్లేషకానికి సంబంధించిన మరిన్ని ముఖ్యమైన అంశాలను, వివిధ వాక్య నిర్మాణాలను విశ్లేషకం ఎలా విశ్లేషిస్తుందో వివరంగా భవిష్యత్తు పరిశోధనల్లో అధ్యయనం చేసి మీ ముందుకు తీసుకువచ్చే ప్రయత్నం జరుగుతుంది.

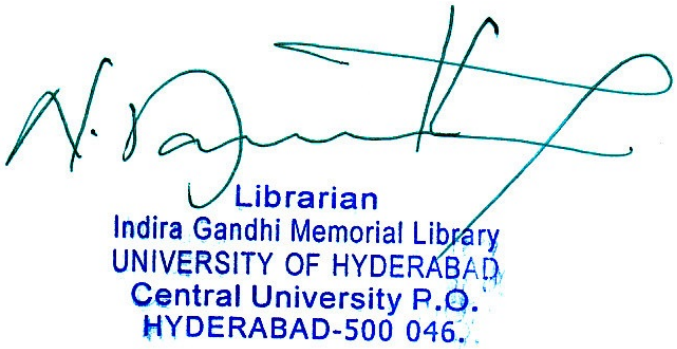
4. యంత్రానువాదంలో వాక్యవిశ్లేషకం యొక్క పాత్ర:

యంత్రానువాదంలో వాక్యవిశ్లేషకం కీలక పాత్ర వహిస్తుంది అనడంలో ఏలాంటి సందేహం లేదు. తెలుగు భాషలోని రచనలను ఇతర భాషలలోకి అనువాదం చేయడంలో తెలుగు వాక్యవిశ్లేషకం చాలా ముఖ్యమైనది. యంత్రానువాదంలో విశ్లేషకం ఒక విభాగంగా చేర్చబడింది. దీని సహాయంతో వాక్య నిర్మాణాన్ని క్షుణ్ణంగా విశ్లేషించి పద సంబంధ సమాచారం సేకరించవచ్చు. అలాంటి సమాచారం ఆ భాష అనువాద ప్రక్రియ జరిగేటప్పుడు చాలా అవసరం అవుతుంది. ముఖ్యంగా, ఒక భాషా కుటుంబం నుండి ఇతర భాషా కుటుంబాలకు అనువదించేటపుడు అందులో తలెత్తే వాక్య నిర్మాణ సందిగ్ధతని (structural ambiguity) తొలగించటంలో వాక్యవిశ్లేషకం చాలా సహాయపడుతుంది. ప్రస్తుత కాలంలో వాక్యవిశ్లేషకాలపై పరిశోధన పోటాపోటీగా సాగుతున్న తరుణంలో, చాలా సాంకేతిక సంస్థలు, భాష శాస్త్రవేత్తలు వీటి నిర్మాణాల్లో నిమగ్నమై ఉన్నారు. యూనివర్సల్ డిపెండెన్సీ (universal dependency) అనే సంస్థ వారు ముఖ్యమైన ప్రపంచ భాషలన్నింటిలోనూ వాక్యవిశ్లేషకాలను నిర్మించేందుకు టీకాసహిత (annotated corpus) పాఠాలను తయారుచేస్తున్నారు. తెలుగులో 1023 వాక్యాల టీకాసహిత పాఠాలను (annotated corpus) వారి జాలచోటులో (website) పొందుపరచడం జరిగింది.

రచయిత పరిశోధక విద్యార్థి
హైదరాబాదు విశ్వవిద్యాలయం,

A Rule-based Dependency Parser for Telugu

by P. Sangeetha



Submission date: 28-Dec-2022 12:21PM (UTC+0530)
Submission ID: 1987044715
File name: 18HAPH01.pdf (3.06M)
Word count: 45470
Character count: 236359

