ICON 2021

The 18th International Conference on Natural Language Processing

Proceedings of the Workshop on Parsing and its Applications for Indian Languages (PAIL)

December 16, 2021

Preface

A workshop on Parsing and its Applications for Indian Languages (PAIL-2021) was organized in conjunction with 18^{th} International Conference on Natural Language Processing held in NIT, Silchar, India. The workshop was organized virtually as a one-day event with two sessions. The first session included a keynote on "Computational Paninian Framework and Parsing Indian Languages" by Prof. Dipti Misra Sharma, followed by the technical sessions of the accepted papers. The afternoon session started with a keynote talk on "Parsing and its applications: Current Status and Future Perspectives" by Prof Marie-Catherine de Marneffe. The workshop was concluded with an insightful panel discussion on "Parsing and its applications: Current Status and Future Perspectives".

Annotated corpora are vital resources for deep learning application development and linguistic analyses. In the case of Indian languages, sufficient annotated quality data are not available publicly for researchers and developers to build upon. Although there are different levels of annotations, syntactically annotated corpora or treebank are very resourceful, especially for Indian languages, which are morphosyntactically rich. They are incorporated in downstream applications for various information extraction. Treebanks are being created for only a few Indian languages, and still, there is a high requirement for building more data in different domains while involving other languages. Indian languages do show language-specific complexities that require special attention. When a handful of languages representing different language families of India are ready with quality treebanks, those can be used to build resources for other languages using different approaches like transfer learning and multilingual learning. There is no way to start with or progress without annotated data; unsupervised approaches are not yet convincing enough, even for resourceful languages like English. Apart from all these justifications, we need to work together to make resources public and acceptable in a task like parsing and treebanking. Otherwise, we cannot create a meaningful and quality impact.

In this context, the PAIL-2021 workshop was organized with the following objects: (i) to bring researchers and developers together who work on treebanks, parsing, and related downstream natural language processing applications. (ii) to provide a platform for researchers to discuss Indian language-specific issues in the morphosyntactic analysis and (iii) to encourage researchers to collaborate and create more annotated resources

Keynote speeches

Two keynote speeches were arranged to introduce two widely used frameworks to annotate the treebanks of Indian languages, namely, the Paninian framework and the Universal Dependencies framework. It was fortunate to get professors who could speak on these topics as keynote speakers.

Professor Emeritus Dipti Misra Sharma from the International Institute of Information Technology (IIIT), Gachibowli, Hyderabad, has spoken on "Computational Paninian Framework and Parsing Indian Languages". Her talk started with the advantages of Paninian Dependency frameworks in encoding syntactic and semantic relations between words in a sentence, especially for free word order languages like Indian languages. She also touched upon different approaches used for dependency parsing, starting from constraint-based parsing to the current neural network-based approaches. She also focused on treebanks developed in different Indian languages using available dependency frameworks. She briefly discussed the conversion of Paninian dependencies to Universal dependencies as well.

Professor Marie-Catherine de Marneffe from the Department of Linguistics, The Ohio State University, delivered the second talk on the topic entitled "Universal Dependencies: the good, the bad and the potential". She shared how the Universal Dependencies framework evolved and its philosophy first. Then she covered in detail the important concepts and the Universality of the framework, and how it is used for various linguistics studies and the development of applications. Finally, she also set some directions for further improvements of the framework.

Technical sessions

Six papers were submitted for the PAIL-2021 workshop from India and Sri Lanka. Three experts reviewed each paper. Based on the outcomes, four papers were accepted for the workshop. While three of them covered the topic related to treebanking and parsing, one of them covered the application of treebanks.

The accepted papers covered a wide variety of topics. One paper was on Treebanking for extremely low resource languages Braj and Magahi, and two papers were on syntactic parsing of Tamil and non-finite clauses in Telugu using feature-based Malt parser and rule-based approaches, respectively. In addition, there was a paper on Tamil grammar detection that touched on parsing usage.

Panel discussion

A panel discussion was also arranged to discuss a topic entitled "Parsing and its applications: Current Status and Future Perspectives". The primary object of this panel was to disseminate knowledge on parsing and treebanking, and their importance and future. In addition, this was arranged to find out the challenges in creating relevant resources for Indian languages and the best practices. The following practitioners who participated in the panel discussion were well-experienced scholars in the area of parsing, tree-banking, and their applications:

- Professor Amba Kulkarni, Department for Sanskrit Studies, University of Hyderabad.
  Prof. Amba Kulkarni gave insights on parsing using Indian grammatical tradition and discussed the history of research in parsing Indian languages in general and Sanskrit in particular. She explained parsing complexities and the importance of including semantic information in parsing.
- Dr. Asif Ekbal, IIT Patna, India.
  Dr. Asif described the importance of parsing in various NLP applications and the optimization methods to integrate parsers in end-to-end neural networks methods. He further discussed in detail how to develop multilingual treebanks with current techniques.
- Dr. Dan Zeman, Institute of Formal and Applied Linguistics, Charles University, Czech Republic.
  Dr. Dan Zemen, a parsing expert in Universal Dependencies, (UD) discussed challenges in maintaining cross-lingual treebanks, mapping other treebanks with UD and building a new UD treebank. He explained the procedures of finding suitable texts with licence and the importance of documenting language-specific decisions.
- Dr. Ritesh Kumar, Dr Bhimrao Ambedkar University, India.
  Dr. Ritesh shared his experiences in treebank annotation for low-resourced languages such as Braj and Magahi. He also shared the language-specific issues that were encountered and how they are handled in treebank building. He iterates that building treebank is a long-term activity that requires understanding the language-specific features, guidelines and selecting the right annotation tool.

Acknowledgement: Organizers would like to thank everyone who supported us to organize the first-ever workshop on parsing and its applications for the Indian Languages. Specifically, the ICON-2021 organizers and the technical programme committee members need to be acknowledged for the workshop facilitation and support in reviewing papers, respectively.

PAIL-2021 Organisers:
Kengatharaiyer Sarveswaran, Department of Computer Science, University of Jaffna, Sri Lanka.
Parameswari Krishnamurthy, Centre for Applied Linguistics and Translation Studies, University of Hyderabad, India.
Pruthwik Mishra, MT & NLP Lab, LTRC, IIIT-Hyderabad, India.

Organizers

- Kengatharaiyer Sarveswaran, Department of Computer Science, University of Jaffna, Sri Lanka.
- Parameswari Krishnamurthy, University of Hyderabad, India.
- Pruthwik Mishra, MT & NLP Lab, LTRC, IIIT-Hyderabad, India.

Technical Programme Committee

- Amba Kulkarni, University of Hyderabad, India
- Anand M Kumar, National Institute of Technology - Surathkal, India
- Ashwath Rao, MIT - Manipal, India
- Asif Ekbal, IIT Patna, India
- Braja Gopal Patra, Weill Cornell Medicine, USA
- Dhanalakshmi V, RV Government Arts college, India
- Asif Ekbal, IIT Patna, India
- Gihan Dias, University of Moratuwa, Sri Lanka
- Govindaru V, C-DIT, Thiruvanathapuram, India
- Irshad Ahmad Bhat, Active Intelligence LLP, India
- Malhar A Kulkarni, Indian Institute of Technology Bombay, India
- Muralikrishna SN, MIT - Manipal, India
- Ritesh Kumar, Dr Bhimrao Ambedkar University, India
- S Mahesan, University of Jaffna, Sri Lanka
- Rajendran Sankaravelayuthan, Amrita Vishwa Vidyapeetham, India
- Samar Hussain, IIT Delhi, India
- Sowmya Vajjala, National Research Council, Canada
- Surangika Ranathunga, University of Moratuwa, Sri Lanka
- Taraka Rama, University of North Texas, USA
- Uthayasanker Thayasivam, University of Moratuwa, Sri Lanka

Table of Contents

Workshop Program

Thursday, December 16, 2021 - 09:30 to 16:50 (IST)

09:30–09:45 ***Morning sessions - Inaugural address***

09:45–10:45 *Keynote speech: Computational Paninian Framework and Parsing Indian Languages*
Professor Dipti Misra Sharma

10:45–11:15 *Developing Universal Dependencies Treebanks for Magahi and Braj*
Mohit Raj, Shyam Ratan, Deepak Alok, Ritesh Kumar and Atul Kr. Ojha

11:15–11:45 *Parsing Subordinate Clauses in Telugu using Rule-based Dependency Parser*
P Sangeetha, Parameswari Krishnamurthy and Amba Kulkarni

11:45–12:15 *Dependency Parsing in a Morphological rich language, Tamil*
Vijay Sundar Ram and Sobha Lalitha Devi

12:15–12:35 *Neural-based Tamil Grammar Error Detection*
Dineskumar Murugesapillai, Anankan Ravinthirarasa, Gihan Dias and Kengatharaiyer Sarveswaran

14:00–14:10 ***Evening sessions - Welcome address***

14:10–15:10 *Keynote speech: Universal Dependencies: the good, the bad and the potential*
Professor Marie-Catherine de Marneffe, Department of Linguistics, The Ohio State University.

15:10–16:45 *Panel discussion: Parsing and its applications: Current Status and Future Perspectives*
Prof. Amba Kulkarni, Dr. Asif Ekbal, Dr. Dan Zeman, and Dr. Ritesh Kumar

Developing Universal Dependency Treebanks for Magahi and Braj

Mohit Raj[!], **Shyam Ratan**[!], **Deepak Alok**[+], **Ritesh Kumar**[!], **Atul Kr. Ojha**[*,+]

[!]Dr. Bhimrao Ambedkar University, Agra [+]Panlingua Language Processing LLP, New Delhi

[*]Data Science Institute, National University of Ireland Galway, Ireland

mohiitraj@gmail.com shyamratan2907@gmail.com deepak06alok@gmail.com riteshkr.kmi@gmail.com panlingua@outlook.com

Abstract

In this paper, we discuss the development of treebanks for two low-resourced Indian languages - Magahi and Braj - based on the Universal Dependencies framework. The Magahi treebank contains 945 sentences and Braj treebank around 500 sentences marked with their lemmas, part-of-speech, morphological features and universal dependencies. This paper gives a description of the different dependency relationship found in the two languages and give some statistics of the two treebanks. The dataset will be made publicly available on Universal Dependency (UD) repository (https://github.com/UniversalDependencies/UD_Magahi-MGTB/tree/master) in the next (v2.10) release.

1 Introduction

Magahi is an Eastern Indo-Aryan Language, spoken mainly in Eastern Indian states including Bihar and Jharkhand, along with some parts of West Bengal and Odisha. Magahi is classified under the Eastern group of the outer sub-branch of Indo-Aryan language (Grierson, 1908). Scholars like Turner have clubbed the 'Bihari' languages with Eastern and Western Hindi (Masica, 1991). There is another kind of classfication of Indian language in which western Hindi is almost an isolated group while Eastern Hindi, Bihari and other languages of Eastern group are clubbed together (Chatterji, 1926). But the classification in which Magahi comes under the Eastern group of the outer sub-branch of Indo-Aryan language is the most widely accepted classification.

Brajbhasha is classified in two ways. According to first classification, Brajbhasha is a Western Indo-Aryan language that is spoken in the states of Western Uttar Pradesh and parts of Rajasthan (Jeffers, 1976). The other classification puts Brajbhasha in the group of Western Hindi of Central Group of Indo-Aryan sub-family of Indo European language family along with Hindustani, Bangaru, Brajbhaka, Kanauji, Bundeli (Grierson, 1908).

The difficulty of tracing the exact historical path of a large number of these Indo-Aryan languages is discussed in detail by Masica (Masica, 1991) and is evident by somewhat incompatible classification given by Chatterji, Turner, Katre, Cardona and Mitra and several other scholar (Masica, 1991);(Chatterji, 1926); (Turner, 1966); (Katre, 1968); (Cardona, 1974); (Mitra et al., 1978). As such the exact status of Magahi and Braj vis-a-vis other Indo-Aryan languages (especially the major ones like Hindi, Bangla and Odia) remains hazy and controversial. This, coupled with the imposition of Modern Standard Hindi (MSH) over what is now popularly known as 'Hindi Belt' and what has historically been established as a rather complex dialect continuum, with several languages and varieties being spoken in different domains of usage (Gumperz, 1957), has resulted in these languages mistakenly classified as varieties of Hindi. This, in turn, has resulted in not only minimal support from the Government for development of different kinds of resources for the language but also a negative attitude of the speakers towards the language (Kumar et al., 2018a).

