International Transactions on Electrical Engineering and Computer Science
Kumari et.al., Vol: 2, No: 1, pp: 37-43, March 2023

Telugu Dependency Treebank

B. V. Seshu Kumari*[,1], M. Susmitha[1], S. Sudeshna[2], P. Bala Kesava Reddy[1]

Abstract: We discuss Telugu Language and Treebanks briefly in this work. Initially, we'll go over the Telugu language briefly. The paninian grammatical model utilized for Telugu dependency representation is then described. Following that, we explain Telugu treebanks and the various formats used to express these treebanks. We also discuss the Telugu language and its representation in the Telugu Dependency Treebank, and we give information on the Telugu language and the Telugu Dependency Treebank. Natural languages are often morphologically rich, and they create sentences in a variety of ways. Researchers have been investigating approaches to annotate text with linguistic knowledge since the advent of machine translation in the 1960s. Previous studies on Indian languages were done at the chunk level. The present shallow parser morphologically parses the input text to the chunk label. Researchers are considering working at the phrase level in the future. They broke the phrases down into smaller parts. The relationship between chunk heads is essential to proceed to sentence-level parsing. This results in reliance parsing.

Keywords: Natural language processing, Telugu language, Dependency parsing, Telugu tree bank

1. Introduction

Natural language processing (NLP) plays vital role in a wide range of applications like machine translation, conversation systems, text creation, word meaning disambiguation, and so on.

Article History
Received: 02-01-2023; Revised: 28-02-2023;
Accepted: 15-03-2023

*Corresponding author: Department of Information Technology, VNR Vignana Jyothi Institute of Engineering and Technology, Hyderabad-500090, India.

E-Mail: seshukumari_bv@vnrvjiet.in

Ph: +91-9949084722

[1]Department of Information Technology, VNR Vignana Jyothi Institute of Engineering and Technology, Hyderabad-500090, India.

E-Mail: susmitha_m@vnrvjiet.in,
balakesavareddy_p@vnrvjiet.in

[2] Department of Computer Science and Engineering, VNR Vignana Jyothi Institute of Engineering and Technology, Hyderabad-500090, India.

E-Mail: sudeshna_s@vnrvjiet.in

Parsing is an important concept, but parsing morphologically ironic, free-word order languages has remained a difficult issue. Dependence construction has been useful for unrestricted word direction languages. Data-driven, Grammar-driven and hybrid dependency phrases are all possible. Grammar-driven parsers are quite difficult to build and should only be attempted by a professional with an extensive understanding of the language. Telugu, an Indian language, is a flexible word order language with a complex morphology [1-5].

1.1 Telugu language

Telugu is a highly diverse language and is one of the top official languages in India. It is also the 13th most spoken language in the world, with an estimated 74 million speakers. Unlike many languages, Telugu does not have a fixed word order, which means that speakers can arrange their words in any order they like. Additionally, Telugu is an agglutinative language, meaning that it uses suffixes to indicate morphological information rather than distinct words [6]. Telugu verbs use suffixes to indicate gender, tense, aspect, and modality. For example, the verb 'tinnadu' (ate) uses the suffix 'du' to indicate masculinity, while the verb 'tinnadhi' is used when the subject is feminine.

1	raamudu	oka	pamdu		tinnadu	
	Ram	one	fruit		eat+past+mas	
			'Ram ate a fruit'			
2	sitha	oka	pamdu		tinnadhi	
	Sita	one	fruit		eat+past+mas	
			'Sita ate a fruit'			
3	raamudu	oka	pamdu		tintunnaadu	
	Ram	one	fruit		eat+cont+mas	
			'Ram is eating a fruit'			
4	raamudu	oka	pamdu	sithaki	patashalalo	ichadu
	Ram	one	fruit	Sita-to	School-in	give+past+mas
			'Ram gave a fruit to Sita in the school'			
5	oka	pamdu	raamudu		tinnadu	
	one	fruit	Ram		eat+past+mas	
			'Ram ate a fruit'			