A Rule-based Dependency Parser for Telugu

ORIGINALITY REPORT

6%	5%	4%	1%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	icon2021.nits.ac.in Internet Source	2%
2	aclanthology.org Internet Source	1%
3	epdf.pub Internet Source	<1%
4	ep.liu.se Internet Source	<1%
5	Amba Kulkarni. "Sanskrit Parsing Following Indian Theories of Verbal Cognition", ACM Transactions on Asian and Low-Resource Language Information Processing, 2021 Publication	<1%
6	Marie-Catherine de Marneffe, Joakim Nivre. "Dependency Grammar", Annual Review of Linguistics, 2019 Publication	<1%
7	"Current Approaches to Syntax", Walter de Gruyter GmbH, 2019 Publication	<1%

8	Subhash C. Kak. "The Paninian approach to natural language processing", International Journal of Approximate Reasoning, 1987 Publication	<1 %
9	www.yumpu.com Internet Source	<1 %
10	"Non-nominative Subjects", John Benjamins Publishing Company, 2004 Publication	<1 %
11	ebin.pub Internet Source	<1 %
12	Submitted to Georgia Institute of Technology Main Campus Student Paper	<1 %
13	www.coursehero.com Internet Source	<1 %
14	www.tdil-dc.in Internet Source	<1 %
15	lml.bas.bg Internet Source	<1 %
16	www.pure.ed.ac.uk Internet Source	<1 %
17	core.ac.uk Internet Source	<1 %

18	Submitted to University of Hyderabad, Hyderabad Student Paper	<1 %
19	archive.org Internet Source	<1 %
20	citeseerx.ist.psu.edu Internet Source	<1 %
21	Submitted to Grand Canyon University Student Paper	<1 %
22	www.cs.uu.nl Internet Source	<1 %
23	Pooja Rai, Sanjay Chatterji. "Annotation Projection-based Dependency Parser Development for Nepali", ACM Transactions on Asian and Low-Resource Language Information Processing, 2022 Publication	<1 %
24	Submitted to Australian National University Student Paper	<1 %
25	Submitted to Napier University Student Paper	<1 %
26	ces.wu.ac.th Internet Source	<1 %
27	ldc-upenn.blogspot.com Internet Source	<1 %

28	Timothy Osborne. "A Dependency Grammar of English", John Benjamins Publishing Company, 2019 Publication	<1 %
29	George Karystianis, Kristina Thayer, Mary Wolfe, Guy Tsafnat. "Evaluation of a rule-based method for epidemiological document classification towards the automation of systematic reviews", Journal of Biomedical Informatics, 2017 Publication	<1 %
30	Guillaume Bonfante, Bruno Guillaume, Guy Perrier. "Dependency Syntax: Surface Structure and Deep Structure", Wiley, 2018 Publication	<1 %
31	www.sconli.org Internet Source	<1 %
32	Joakim Nivre. "Inductive Dependency Parsing", Springer Science and Business Media LLC, 2006 Publication	<1 %
33	T. Mala, T.V. Geetha .. "Story Summary Visualizer Using L Systems", Journal of Artificial Intelligence, 2007 Publication	<1 %
34	Text Speech and Language Technology, 2003. Publication	<1 %

35	www.ijoes.vidyapublications.com Internet Source	<1 %
36	"Mathematics of Language", John Benjamins Publishing Company, 1987 Publication	<1 %
37	erepo.uef.fi Internet Source	<1 %
38	tugantel.tatar Internet Source	<1 %
39	"The Languages and Linguistics of South Asia", Walter de Gruyter GmbH, 2016 Publication	<1 %
40	ipfs.io Internet Source	<1 %
41	"Discontinuous Constituency", Brill, 1987 Publication	<1 %
42	"Sentential Complementation", Walter de Gruyter GmbH, 1984 Publication	<1 %
43	Lecture Notes in Computer Science, 2015. Publication	<1 %
44	aclanthology.lst.uni-saarland.de Internet Source	<1 %
45	etd.astu.edu.et Internet Source	<1 %

46	Paolo Lorusso. "Lexical Parametrization and Early Subjects in L1 Italian", International Journal of Linguistics, 2018 Publication	<1 %
47	dl.icdst.org Internet Source	<1 %
48	Chaithra Puttaswamy. "Multi-verb constructions in Malto", Journal of South Asian Languages and Linguistics, 2018 Publication	<1 %

Exclude quotes	On	Exclude matches	< 14 words
Exclude bibliography	On		