However, despite this disadvantageous situation of the language, there has been some efforts at developing language technologies and resources for these languages, especially for Magahi viz. monolingual written and speech

Proceedings of the ICON 2021 Workshop on Parsing and its Applications for Indian Languages (PAIL) , pages 1–11
December 16, 2021. Silchar, India.

corpora (Kumar et al., 2014), Magahi part-of-speech tagger (Kumar et al., 2012) and Magahi language identification system (Rani et al., 2018). This paper describes another such effort towards developing a Universal Dependency (UD) based treebank for the language which may prove to be useful in processing and analysing the language for different applications.

2 Universal Dependencies and Low-resource Languages

Universal Dependencies framework provides some unique advantages for low-resource languages both in terms of making the language for cross-lingual comparison and studies as well as making transfer learning and multilingual techniques for technology development possible. As a result in recent times we have seen the development of UD treebanks for quite a few low-resource languages viz. Yorùbá (Ishola and Zeman, 2020), Latin Treebank for UD (Cecchini et al., 2020), Hittite (Andersen and Rozonoyer, 2020), Manx Gaelic (Scannell, 2020), Laz (Tu̯rk et al., 2020), Albanian (Toska et al., 2020) and others.

The UD treebanks have also been built for Indian languages such as Bhojpuri, Hindi, Marathi, Sanskrit, Tamil,Telugu and Urdu (Zeman and et al., 2021; Ojha and Zeman, 2020). Except Hindi, all of the (Indian) languages mentioned above are low-resourced languages. Recently, Dash et al. (2021) reported the development of a treebank in Santhali, another low-resourced language spoken in India.

However, one of the biggest challenges in building treebanks for a large number of low-resource languages is the absence of grammatical descriptions and hence a reference point for deciding on the analysis needed to give the dependency relationships. There are many treebanks that could be roughly classified in two groups. Treebanks of well-known and well-described languages like Hindi, English, French etc and treebanks of lesser known and sparsely described languages. Magahi and Braj come in the second group which are sparsely described languages. There have been very few linguistic studies on these languages with both of these languages lacking an exhaustive grammatical description or even a dictionary. These are some linguistics studies towards the description of Magahi: a basic (although not completely accurate) description of Magahi is given by Shila Verma (Verma and Verma, 1983; Verma, 1985), a description of Magahi case system (Lahiri, 2021, 2014; Kumar et al., 2014), discussion on Magahi honorific system within the minimalist framework (Alok, 2021), morpho-syntactic properties of nominal particle *-wa* (Alok, 2014), and study on the Magahi spatial postpositions (Alok, 2012).

For Braj, to the best of our knowledge, the only modern linguistic studies are on its ergativity under the minimalist framework (Chandra and Kaur, 2020a,b). .

This lack of an exhaustive description of different aspects of Magahi and Braj morphosyntax made the task of developing the treebank quite challenging and required establishing multiple grammatical analyses of the languages while working on the treebank. The aim of this paper is to give a broad description of Magahi and Braj morphosyntax within the Universal Dependencies framework with respect to different syntactic dependency relationships in the languages, along with a discussion on the process of the development of this treebank.

3 Treebank Creation in Magahi and Braj

We annotate the Magahi and Braj treebank with lemma, Universal Parts-of-Speech (POS) tags, a subset of morphological features and the Universal dependency relation. The dependency relation for Magahi and Braj are annotated using a subset of the 37 dependency relations included in the Universal Dependency tagset[1]. In the following sections, we describe each of the features marked for building Magahi and Braj dependency treebank. Most of these relationships follow the canonical patterns as discussed in the UD guidelines (and as witnessed in common Indo-European languages).

[1] https://universaldependencies.org/u/dep/

3.1 POS and Morphological Features

We use the Universal POS tags for annotating the POS tags for the data. However, for morphological features we mark only those features which have explicit morphological realisation in the two languages. Thus, for example, we mark gender and number on Braj verbs but not on Magahi verbs since there is no number or gender agreement in Magahi. Table 1 gives a list of all the morphological features and their values that we mark for each category of words in each of the language [2].

3.2 Core Dependency arguments

3.2.1 Nominal Subject : nsubj

Nominal subject in Magahi and Braj plays the role of syntactic subject and it is dependent on the verb. Let us take a look at the Figure 1 and 2.

- राजा लाल लेके रानी के दे देलन ।

  rɑʤɑː lɑl leke rɑniː ke ḍe ḍelən .

  King took the perl and gave to the queen.

- मैं एक गाँव की पाठशाला में कक्षा दूसरी कूँ पढ़ा रह्यौ ।

  maiⁿ ek gɑⁿv kiː pɑʈʰʃɑlɑː meⁿ kəkʂɑː ḍusriː kuːⁿ pəɖʰəːɑː rəɦjaːu .

  I was teaching class second in a village school.

3.2.2 Object : obj

Object of a sentence in the two languages is also dependent on the verb. Let us take a look at the Figure 1 and 2.

3.2.3 Indirect Object : iobj

The role and nature of indirect objects is similar to direct objects except the syntactic closeness with verbs. Indirect objects are also dependent on verbs and this relationship is indicated by iobj. Let us take a look at the Figure 1 and 2.

[2]In the table * indicate that the given feature or value is marked only for Braj

3.2.4 Clausal complement : ccomp

Clausal complement occurs with complex structure of sentence in Magahi and Braj. When the sentence is formed with two clauses (principle and subordinate), the root (generally, verb) of the subordinate clause depends on the root of the principal clause. Subordinate clause behaves like an object of the main clause.

3.2.5 Open clausal complement : xcomp

Open clausal complement differs from the clausal complement in that in this case the head of the subordinate clause does not seem to have an overt subject and as such there is a dependence relation between the root of the subordinate clause and a word of the higher clause of Magahi and Braj.

3.3 Non-core dependents

3.3.1 Oblique Nominal : obl

Oblique is the nominal element of a sentence which appears as an adjunctive argument and it depends upon the main verb of the sentence. Sometimes it adds extra information about a verb, adjective and adverb so it functionally acts as an adverbial attachment. Oblique is grammatically categorised as a noun or pronoun and used as a temporal and nominal locational modifier as in the given example of Magahi and passive agent are also labelled as oblique. In another example from Braj, the adverbial modifier also bears an obl relation with the verb.

3.3.2 Adverbial clause modifier : advcl

Adverbial clause modifier is an entity of complex structural sentences in which it is the main predicate of a dependent clause. Like an adverb, its functional role is to modify a verb or other predicate such as an adjective of principal clause or other clausal entity, but the difference lies in the fact that adverbial clause modifier establishes an interclausal relationship.

3.3.3 Adverbial modifier: advmod

Adverbial modifier defines the relationship in between main verb of a clause and the adverb. In the given example of Magahi word खुब'/'kʰubə' is modifying the main verb or root नेचबवऽ'/'nəcəbəvə' of sentence. Another

Feature	UPOS	Values
Case	NOUN, PRON, ADP	Nom, Acc, Dat, Gen, Erg*, Abl
Gender*	NOUN, PRON, VERB, PROPN,	Fem, Masc
Number*	NOUN, PRON, VERB	Sing, Plur
Person	PRON, VERB	1,2,3
Tense	VERB, AUX	Pres, Past, Fut*
Aspect	VERB, AUX	Prog, Imp, Perf, Hab
Politeness	PRON, VERB, AUX	Form, Infm

Table 1: Feature values for Magahi & Braj grouped by UPOS

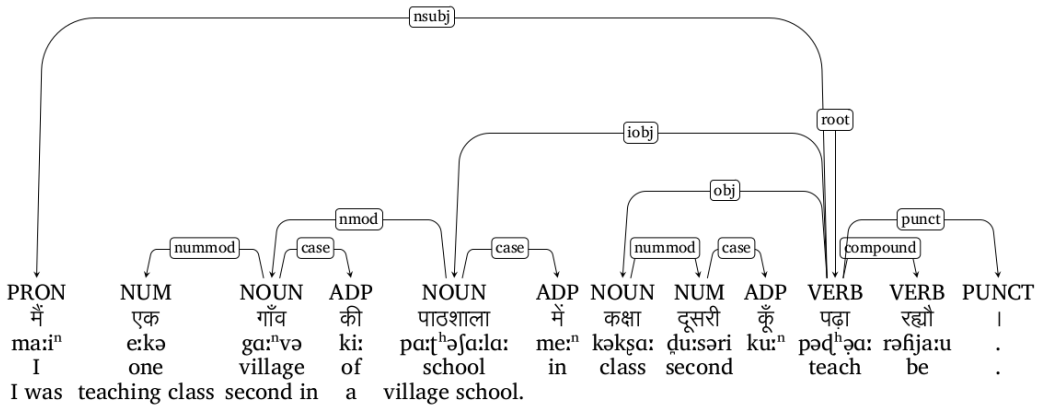

Figure 1: Braj Example for nsubj, obj, iobj

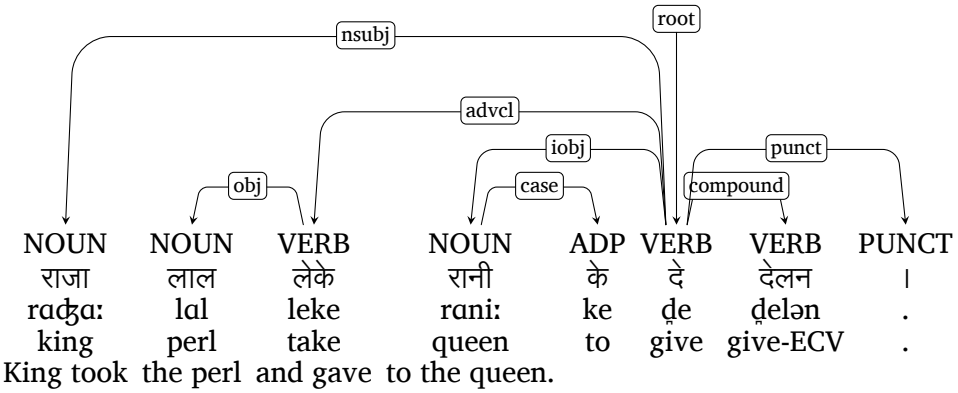

Figure 2: Magahi Example for nsubj, iobj, obj

kind of adverbial modifier is given in the Braj example which is negating the act of event. Let us take a look at the Figure 4 and 3.

- आउ हम खुब नचबवऽ ।

  ɑːu ɦəm kʰub nəcbəvə .

  And I will dance a lot.

- पै बूआ कहीं नाँय जाती।

  pai buɑː kəɦiːⁿ nɑːⁿj ʤaːʈiː .

  But the aunt did not go anywhere.

3.3.4 Auxiliary : aux

aux is the dependency relation between a verbal predicate and the auxiliary in the two languages. Let us take a look at the Figure 5.

- या कारन मैं चुप्प खींच गयौ हो ।

  jɑː kɑːrən maiⁿ cuppə kʰiːⁿc gəjaːu ɦo .

  For this reason I remained silent.

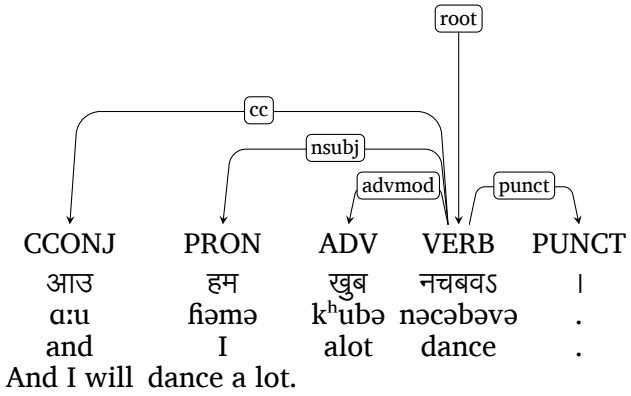

Figure 3: Magahi Example for advmod

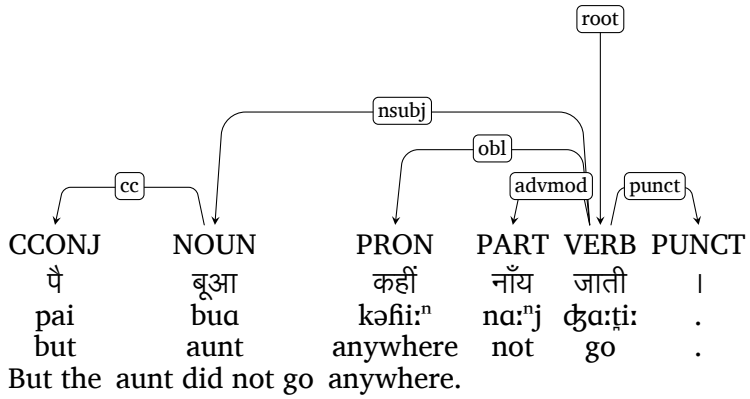

Figure 4: Braj Example for advmod

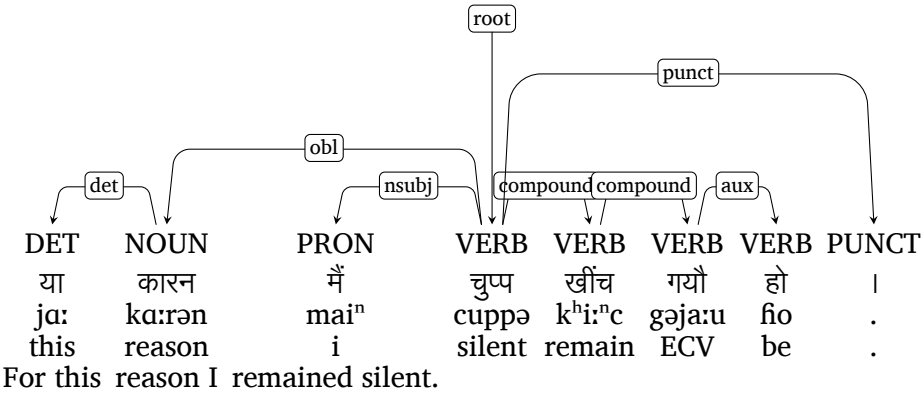

Figure 5: Braj Example for aux

3.3.5 Copula : cop

aux is the dependency relation between a non-verbal predicate and the auxiliary in the two languages. Let us take a look at the Figure 9 and 6.

- सीर्सक हौ " गोबिंद बढई " ।

  siːrsək ɦau “ gobiⁿd̪ə bəɖʰəiː ” .

  Title was Govind carpenter.

We found two type of copula in Braj (simple and complex). Simple copula (हतौ/ɦət̪ɒː, है/ɦai, ही/ɦiː) are like copula of Magahi, and complex copula (नांओ᳾/‘nɑːⁿɒ’, आओ᳾/‘ɑːɒ’, गओ᳾/‘gəɒ’) are not present in Magahi.

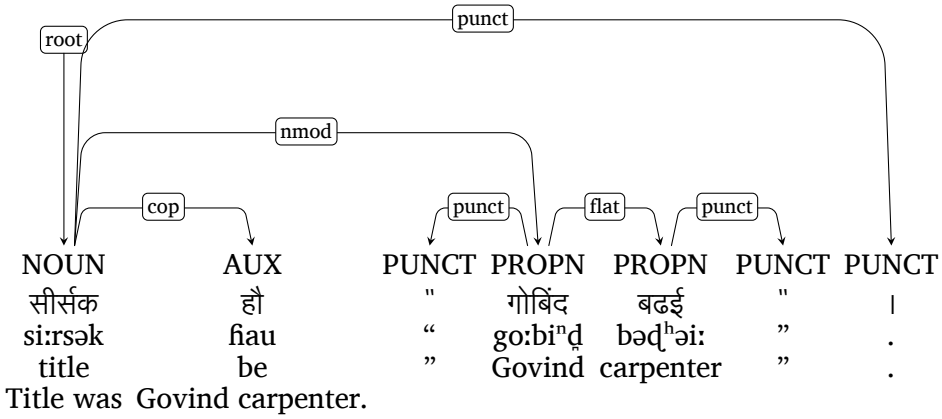

Figure 6: Braj Example for cop

3.3.6 Marker : mark

Typically the subordinating conjunction depends on the head of the subordinate clause - such dependency relationships are called mark.

3.4 Nominal dependents

3.4.1 Nominal modifier : nmod

The nominal modifier is noun or pronoun and it is syntactically dependent upon another noun or noun phrase and functionally corresponds to an attribute or genitive complement.

3.4.2 Numeral modifier : nummod

It is the relationship between a numeral and the noun which is attached to the numeral. Let us take a look at the Figure 1.

3.4.3 Clausal modifier of noun : acl

In the acl syntactic relation head is noun that is modified and the dependent is the head of the clause that modifies the noun.

3.4.4 Adjectival modifier : amod

amod defines the dependency relationship between the noun and the adjective.

3.4.5 Determiner : det

The relation between determiner and nominal head is annotate with syntactic relation det. Let us take a look at the Figure 7.

- मैं जा बहली में बैठ बू पलट गई ।

  mai^{n} ʤa: bəfiəli: me^{n} baiʈ^{h} bu: pələʈ gəi: .

  The bullock cart I was sitting in overturned.

3.4.6 Classifier : clf

Numeral classifiers are used only in Magahi and clf defines the relation between the numeral and the classifier. Let us take a look at the Figure 8.

- इधर राजा सात गो बियाह कर लेलन ।

  iḍ^{h}ərə rɑʤa: sɑ:ʈ go bijɑ:fi kər lelən .

  Meanwhile king has married to seven girl.

3.4.7 Case marker : case

Postpositions are the case marking elements in both Braj and Magahi. The postpositions are syntactically dependent on the noun to which they attach and are related by the 'case' relation.

3.5 Coordination

3.5.1 Conjunct : conj

In case of coordinating conjunction construction, the head of the second clause depends upon the head of the first clause and are related by the 'conj' relation.

3.5.2 Coordinating Conjunction : cc

The coordinating conjunction itself is dependent upon the head word of the second clause or phrase and have a 'cc' relation. Let us take a look at the Figure 10 and 9.

- जैन बड़ी मजबुत आउ बहादुर हल ।

  ʤain bəḍ̤i: məʤbuʈ ɑ:u bəfiɑ:ḍ̤ur fiəl .

  Jain was very strong and brave.

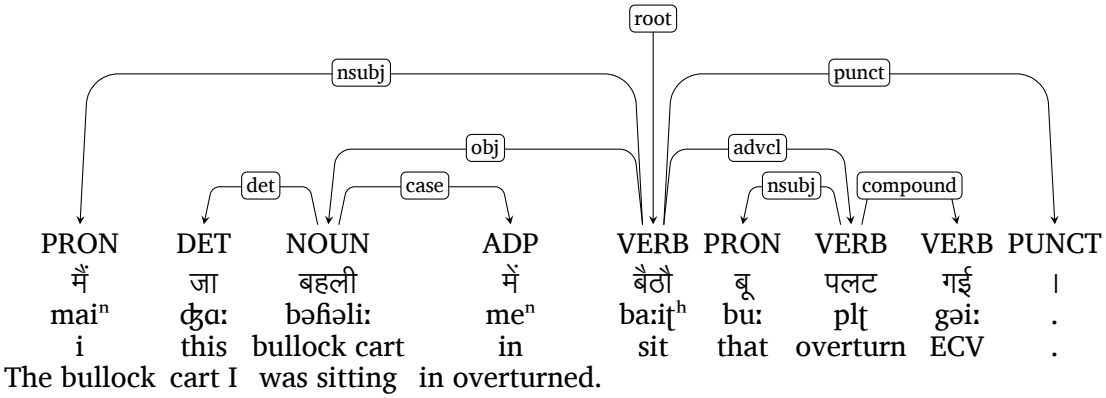

Figure 7: Braj Example for det

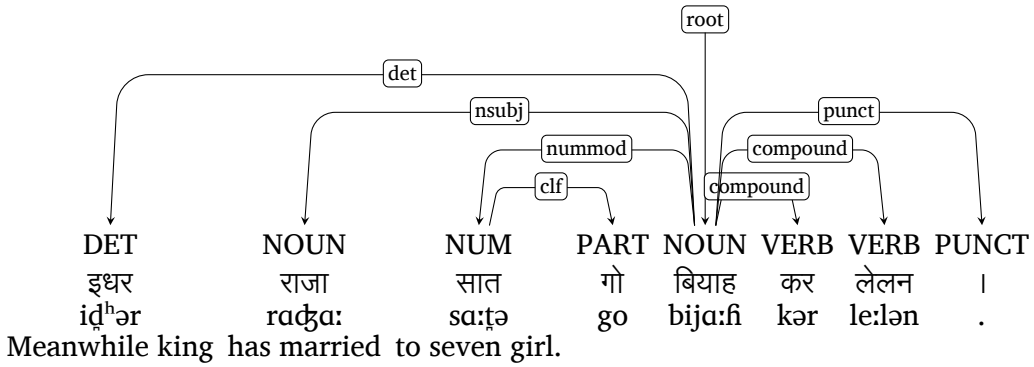

Figure 8: Magahi Example for clf

- झारू बुहारी कपड़ा – लत्ता और रसौई ।
- ʤʰaːruː buɦaːriː kəpəɖ̤əː - ləṯ̤ːtaː aur rəsauiː .

  Broom, cloth and kitchen.

3.6 Multi Word Expressions

3.6.1 Flat multiword expression : flat

The flat is a syntactic relation that is used for such multiword expression in which there is no specific head word like in name(Sohan Lal Mishra) or dates(2nd B.C.) etc.

3.6.2 compound : compound

A compound syntactic relation is used for different kinds of multiword expressions in Magahi and Braj including compound nouns, compound verbs, conjunct verbs (noun/adjective+verb), echo words, and reduplication. While most of these constructions are quite predominant in almost all of the South Asian languages and could actually have different kinds of syntactic and semantic impacts, unfortunately UD provides very limited ways of distinguishing across these and it proved to be one of the most challenging aspects of building this treebank.

4 Magahi and Braj Treebank

We have used Magahi and Braj plain text to prepare a treebank. Magahi plain text is part of a large monolingual written and speech corpora (Kumar et al., 2014), which is prepared from the Magahi literature. Braj plain text is also prepared from the literary domain (Kumar et al., 2018b). We have used Conllueditor (Heinecke, 2019) tool to build a treebank. The tool facilitate us to attach several kinds of information with words like UPOS, lemma, morph feature and dependency relation among different word of sentence etc.

Currently, Braj treebank has a total of around 5.8k tokens (excluding punctuations) in a total of 500 sentences. Magahi, on the other hand, has a total of over 12k tokens

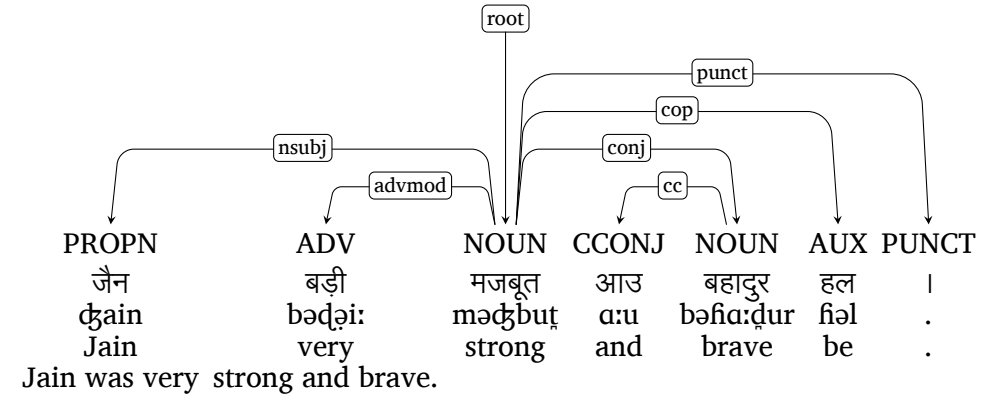

Figure 9: Magahi Example for cc and cop

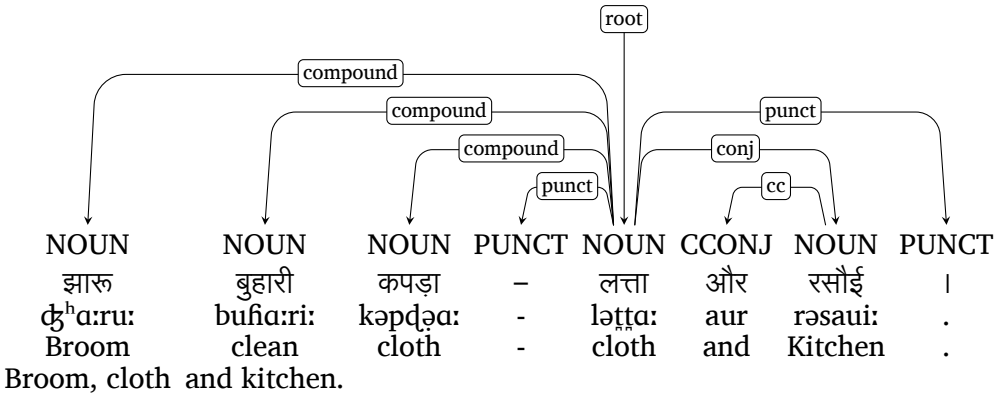

Figure 10: Braj Example for cc

UPOS	Braj Count	Braj %	Magahi Count	Magahi %
NOUN	1507	23.17 %	3203	24.01 %
VERB	1203	18.49 %	3040	22.78 %
PART	224	3.44 %	282	2.11 %
PRON	562	8.64 %	1172	8.78 %
CCONJ	113	1.73 %	330	2.47 %
ADV	83	1.27 %	273	2.05 %
PROPN	246	3.78 %	249	1.87 %
ADP	778	11.96 %	1681	12.60 %
SCONJ	112	1.72 %	418	3.13 %
NUM	253	3.89 %	356	2.67 %
ADJ	276	4.24 %	168	1.26 %
DET	181	2.78 %	385	2.89 %
AUX	230	3.53 %	440	3.30 %
INTJ	0	0 %	15	0.11 %
PUNCT	842	12.94 %	1331	9.98 %
TOTAL	**6,610**	**100 %**	**13,343**	**100 %**

Table 2: Braj & Magahi UPOS Category Statistics

from 945 sentences. Table 2 and Table 3 gives the detailed statistics of each UPOS and UD category and morphological features in the two treebanks. As expected, nouns and verbs form the most predominant POS categories in both the languages, both of them together ac-

Dependency Relation	Braj Count	Braj %	Magahi Count	Magahi %
root	500	7.41 %	945	7.01%
obj	444	6.58 %	808	5.99%
nmod	547	8.11 %	531	3.94%
nsubj	578	8.57 %	1065	7.90%
obl	106	1.57 %	580	4.30%
advmod	169	2.50 %	419	3.11%
cc	113	1.67 %	313	2.32%
case	771	11.44 %	1861	13.81%
conj	516	7.65 %	320	2.37%
mark	126	1.86 %	419	3.11%
advcl	244	3.62 %	748	5.55%
nummod	117	1.73 %	293	2.17%
iobj	188	2.78 %	851	6.31%
det	172	2.55 %	338	2.50%
amod	211	3.11 %	166	1.23%
xcomp	28	0.41 %	76	0.56%
aux	146	2.16 %	296	2.19%
cop	96	1.42 %	126	0.93%
compound	577	8.56 %	1327	9.85%
dep	99	1.46 %	163	1.20%
ccomp	13	0.19 %	312	2.31%
acl	3	0.044 %	54	0.40%
flat	122	1.81 %	124	0.92%
clf	0	0 %	6	0.04%
punct	842	12.49 %	1331	9.87%

Table 3: Braj & Magahi Dependency Relation Statistics

counting for over 40% of the tokens. These are followed by adpositions and pronouns as the most frequent category of words in the treebank.

5 Conclusion

In this paper, we have discussed the development of treebanks for Braj and Magahi - two extremely low-resource Eastern Indo-Aryan languages spoken in India. The treebank is annotated with lemma, UPOS, morphological features and UD relations. As of now the Braj treebank has 500 sentences (with around 5.8k tokens) while the Magahi treebank has 945 sentences (with over 12k tokens). A comparative analysis of dependency relation of Magahi and Braj treebank reveal that, as expected, most of the syntactic relations are shared across the two languages except the two dependency relation in our available dataset. The first one is copula (cop) relation - Magahi has a simple syntactic structure of cop while in Braj it could have a complex structure and it may vary between two types of lexical structure. The second dependency relation is that of classifier (clf) - Magahi has numeral classifier while this is not present in Braj.

6 Future Work

The analysis of the two languages as well as the development of the treebank is currently in progress - we are exploring the semi-automatic means of further increasing the treebank size such as developing and using parsers for annotating the data and then manually validating it. We are also exploring ways of getting data from varied domains (including narrations and conversational data) for including in the treebank.

Acknowledgments

We would like to express our heartfelt gratitude to Panlingua Language Processing LLP for supporting the creation of these treebanks both academically and financially. We would also like to thank Dr. Mayank for helping us out in figuring out certain aspects of the Braj grammar and annotation.
Atul Kr. Ojha would like to acknowledge the EU's Horizon 2020 Research and Innovation programme through the ELEXIS project under grant agreement No. 731015.

References

Deepak Alok. 2012. A language without articles: the case of magahi.

Deepak Alok. 2014. The morpho-synatx of nominal particle-wa. *Indian Linguistics*, 75:39–44.

Deepak Alok. 2021. The morphosyntax of magahi addressee agreement. *Syntax*, 24(3):263–296.

Erik Andersen and Benjamin Rozonoyer. 2020. A small Universal Dependencies treebank for Hittite. In *Proceedings of the Fourth Workshop on Universal Dependencies (UDW 2020)*, pages 1–7, Barcelona, Spain (Online). Association for Computational Linguistics.

George Cardona. 1974. *The Indo-Aryan languages*.

Flavio Massimiliano Cecchini, Timo Korkiakangas, and Marco Passarotti. 2020. A new Latin treebank for Universal Dependencies: Charters between Ancient Latin and romance languages. In *Proceedings of The 12th Language Resources and Evaluation Conference*, pages 933–942, Marseille, France. European Language Resources Association.

Pritha Chandra and Gurmeet Kaur. 2020a. Braj in the ergativity hierarchy. In *Formal Approaches to South Asian Languages*.

Pritha Chandra and Gurmeet Kaur. 2020b. Macro differences in dialects. *University of Pennsylvania Working Papers in Linguistics*, 25:Article 8.

Suniti Kumar Chatterji. 1926. *The origin and development of the Bengali language*. Calcutta University Press, Calcutta, India.

Satya Dash, Sunil Sahoo, Brojo Kishore Mishra, Shantipriya Parida, Jatindra Nath Besra, and Atul Kr Ojha. 2021. Universal dependency treebank for santali language. *SPAST Abstracts*, 1(01).

George A. Grierson. 1908. *Indo-Aryan Family: Central Group: Specimens of the Rājasthānī and Gujarātī*, volume IX(II) of *Linguistic Survey of India*. Office of the Superintendent of Government Printing, Calcutta.

John J. Gumperz. 1957. Language problems in the rural development of north india. *The Journal of Asian Studies*, 16:251–259.

Johannes Heinecke. 2019. ConlluEditor: a fully graphical editor for Universal dependencies treebank files. In *Universal Dependencies Workshop 2019*, Paris.

Olájídé Ishola and Daniel Zeman. 2020. Yorùbá dependency treebank (YTB). In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 5178–5186, Marseille, France. European Language Resources Association.

Robert J. Jeffers. 1976. The position of the bihārī dialects in indo-aryan. *Indo-Iranian Journal*, 18:215–225.

S M Katre. 1968. *Problems of reconstruction in Indo-Aryan*. Indian Institute of Advanced study, Shimla, India.

Ritesh Kumar, Bornini Lahiri, and Deepak Alok. 2012. Developing a POS tagger for Magahi: A comparative study. In *Proceedings of the 10th Workshop on Asian Language Resources*, pages 105–114, Mumbai, India. The COLING 2012 Organizing Committee.

Ritesh Kumar, Bornini Lahiri, and Deepak Alok. 2014. Developing lrs for non-scheduled indian languages. pages 491–501.

Ritesh Kumar, Bornini Lahiri, and Deepak Alok. 2018a. Descriptive study of eastern hindi: A mixed language. *Linguistic Ecology of Bihar*.

Ritesh Kumar, Bornini Lahiri, Deepak Alok, Atul Kr. Ojha, Mayank Jain, Abdul Basit, and Yogesh Dawar. 2018b. Automatic identification of closely-related indian languages: Resources and experiments. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Paris, France. European Language Resources Association (ELRA).

Bornini Lahiri. 2014. *A typological study of cases in eastern indo-aryan language*. Ph.D. thesis.

Bornini Lahiri. 2021. *The Case System of Eastern Indo-Aryan Languages: A Typological Overview*, 1 edition. Routledge India.

C.P. Masica. 1991. *The Indo-Aryan Languages*. Cambridge Language Surveys. Cambridge University Press.

Asok Mitra, Raj Nigam, and Sukumar Sen. 1978. Grammatical sketches of Indian languages with comparative vocabulary and texts. Number v. 1 in Census of India ... language monograph.

Atul Kr. Ojha 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, Marseille, France. European Language Resources Association (ELRA).

Priya Rani, Atul Kr. Ojha, and Girish Nath Jha. 2018. Automatic language identification system for hindi and magahi. *CoRR*, abs/1804.05095.

Kevin Scannell. 2020. Universal Dependencies for Manx Gaelic. In *Proceedings of the Fourth Workshop on Universal Dependencies (UDW 2020)*, pages 152–157, Barcelona, Spain (Online). Association for Computational Linguistics.

Marsida Toska, Joakim Nivre, and Daniel Zeman. 2020. Universal Dependencies for Albanian. In *Proceedings of the Fourth Workshop on Universal Dependencies (UDW 2020)*, pages 178–188, Barcelona, Spain (Online). Association for Computational Linguistics.

Utku Tụrk, Kaan Bayar, Ayşegụl Dilara É¨zercan, Gọrkem Yiğit É¨ztụrk, and Şaziye Betụl É¨zateş. 2020. First steps towards Universal Dependencies for Laz. In *Proceedings of the Fourth Workshop on Universal Dependencies (UDW 2020)*, pages 189–194, Barcelona, Spain (Online). Association for Computational Linguistics.

Ralph L. Turner. 1966. *A comparative dictionary of the Indo-Aryan languages (CDIA L)*, volume 3. Oxford University Press, London.

Sheela Verma. 1985. *The Structure of the Magahi Verb*. New Delhi:Manohar.

Sheela Verma and Manindra K. Verma. 1983. *The auxiliary with special reference to Magahi*, volume 44. Pune.

Daniel Zeman and et al. 2021. Universal dependencies 2.8.1. LINDAT/CLARIAH-CZ digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

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.

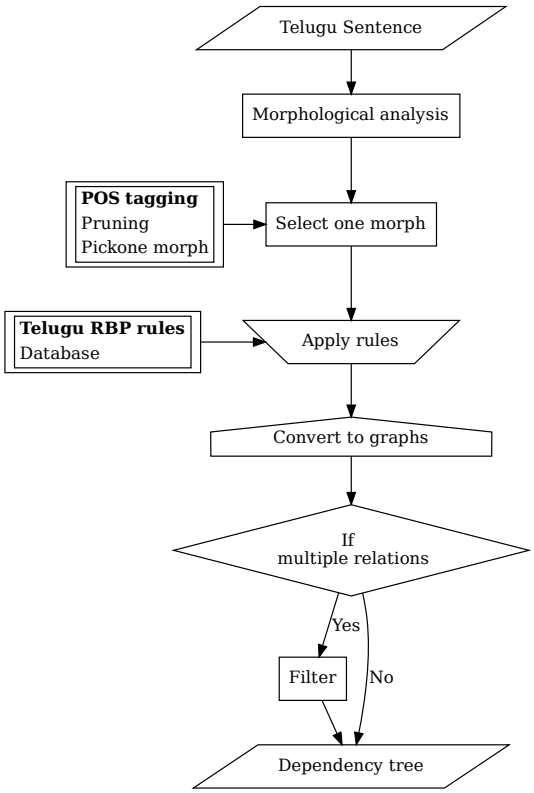

Figure 1: Architecture

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

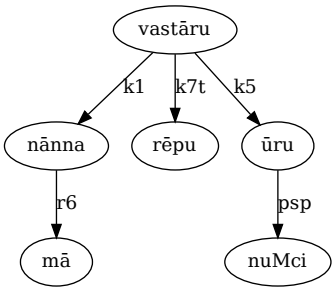

Figure 2: Dependency tree for sentence(1)

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.

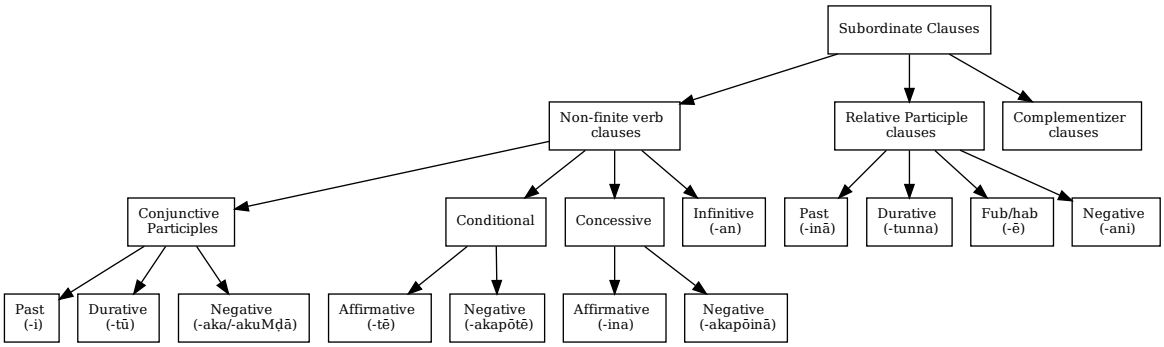

Figure 3: Types of non-finite clauses in Telugu

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'

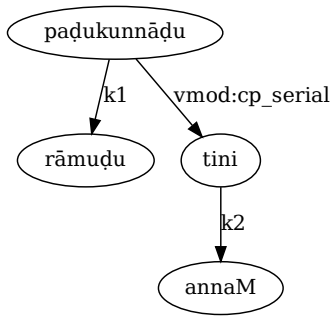

Figure 4: Dependency tree for (2)

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).

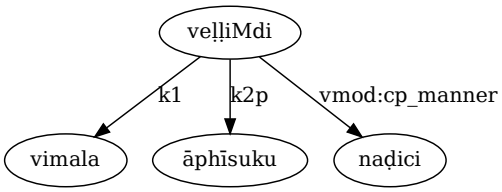

Figure 5: Dependency tree for (3)

(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`.

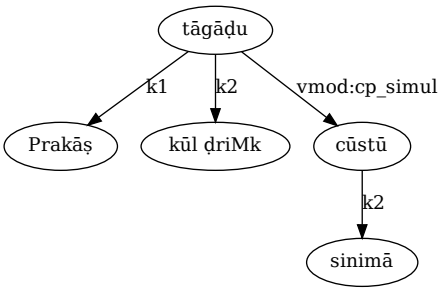

Figure 6: Dependency tree for (4)

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'

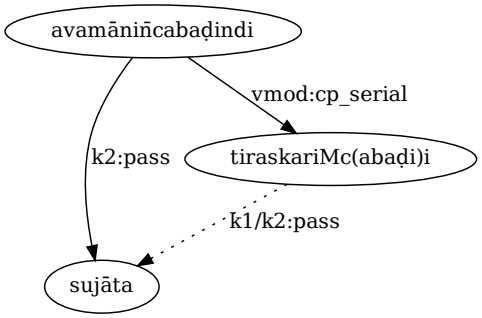

Figure 7: Dependency tree for (5)

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'

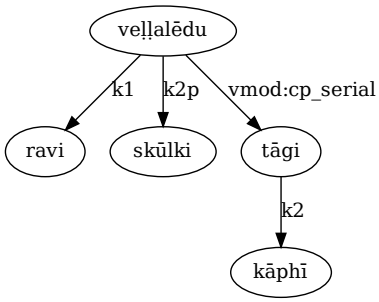

Figure 8: Dependency tree for (8)

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'

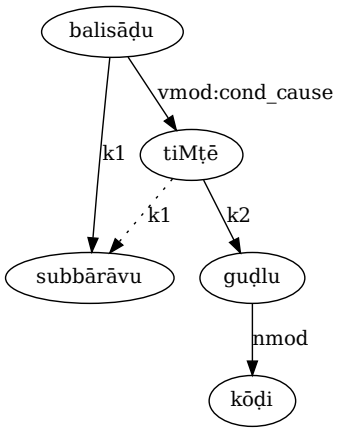

Figure 9: Dependency tree for (12)

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.

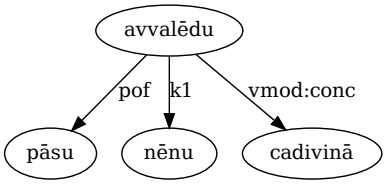

Figure 10: Dependency tree for (13)

(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'

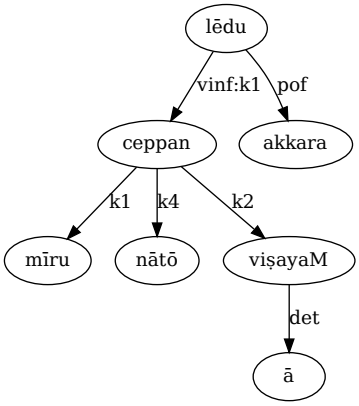

Figure 11: Dependency tree for (14)

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 manisi 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 manisi iMṭi-ki
I-ACC see-RP.PST man home-DAT
vacc-āḍu
come-PST-3.SG.M
'The man who saw me came home'

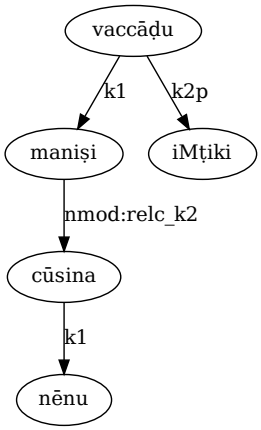

Figure 12: Dependency tree for (15)

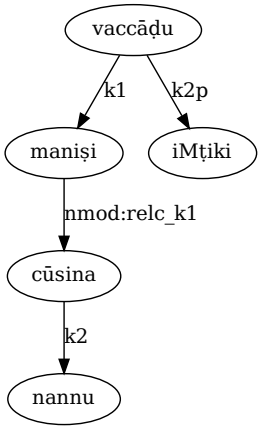

Figure 13: Dependency tree for (16)

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'

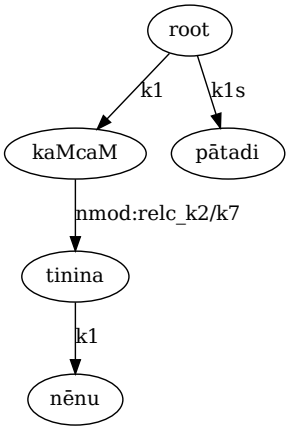

Figure 14: Dependency tree for (17)

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).

Dependency Parsing in a Morphological rich language, Tamil

Vijay Sundar Ram R and Sobha Lalitha Devi
AU-KBC Research Centre
MIT Campus of Anna University
Chennai
sobha@au-kbc.org

Abstract

Dependency parsing is the process of analysing the grammatical structure of a sentence based on the dependencies between the words in a sentence. The annotation of dependency parsing is done using different formalisms at word-level namely Universal Dependencies and chunk-level namely AnnaCorra. Though dependency parsing is deeply dealt in languages such as English, Czech etc the same cannot be adopted for the morphologically rich and agglutinative languages. In this paper, we discuss the development of a dependency parser for Tamil, a South Dravidian language. The different characteristics of the language make this task a challenging task. Tamil, a morphologically rich and agglutinative language, has copula drop, accusative and genitive case drop and pro-drop. Coordinative constructions are introduced by affixation of morpheme 'um'. Embedded clausal structures are common in relative participle and complementizer clauses. In this paper, we have discussed our approach to handle some of these challenges. We have used Malt parser, a supervised learning- approach based implementation. We have obtained an accuracy of 79.27% for Unlabelled Attachment Score, 73.64% for Labelled Attachment Score and 68.82% for Labelled Accuracy.

1 Introduction

Dependency parsing is the process of analysing the grammatical structure of a sentence based on the dependencies between the words in a sentence. It gives the necessary information for various sophisticated NLP tasks such as Information Extraction, Machine Translation, and detailed Sentiment Analysis etc. Extensive research in development of Dependency Treebanks and Dependency parsers are done in languages such as English, Czech, French, German, Arabic, Turkish etc. The annotation of dependency relations is done using different formalisms at word-level namely Universal Dependencies and inter and intra chunk-level namely AnnaCorra. In Indian languages, particularly, in Hindi, Telugu and Bengali, there are a good number of publications on Dependency parser development compared to other Indian languages. Lack of dependency relation annotated corpus in most of the Indian languages is the prime reason. In this paper, we have described dependency relation annotation task and the development of dependency parser for Tamil using a Data-driven approach.

The paper is structured as follows. In section 2, we have discussed the previous attempts in the development of dependency parsers in various Indian languages. A brief introduction on the characteristics of Tamil language is given in section 3. In section 4, we describe the annotation of dependency relations in Tamil sentences. Section 5 has the details of our approach for the development of Tamil dependency parser. This is followed by a section containing the details of the experiment, results and discussion. The paper concludes with the conclusion section.

2 Recent Works

Dependency Treebank work in Indian languages started with development of annotation schema for Indian languages developed by Bharati et. al. (2006). Few of the initial works in Indian

Proceedings of the ICON 2021 Workshop on Parsing and its Applications for Indian Languages (PAIL) , pages 20–26
December 16, 2021. Silchar, India.

languages in Dependency parser are as follows. Bharati et. al. (2008) has proposed a framework for dependency parsing for Indian languages using a grammar-driven methodology. They have presented how the rules made for one language can be effectively transferred to other similar languages. Bharati et. al. (2009) presented a two-stage constraint-based approach to dependency parsing. Here the different grammatical constructions were processed at appropriate stages. This algorithm was tested with Hindi Dependency Treebank data.

ICON Tool Contest on Dependency Parsing during 2009 and 2010 boosted the Dependency parser research in languages namely Hindi, Telugu and Bengali. Different Dependency parsers using Grammar-driven approaches, Malt parser and MST parser and hybrid approaches were developed. The detailed report on the tool contents is available in Husain S. et. al. (2009) and Husain S. et. al. (2010). There are substantial works in these languages using the data released in these tool contests. Few of the notable work in these languages are as follows.

Kesidi R. S. et. al. (2011) has presented a two-stage constraint-based approach to Telugu dependency parser, where they perform a selective identification and resolution of the dependency relation at different stages. The ranking strategy using S-constraints is used to get the best parse. Praveen Gatla (2019) has presented a work on development of Telugu annotated corpus and their experiment in developing Telugu dependency parser using Malt parser and MST parser. The annotated treebank had 2424 sentences. Nallani.S et. al. (2020) has presented a simple and effective dependency parser for Telugu using BERT model built using Telugu Wikipedia corpus. They have attempted to use contextual vector representations instead of hand-crafted features using linguistic information such as part-of-speech and morphology.

Naman Jain (2016) has done a considerable study on Hindi Dependency parser development. He has explained two different ensembling approaches namely re-parsing algorithm and word-by-word voting algorithm in improving the Malt parser. Dhar A. et. al. (2012) came up with a two-stage dependency parser for Bengali, where in the second stage, Bangla specific constraints using Bangla verb frames were used. Morphological features, Part-of-speech, Chunk and Named Entity information were used in both stages.

In Tamil, there are very few published works in Dependency parser. Ramaswamy L. and Zabokrtsky Z. (2011) attempted to build Tamil dependency parser using rule-based technique and also using Malt parser and MST parser. Their annotation schema was partially based on Prague Dependency Treebank (PDT). They annotated a corpus of 3000 words. They observed that the both the rule-based and corpus-based approaches performed poorly in identifying the co-ordinate constructions. Sarveswaran K. and Dias G. (2020) has presented a neural-based dependency parser for Tamil namely TamizhiUDp, developed using UUparser engine developed using Styme S. et. al. (2018) for training with heterogeneous treebanks. As they had 600 Tamil Dependency relation annotated sentences, they experimented multilingual training. They jointly trained Tamil data with Hindi HTTB v2.6 sentences. It has 16647 sentences. They also tried training with other languages such as Telugu, Turkish, but they got better result with Hindi data as the data size was bigger. They got 62.39% as Label Attachment Score (LAS) accuracy. Since Telugu MTG Udv2.6 had only 1328 sentences and without morphological information, it did not suit for combined training in their experiment. In the following section, we give a brief introduction on characteristics of Tamil language.

3 General Characteristics of Tamil Language

Tamil belongs to the South Dravidian family of languages. It is a verb final language and allows scrambling. It has postpositions, the genitive precedes the head noun in the genitive phrase and the complementizer follows the embedded clause. Adjective, participial adjectives and free relatives precede the head noun. It is a nominative-accusative language like the other Dravidian languages. The subject of a Tamil sentence is mostly nominative, although there are constructions with certain verbs that require dative subjects. Tamil has Person, Number and Gender (PNG) agreement.

Tamil is a relatively free word order language, but when it comes to noun phrases and clausal constructions it behaves as a fixed word order language. As in other languages, Tamil also has optional and obligatory parts in the noun phrase.

Head noun is obligatory and all other constituents that precede the head noun are optional. Clausal constructions are introduced by non-finite verbs. Complementizer clause occurs with complementizer markers 'endru' and 'ena'. Subject drop occurs in Tamil. The other characteristics of Tamil are copula drop, accusative drop, and genitive drop. Co-ordinate constructions are also introduced with 'um' suffix. In the next section, we will discuss about the annotation of Dependency relation in Tamil sentences.

4 Tamil Dependency Treebank Annotation

As there is no publicly available Tamil Dependency Treebank, we prepared it inhouse. In the present work of annotating Tamil sentences with dependency relations, we have followed the annotation guidelines given in AnnaCorra (Dipti et. al.. 2012). It is based on modifier-modified relationship. The dependency relations are hierarchically defined as inter-chunk and intra-chunk relations. The grammatical relations that are considered in the guidelines are of two types; (1) Karaka, and (2) Relations other than karakas.

'Karakas' are the roles of various participants in an action. An action in a sentence is denoted through a verb. For a noun to hold a karaka relation with a verb, it is important that they (noun and verb) have a direct relation.

4.1 Relations and Tag labels

The scheme contains about 40 tags which are arrived at considering various types of sentence constructions. These labels represent the following relations (a) karaka and non-karaka dependency relations (b) some underspecified tags of the type vmod, nmod etc and (c) some tags which indicate relations which are not exactly dependency relations but are required to represent the sentence structures. The labels are presented in fig 1.

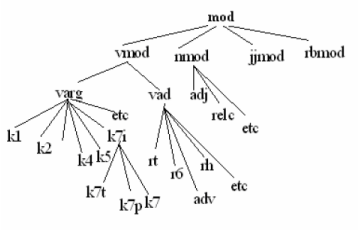

Figure 1: Dependency Relations

S.No	Relation	Meaning
1	k1	Agent / Subject / Doer
2	k2	Theme / Patient / Goal
3	k3	Instrument
4	k4	Recipient / Experiencer
5	k5	Source
6	k7	Spatio-temporal
7	rt	Purpose
8	rh	Cause
9	ras	Associative
10	k*u	Comparative
11	k*s	(Predicative) Noun / Adjective Complements
12	r6	Genitives
13	relc	Modification by Relative Clause
14	rs	Noun Complements (Appositive)
15	adv	Verb modifier
16	adj	Noun modifier

Table 1: Dependency relations and their meaning

To handle the co-ordinate sentence formed by 'um' suffix, and sentences with copula drop, pro-drop, we manually introduce a NULL in that required slot in the sentence and mark the dependency relations. Few of the example sentences are given below.

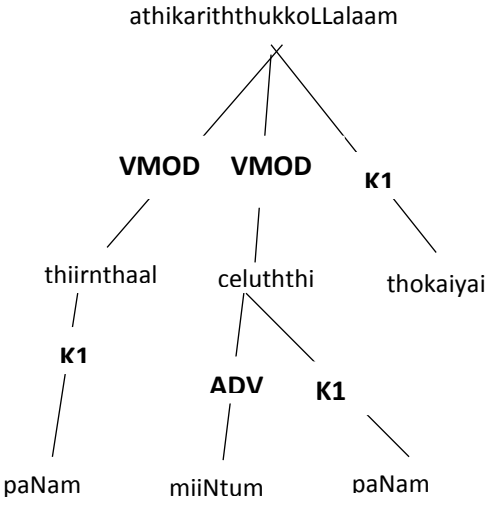

Figure 2: Dependency Tree for example 1

Ex 1:
paNam thiirnthaal miiNtum
Money(N) complete(V+COND) again (Adverb)
paNam celuththi thokaiyai
money(N) pay(V+VBP) amount(N)
athikariththukkoLLalaam .
increase(V+Finite)

(If the money gets emptied, again paying the money, we can increase the amount.)

The sentence in example 1, has a conditional clause 'paNam thiirnthaal' (if money gets empty) and a non-finite clause 'miiNtum paNam celuththi' (by paying money again). These two clausal verbs are attached to the finite verb has vmod relation. And it is shown in fig 2.

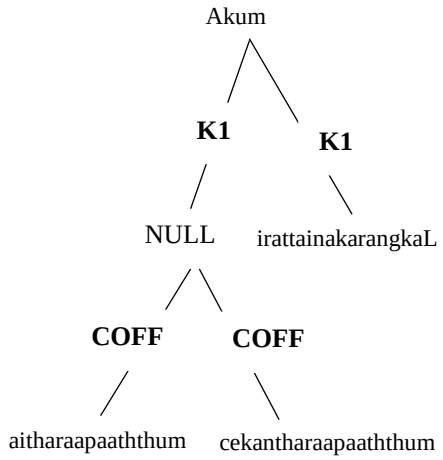

Figure 3: Dependency Tree for example 2

Ex 2:
aitharaapaaththum cekantharaapaaththum
Hyderabad(N+INC) Secunderabad(N+INC)
irattainakarangkaL aakum.
twin-cities be(copula)
(Hyderabad and Secunderabad are twin cities.)

In example 2, the sentence has co-ordination between two nouns aitharaapaaththum (Hyderabad) cekantharaapaaththum (Secunderabad) introduced by 'um' suffix. While annotating this sentence, a NULL is introduced between these two nouns and the nouns are related to the NULL with 'ccof' relation. The dependency tree for this sentence is given in fig 3.

4.2 Corpus Details

We have collected sentences with different sentence structures from various Tamil web Newspapers. We have annotated 2500 sentences, where 70% of the sentences are simple sentences, 20% of the sentences are two clause sentences, 5% of compound sentences and remaining 5% of the sentences had multi clauses (more than two clauses).

5 Our Approach

We have used Malt parser for developing Tamil dependency parser. Malt parser is a language independent system, which can be used to train for any given language. Maltparser is an implementation of inductive dependency parsing. The syntactic analysis of a sentence amounts to the derivation of dependency structure. It is possible to improve the performance of the system by optimizing the parameter of the transition system and optimizing the features used for the classifier system. Maltparser implementation has nine deterministic parsing algorithms, viz, Nivre arc-eager, Nivre arc-standard, Convington non-projective, Convington projective, Stack projective, Stack swap-eager, Stack swap-lazy, planner and 2-planner. It supports two machine learning packages, LIBSVM and LIBLINEAR. In the present work, we have used arc-eager transition and morphologically rich features. Nivre. J (2009) has presented the importance of morphological information in developing Dependency parser for morphologically rich language namely Telugu. We have used LIBSVM classifier with the following features POS, chunk information along with the root word, case suffix, other suffixes and TAM (tense-aspect-model) and PNG (Person, Number and Gender) information from verbs. This information is obtained from the morphological analysis of the words.

Dependency relation annotated sentences were enriched with morphological information, Part-of-Speech, Chunk and chunk-head information using a robust Morphological analyser build using a paradigm approach, CRFs based part-of-speech tagger and chunker and rule-based head chunk identification module. The head chunk information is required for marking the inter-chunk relation.

As Tamil sentences have copula drop and co-ordinate sentences with 'um' suffix, we need to re-structure these sentences by adding NULL in the required positions.

5.1 Copula Drop Handler

We check for the finite verb in the sentence, if it does not exist, NULL is introduced as the last word of the sentence before the punctuation marker. Consider the following example sentence.
Ex 3:
naan doctor.
I am a doctor .

After addition of NULL, the sentence is as follows,

"naan doctor NULL ."

5.2 Co-ordinate Construction Handler

'um' suffix is also used as an emphatic marker. Using linguistic rules, we disambiguate the sentences with emphatic 'um' and co-ordinate suffix 'um'. Co-ordinate suffix 'um' suffix can occur in the series of nouns and verbs. The algorithm to handle this is explained below.

Step1: If sentence has multiple words with 'um' suffixes, then step 2

Step2: If sentence has a series of nouns or verbs with 'um' suffix
then
introduce NULL between the last pair of noun/verb with 'um' suffix.

Consider the sentence in example 4.

Ex 4:
aitharaapaaththum cekantharaapaaththum
Hyderabad(N)+INC Secendrabad(N)+INC
irattainakarangkaL aakum.
twin-cities be(copula)
(Hyderabad and Seceundrabad are twin cities.)
The sentence in example 4, has two nouns with 'um' suffix.

After Correction:
"aitharaapaaththum NULL cekantharaapaaththum irattainakarangkaL aakum. "

Dependency tree for the sentence in example 4 is shown in figure 3.

6 Experiment and Results

The dependency relation annotated sentences were randomly divided into 80% and 20% for training and testing purpose. Both the training data and the testing data are processed with morphological analyser, POS tagger, Chunker and chunk head identification module. Further sentences with Copula drop and sentences with 'um' co-ordinate suffix are corrected by introducing a NULL. This processed data is presented to Malt parser in CoNLL column format for training and testing.

We have evaluated the performance of the model with standard measures namely Label Attachment Score (LAS), Unlabeled Attachment Score (UAS), Labeled Accuracy (LA) metrics. The training data had 2000 sentences and the testing data had 500 sentences. We performed a detailed analysis on the contribution of each of the features in improving the accuracy. The performance scores obtained when introducing different features are given table 2.

S.No	System	Features	UAS	LAS	LA
1	Baseline	Word, root word	61.45	57.29	54.84
2	Baseline +POS	Word, Root word, POS	66.14	61.72	56.34
3	Baseline +POS +Case	Word, root word, POS, Case marker	75.46	71.84	66.23
4	Baseline +POS +Case +Other suffix	Word, root word, POS, Case marker, PNG information, other suffix	79.27	73.64	68.82

Table 2: Performance Scores of different systems

The performance scores in table 2, show the contribution of different features in improving the accuracy of the parsing. Tamil being a morphologically rich language with highly productive suffixation, the information from the processing modules contribute a lot of information. Case markers affixed to the nouns help in determining the semantic role of that noun phrase in that sentence. Both finite and non-finite verbs have clear suffix markers affixed to the verbs. These suffixes are vital features for the classifier. This is evident from the accuracy obtained for 3rd and 4th system, where, case markers and other suffixes are included as features. Though 'vmod' relations are identified properly, the system has poorly identified the Karakas relations in the sub-ordinate clause. Dropping of genitive case and accusative case

markers are common characteristics in Tamil. This affects the proper identification of K2 and r6 relations. It also introduces wrong K1. The sentences with embedded clause were not correctly handled. This requires re-ordering of sentence into linear form to correctly identify the relations.

The Overall performance measures obtained are presented in the following table 3.

UAS	LAS	LA
79.27%	73.64%	68.82%

Table 3: Overall Performance Measures

The performance measures for the frequently occurring tags are given in table 4.

Dep-Rel	Recall	Precision	F-measure
k1	81.64	78.23	78.84
k1s	62.49	64.45	63.45
k2	76.62	72.41	74.46
k2p	74.98	73.12	74.04
k2s	23.92	31.48	27.18
k4	71.53	64.54	67.86
k4a	19.34	56.62	28.83
k7	43.69	49.27	46.32
k7p	63.37	62.74	63.05
k7t	67.84	65.23	66.51
nmod	26.54	68.57	38.27
vmod	82.74	75.61	79.02
adv	77.83	74.81	76.29
Ccof	77.34	73.38	75.31
r6	74.85	67.45	70.96
rh	77.63	71.54	74.46

Table 4: Performance Measures of Major Tags

On analysing the output, we had the following observations.

k2 karaka relation is marked as k1 is a common error. It has happened in the sentences where the accusative case marker is dropped in the object noun phrase. There are instances where k1 is marked as k2. r6, which marks the possessive relation between the nouns were poorly identified. As genitive case drop is common in Tamil sentences, the relation between the nouns were not captured. This led to wrong tagging of dependency relations.

The parser has failed to handle embedded clause sentences, where karaka relation of the clausal verb and the main verb were wrongly attached. In these sentences, the 'vmod' relation between the clausal verb and the main verb were marked correctly.

K4a- karaka relation which refers to the experiencer relation, were wrongly tagged as k4 or k1. And these relations were less frequently occurred in the corpus. The parser has not handled the multiclause sentences correctly. Here the 'vmod' relations were correctly tagged but the karaka relations with clausal verb and main verb were not correctly tagged.

Thus, to improve the parser efficiency, we need to increase the training data. Further the rules should be included to handle genitive case drop, accusative case drop and subject drop. We need to identify the sentence structures using the clausal verb and handle them separately, it will reduce the errors due to wrongly attached NPs with other verbs. We should also use the post-processing rules to improve the parser efficiency.

Conclusion

It is also advised to supplement non-English characters and terms with appropriate transliterations and/or translations since not all readers understand all such characters and terms. We presented a work on developing Dependency parser for Tamil, which is a less attempted task. Tamil, a south Dravidian language, is a morphologically rich and highly agglutinative language. We used Malt parser to train the language model with morphologically rich features. We enrich the sentences with morphological information, PoS tags, Chunks and chunk head information using shallow processing tools. The copula-drop and sentences with co-ordinate 'um' suffixes are rewritten by introducing NULL in the required position in the sentences, before processing it with a parser. Accusative case drop, genitive case drop and subject drop are common in Tamil sentences and these affect the efficiency of the parser. Embedded clause occurs in relative participle clause and complementizer clause sentences and these sentence constructions are poorly handled by the parser. We plan to build a confusion matrix for deeper understanding of the errors.

To improve the efficiency of the parser we plan to train the model with more annotated data and to

add linguistic rules to handle accusative case drop, genitive case-drop and subject drop. Further we plan to improve the algorithm to selectively handle the different sentence structures. Increasing the annotated data substantially, we also plan to train a neural-based dependency parser, where we can use resources such as BERT which provides a contextual vector representation, to build a robust parser.

References

Bharati, A., Sharma, D. M., Bai, L, Sangal, R.: AnnCorra: Annotating Corpora Guidelines for POS and Chunk Annotation for Indian Languages. LTRC-TR31. (2006)

Bharati, A., Husain, S, Sharma, D.M., Sangal, R.: A Two-Stage Constraint Based Dependency Parser for Free Word Order Languages. In: the COLIPS IALP. (2008)

Bharati, A., Husain, S., Vijay, M., Deepak, K., Misra, D., Sangal, R.: Constraint Based Hybrid Approach to Parsing Indian Languages. In: The 23rd Pacific Asia Conference on Language, Information and Computation. Hong Kong (2009)

Dhar. A., Chatterji. S., Sarkar. S., Basu. A. 2012, In: Proceedings of the 10th Workshop on Asian Language Resources, COLING 2012, Mumbai, December 2012, pages 55–64

Husain, S., 2009. Dependency Parsers for Indian Languages. In: Proceedings of the ICON09 NLP Tools Contest: Indian Language Dependency Parsing. India.

Husain, S., Mannem, P., Ambati, B., Gadde, P.2010. The ICON-2010 tools contest on Indian language dependency parsing. In: ICON-2010 tools contest on Indian language dependency parsing. Kharagpur, India

Jain, N., 2016. Advancements to Hindi Dependency Parsing: Semantic Information, Ensembling and PAC (Doctoral dissertation, PhD thesis, International Institute of Information Technology Hyderabad).

Kesidi S.R., Kosaraju. P., Vijay. M. and Husain. S. 2011. A Constraint Based Hybrid Dependency Parser for Telugu. In Proceedings of the 12th CICLing, Tokyo, Japan.

Amba kulkarni, 2021, Sanskrit Parsing Following Indian Theories of Verbal Cognition, ACM Transactions on Asian and Low-Resource Language Information Processing, Vol 20, Number 2, pp 1-38,

Kulkarni A 2019, Sanskrit Parsing based on the theories of Shabdabodha, D. K. PrintWorld and Indian Institute of Advanced Study, August 2019

Nallani, S., M. Shrivastava & D. Sharma. 2020.A Fully Expanded Dependency Treebank for Telugu. Proceedings of the WILDRE5– 5th Workshop on Indian Language Data: Resources and Evaluation. Marseille: Language Resources

Nivre, J., Hall, J., and Nilsson, J. 2006. Maltparser: A data-driven parser-generator for dependency parsing. In Proceedings of LREC, volume 6, pages 2216–2219.

Nivre, J., 2009. Parsing Indian Languages with MaltParser. In: Proceedings of the ICON09 NLP Tools Contest: Indian Language Dependency Parsing.

Praveen Gatla 2019. Dependency Parsing for Telugu Using Data-driven Parsers. Language in India. Vol. 19:1

Ramaswamy L. and Zabokrtsky Z. 2012. Prague dependency style treebank for Tamil. In Proceedings of Eighth International Conference on Language Resources and Evaluation (LREC 2012), pages 1888–1894, Istanbul, Turkey

Sarveswaran K. and Dias. G. 2020. ThamizhiUDp: A Dependency Parser for Tamil. CoRR, abs/2012.13436. https://arxiv.org/abs/2012.13436}

Neural-based Tamil Grammar Error Detection

Murugesapillai Dineskumar
University of Moratuwa, Sri Lanka
170141X@uom.lk

Ravinthirarasa Anankan
University of Moratuwa, Sri Lanka
170032N@uom.lk

Kenatharaiyer Sarveswaran
University of Moratuwa, Sri Lanka
sarves@cse.mrt.ac.lk

Gihan Dias
University of Moratuwa, Sri Lanka
gihan@uom.lk

Abstract

This paper describes an ongoing development of a grammar error detector for the Tamil language using the state-of-the-art deep neural-based approach. This proposed checker captures a vital grammar error called subject-predicate agreement errors. In this case, we specifically target the agreement error between nominal subjects and verbal predicates. We also created the first-ever grammar error annotated corpus for Tamil. In addition, we experimented with different multi-lingual pre-trained language models to capture syntactic information and found that IndicBERT gives better performance for our tasks. We implemented this grammar checker as a multi-class classification on top of the IndicBERT pre-trained model, which we fine-tuned using our grammar-error annotated data. This baseline model gives an F1 Score of 84.0. We are now in the process of improving this proposed system with the use of a dependency parser.

1 Introduction

Grammar error detection is the task of identifying grammatical errors in the text. This feature is available as a part of stand-alone applications, such as Microsoft Word, Libre Office, and online applications, such as Grammarly and Google Docs. However, none of these applications supports the grammar error detection of Tamil and most other Indian languages.

In recent times, neural-based approaches are also being employed for grammar error detection tasks. However, unlike other well-resourced languages such as English and German, applying neural-based approaches to Tamil is difficult due to the lack of quality annotated data.

This paper outlines how we implemented an application to detect grammar errors related to the subject-predicate agreement in Tamil. We have created a grammar error annotated corpus to train the application. We have employed a neural-based approach and a transfer learning technique to implement the proposed application.

2 Motivation

Tamil is a morphosyntactically rich and free-order language. It is spoken by more than 78 million people around the world,[1] and is the official language of Sri Lanka, Singapore, and Tamil Nadu, India. Tamil is a diglossic language with a spoken and written form. The spoken form varies from region to region; however, the written form is almost the same among regions. Tamil documents are being prepared electronically nowadays, including official documents. However, most of the time, these documents are typed in by people who are not well versed with Tamil grammar.

On the other hand, official government documents are not supposed to have any grammar mistakes. It is even more critical in a multi-lingual country such as Sri Lanka, where sometimes even a non-Tamil person may type in official letters. Therefore, all documents have to be checked for all types of errors and corrected. Further, nowadays, many efforts are being made to develop machine translation systems. We need to ensure that translations are grammatically correct before using them to train systems.

3 Literature review

Several studies have been carried out to develop spell checkers for Tamil, including (Sakuntharaj and Mahesan, 2016, 2018, 2019; Segar and Sarveswaran, 2015; Uthayamoorthy et al., 2019; Rajendran, 2012). However, no grammar error checkers are found online or integrated into other

[1] http://www.languagesgulper.com/eng/Tamil.html

Proceedings of the ICON 2021 Workshop on Parsing and its Applications for Indian Languages (PAIL) , pages 27–32
December 16, 2021. Silchar, India.

applications. On the other hand, grammar error checkers for well-resourced languages are readily available online as cloud-based tools such as Google Docs, Grammarly, and stand-alone office suites.

There are 28 different types of errors that have been reported in literature (Ng et al., 2014). In addition to the listed errors, Tamil also has a special type of error called *Sandhi* error. Although *Sandhi* is considered as the result of a phonological operation between two words or two morphs, *Sandhi* also shows syntactic clues as discussed shown by Sarveswaran and Butt (2019). In this respect, *Vaani*[2], which is developed using a rule-based approach, can be considered as a partial grammar checker as it handles *Sandhi* errors.

Nowadays, several multi-lingual pre-trained models(Conneau and Lample, 2019; Conneau et al., 2019; Xue et al., 2021; Kakwani et al., 2020; Devlin et al., 2019) are available online. These models are trained with millions of sentences and tokens. The pre-trained models capture various linguistic information, including morphological, syntactical and semantic information of sentences. However, these details are not in a specific format; therefore, not very easy to retrieve. XLM-R (Conneau et al., 2019), IndicBERT (Kakwani et al., 2020), and BERT (Devlin et al., 2019) are also trained with Tamil data. Therefore, these models also capture linguistic features of Tamil.

We require a large set of annotated corpus to train a machine learner to carry out the task of our interests. However, the Tamil language does not have an error annotated corpus. This kind of error annotated corpora can be created not only by hand but also with the assistance of tools like Part of Speech taggers, morphological analysers, and syntactic parsers.

4 The proposed grammar error detector

This section outlines the process that has been followed to develop the proposed grammar error detector using a neural-based approach and transfer-learning technique.

4.1 Scope

We handle only the modern written Tamil text. Because Tamil is a diglossia language that evolved over several millennia, even the spoken forms vary significantly among different regions. Therefore, it is complicated to draw grammar rules for them. Further, over time Tamil also underwent several grammatical changes. Therefore, we decided to focus only on the modern text that was written after 2000. We collected text from this period for training, evaluation, and testing of the proposed grammar checker.

Further, instead of considering all the grammar errors, we handle only the type of error called subject-predicate agreement. In Tamil, the subject-predicate agreement is an important condition that needs to be met for any sentence to be grammatical. Tamil can have nominated subjects and non-nominative subjects. However, in our case, we focus only on nominative subjects as there is no agreement between non-nominative subjects and the verbal predicates. Similarly, we do not handle a nominal predicate as there are no agreements between a subject and the nominal predicates. Therefore, our focus is only on the nominative subject-verbal predicate agreement where both of them need to agree on gender, number and person. Even if one of these does not match, it is considered a grammar error. Although this agreement needs to be held on rationality, we do not handle it separately as rationality errors can be tracked using person, number, and gender errors.

4.2 Data

Except for a spelling error annotated word list,[3] which is tiny in size, there was no other error annotated list found online. Therefore, we created a grammar annotated dataset that marks subject-predicate agreement errors, specifically person, number, and gender errors. Table 1 shows details of our corpus. The dataset has 5546 sentences taken from news sources. We decided to use this to develop a baseline system and then get the baseline system to generate more error annotated datasets incrementally.

The task of grammatical error detection is formalized as such, given Tamil sentence **X** as input, the error detector outputs its prediction **Y** where,

$$Y = \begin{cases} 0, & \text{if X is correct.} \\ 1, & \text{if X has gender error.} \\ 2, & \text{if X has person error.} \\ 3 & \text{if X has number error.} \end{cases}$$

The dataset we collected has been divided into training, validation, and testing sets, containing 4645, 460, and 481 sentences. It is non-

[2] http://vaani.neechalkaran.com/

[3] https://www.kaggle.com/neechalkaran/error-annotated-tamil-corpus

Table 1: Size of each class in the dataset

Class	Number of sentences			Total
	Train	Validation	Test	
grammatical	2455	120	121	2696
person	913	120	120	1153
number	772	120	120	1012
gender	505	100	120	725
Total	**4645**	**460**	**481**	**5546**

Table 2: Example data entries from our error annotated corpus

Erroneous sentence	Errorless sentence	Error type
கவிதா வந்தான் . kavitā vantān Kavitha.NOM-**3SgF** come.**3SgM** .	கவிதா வந்தாள் . kavitā vantāḷ Kavitha.NOM-**3SgF** come.**3SgF**.	Gender
நான் நாளை வந்தாள் . nān nāḷai vantāḷ I.NOM-**1Sg** come.**3Sg** .	நான் நாளை வருவேன் . nān nāḷai varuvēn I.NOM-**1Sg** come.**1SgF**.	Person

overlapping and balanced in terms of the type of errors. Table 2 shows two example entries of error annotated corpus, a number error and a gender error.

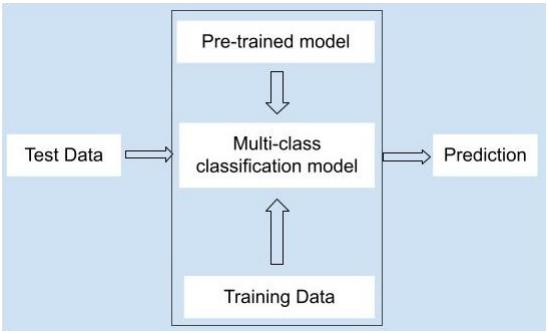

Figure 1: Overview of methodology

5 Approach

As illustrated in Figure 1, we used a supervised approach to develop the proposed grammar error detector. However, instead of training a model from scratch, which requires a significant amount of data and processing power, we used a pre-trained language model to capture the morphosyntax and then modelled the grammar error detection as a multi-class classification problem on top of it. In order to do that, we have created a grammar error annotated corpus to fine-tune the pre-trained model and implement our classification model. We used our training set and validation set for this purpose, and then we evaluated the system using the test set.

5.1 Identifying a pre-trained model

As a first step, we have identified a pre-trained model which works better for our problem. We experimented with XLM-R (Conneau et al., 2019), IndicBERT (Kakwani et al., 2020), and mBERT (Devlin et al., 2019). Table 3 shows the comparison of these models in respective to their token size, parameters, and test results as reported by Kakwani et al.,(2020). We made use of a framework called Simple Transformers[4] to carry out our experiments.

The Simple Transformer framework provides supports for various pre-trained models and tasks such as text classification, token classification, question answering, and language modelling. We can easily set up a classification layer on top of the pre-trained model using this framework. Further, this framework also supports changing various parameters, including learning rate, batch size, and epochs.

We fine-tuned the given three models using our error annotated corpus and by varying different parameters as shown in Table 5. Finally, we also evaluated the model using the test set.

Table 4 shows the results we obtained for all three models, and from which it is clear that the IndicBERT pre-trained model outperforms other models with the F1 score of 73.4%. Therefore, we

[4] https://simpletransformers.ai/

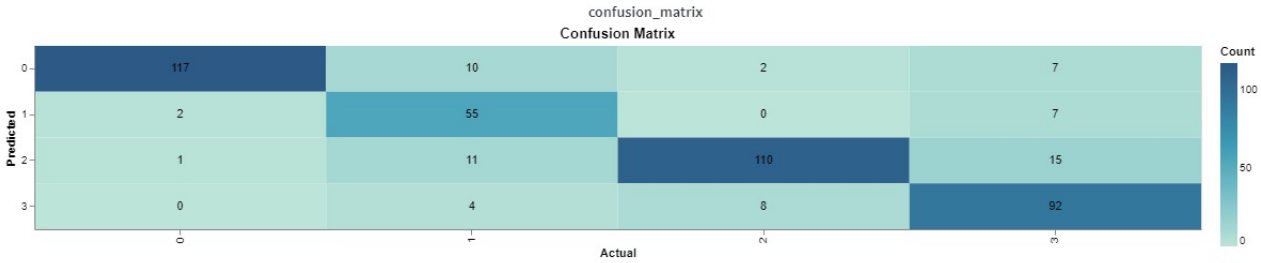

Figure 2: confusion matrix

Table 3: Pre-trained models, token size, number of parameters, and the test results for different tasks in indicGLUE - Source: (Kakwani et al., 2020)

Language model	token size	parameters	test accuracy
XLM-R-base	595 Million	125M	61.09
IndicBERT	**549 Million**	**12M**	**66.66**
bert-base-multilingual-cased		110M	64.62

Table 4: F1 score of different pre-trained models

Pretrained model	MCC	F1 Score
XLM-R-base	0.58048	0.73052
IndicBERT	**0.59426**	**0.73684**
bert-base-multiling-cased	0.58933	0.73474

Table 5: Different hyperparameter used for evaluation

Hyper parameter	values
Learning Rate	1E-5,2E-5,3E-5,4E-5,5E-5
Batch Size	16, 32
Epochs	2, 3, 4

decided to use this model to improve grammar error detection for future experiments.

5.2 Evaluation

We used two standard metrics, namely MCC (Matthew Correlation Coefficient)(Matthews, 1975) and F1 score, to evaluate the model that we trained. Table 4 shows the initial performance of different fine-tuned classification models for the test set. It is evident from the results that the IndicBERT outperforms other pre-trained models. Moreover, since hyper-parameters also affect the results, we experiment with different hyper-parameter combinations to fine-tune the classification model. Table 5 shows fine-tuned values for the set of hyper-parameters. We change the hyper-parameters to get a better F1 score. Initially, the F1 score was 73%. Also, we found significant confusion among number errors and gender errors. The dataset has number error, gender error, person error and error-less sentences. We also found that some sentences have two kinds of errors when we look deeper. Therefore, we defined error precedence to the prioritise error labels as number > gender > person. For instance, Table 6 shows how number error is prioritised over the gender error in the dataset. After this precedence setting, the grammar error detection showed the F1 score of 84%. Figure 2 shows the current confusion matrix among different type of errors. Eventually, we obtained the best results for the combination of learning rate = 3E-05, batch size = 16, and epochs = 4 along with IndicBERT. Equations 1, 2, and 3 show that how we calculated the F1 score from True Positive (TP), False Positive (FP), and False Negative (FN) values.

$$Precision = \frac{\text{TP}}{\text{TP} + \text{FP}} = \frac{374}{374 + (19{+}9{+}27{+}12)} = 0.84 \quad (1)$$

$$Recall = \frac{\text{TP}}{\text{TP} + \text{FN}} = \frac{374}{374 + (3{+}25{+}10{+}29)} = 0.84 \quad (2)$$

$$F_1 = \frac{\text{Precision} \times \text{Recall}}{\text{Precison} + \text{Recall}} = \frac{2 \times 0.84 \times 0.84}{0.84 + 0.84} = 0.84 \quad (3)$$

Table 6: Precedence of errors

Erroneous sentence	number error	gender error	error type
கவிதா வந்தான் . kavitā vantān Kavitha.NOM-**3SgF** come.**3SgM** .	false	true	gender error
தமிழ் மொழி பழமையானவை Tamil moli palamaiyānavai Tamil language.NOM-**3Sg** old.**3Pl** .	true	true	number error

6 Conclusion

We have implemented a baseline application for Tamil grammatical error detection using the state-of-the-art approach. The application outlined here detects grammatical errors related to the person-number-gender agreement between the nominative subject and the verbal predicate in a sentence. We used a multi-lingual pre-trained model to capture the Tamil structures and then fine-tuned it using the grammar error annotated data we created. We found that the IndicBERT model gives better accuracy than other pre-trained models. Our baseline model shows an F1 Score of 84.0% for unseen a test set.

As the next step, we are planning to use *Thamizhi*Morph (Sarveswaran et al., 2021) — A Morphological analyser to create more annotated data to train the grammar checker. The current model relies on the pre-trained model to capture the syntactic information such as subject and predicate. However, this can be obtained using a syntactic parser, and the syntactically parsed data may increase the score. Therefore, as the next step, we will also experiment with a Tamil dependency parser called *Thamizhi*UDp (Sarveswaran and Dias, 2020) to incorporate syntactic information such as subject and predicate information into our datasets to see whether the proposed system can be improved further.

References

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Unsupervised cross-lingual representation learning at scale. *arXiv preprint arXiv:1911.02116*.

Alexis Conneau and Guillaume Lample. 2019. Cross-lingual language model pretraining. *Advances in Neural Information Processing Systems*, 32:7059–7069.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Divyanshu Kakwani, Anoop Kunchukuttan, Satish Golla, NC Gokul, Avik Bhattacharyya, Mitesh M Khapra, and Pratyush Kumar. 2020. inlpsuite: Monolingual corpora, evaluation benchmarks and pre-trained multilingual language models for indian languages. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 4948–4961.

Brian W Matthews. 1975. Comparison of the predicted and observed secondary structure of t4 phage lysozyme. *Biochimica et Biophysica Acta (BBA)-Protein Structure*, 405(2):442–451.

Hwee Tou Ng, Siew Mei Wu, Ted Briscoe, Christian Hadiwinoto, Raymond Hendy Susanto, and Christopher Bryant. 2014. The conll-2014 shared task on grammatical error correction. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*, pages 1–14.

S Rajendran. 2012. Preliminaries to the preparation of a spell and grammar checker for Tamil. *Language in India*, 2.

Ratnasingam Sakuntharaj and Sinnathamby Mahesan. 2016. A novel hybrid approach to detect and correct spelling in Tamil text. In *2016 IEEE International Conference on Information and Automation for Sustainability (ICIAfS)*, pages 1–6. IEEE.

Ratnasingam Sakuntharaj and Sinnathamby Mahesan. 2018. Detecting and correcting real-word errors in Tamil sentences. *Ruhuna Journal of Science*, 9(2).

Ratnasingam Sakuntharaj and Sinnathamby Mahesan. 2019. Detecting and Correcting Grammatical Mistakes due to Subject-Verb Inconformity and Conflicts in Tense Aspects in Tamil Sentences. In *6th Ruhuna International Science and Technology Conference (RISTCON)*.

Kengatharaiyer Sarveswaran and Miriam Butt. 2019. Computational challenges with Tamil complex predicates. In *Proceedings of the LFG19 conference, Australian National University. CSLI, Stanford*, pages 272–292.

Kengatharaiyer Sarveswaran and Gihan Dias. 2020. ThamizhiUDp: A dependency parser for Tamil. In *Proceedings of the 17th International Conference on Natural Language Processing (ICON)*, pages 200–207, Indian Institute of Technology Patna, Patna, India. NLP Association of India (NLPAI).

Kengatharaiyer Sarveswaran, Gihan Dias, and Miriam Butt. 2021. ThamizhiMorph: A morphological parser for the Tamil language. *Machine Translation*, 35(1):37–70.

J Segar and K Sarveswaran. 2015. Contextual spell checking for Tamil language. In *14th Tamil Internet conference*, pages 1–5.

Keerthana Uthayamoorthy, Kirshika Kanthasamy, Thavarasa Senthaalan, Kengatharaiyer Sarveswaran, and Gihan Dias. 2019. DDSpell-A Data Driven Spell Checker and Suggestion Generator for the Tamil Language. In *2019 19th international conference on advances in ICT for emerging regions (ICTer)*, volume 250, pages 1–6. IEEE.

Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. 2021. mT5: A Massively Multilingual Pre-trained Text-to-Text Transformer. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 483–498. Association for Computational Linguistics.

Author Index