Fig. 1: Example sentences in Telugu

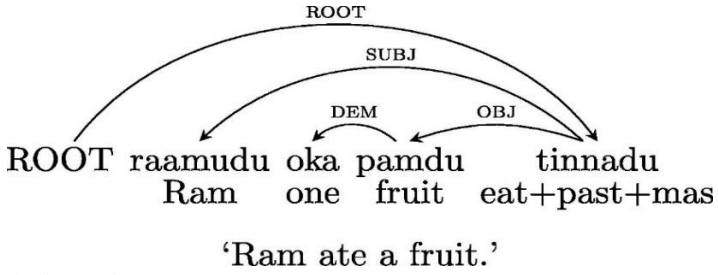

Fig. 2: Dependency Tree example for Telugu sentence

Similarly, the verb suffix changes when the tense shifts, as seen in the verb 'tunnaadu,' which indicates present continuous tense. Noun suffixes in Telugu indicate case or preposition, as seen in the phrases 'sithaki' (to Sita) and 'patashalalo' (in school) in the sentence 'Ram presented a fruit to Sita at the school.' The suffix 'ki' indicates the dative case, while 'lo' indicates the preposition 'in'. Finally, Telugu is highly flexible in terms of word order, as demonstrated in sentence 5. This allows speakers to place words in any order they like, adding to the rich linguistic diversity of the language [7-10].

Despite the usual word, the order is Subject-Object-Verb alternative usage is also feasible in Telugu laguage, as in sentence 5, with Object-Subject-Verb order.

1.2 Telugu dependency treebank

Our work involves the use of the Telugu dependency treebank, which was provided as part of the ICON 2010 Tools challenge. The treebank contains annotations such as POS tagging, chunking, and various annotation rules. It also includes morphological data such as gender, number, suffix, root, coarse pos-tag, person, case marking, and TAM (Aspect, Tense, and Modality marker). Additionally, the treebank provides information on POS, chunk, and dependency. The dependency annotation technique used in the treebank is based on Paninian grammar, which is well-suited to Indian languages. According to the Paninian approach, a sentence is considered a collection of modifier-modified relations [11-13]. The main verb in a sentence is typically the principal modification, and the parts that modify the verb are involved in the activity specified by the verb. Karaka refers to the participant relationships with the verb, which encompass the 'local' semantics of the verb in a phrase, as well as surface-level morpho-syntactic information. Fig. 2 shows the dependency tree

pictorial representation for an example Telugu sentence [14]. The details of the Telugu dependency treebank's data sets are shown in Table 1. This table provides statistics on word count, sentence count, and average sentence length.

2. Representation

Shakti Standard Format (SSF) is used for the actual annotation of Telugu Dependency tree bank. Using data-driven parsers such as MaltParser and MSTParser, the annotated data is additionally translated to CoNLL format for ease of the experiments [16-17].

(1) raamudu oka pamdu tinnadu
'Ram' 'one' 'fruit' eat+past+mas
'Ram ate a fruit

Table. 1: Statistics of Telugu tree bank

Type	Word Count	Sent Count	Avg. sent length
Test	836	150	5.57
Devel	839	150	5.59
Train	7602	1,400	5.43

2.1 SSF Format

There are four columns in the SSF format. Token id, token/chunk boundaries, POS/Chunk tags, and feature structure are displayed in the four columns, in that order. A detailed description of SSF can be found in [19-20]. Consider an example Telugu sentence shown below. SSF representation with POS, chunk, and dependency annotation for the above sentence is as follows.

```
1     ((    NP    <fs drel = 'SUBJ:VGF' name='NP'>
1.1   raamudu NNP
      ))
2     ((    NP    <fs drel='OBJ:VGF' name='NP2'>
2.1   oka   DEM
2.2   pamdu NN
      ))
3     ((    VGF   <fs name='VGF'>
3.1   tinnaduVM
      ))
```

The SSF above depicts the relationships between chunk heads. The dependencies are listed in the SSF's fourth field. A unique id is assigned to the head via an attribute value pair. This can be seen in node 3 above, where 'name' is given the id 'VGF'. The dependents are then linked to the head through the 'drel' attribute. "dependency relation: head id" is the value for drels. SSF representation gives information at the chunk level, allowing us to see only inter-chunk dependency relations. Dependencies between chunks are not supported. An automatic tool is used to extract intra-chunk dependencies. The annotation of sentence-level information can be seen in addition to the chunk-level annotation, as demonstrated by the expanded tree of the preceding sentence. It is important to note that information regarding chunk boundaries and chunk heads is retained in the expanded trees through the use of 'chunk Id' and 'chunk type' features.

1. raamudu NNP <fs drel='SUBJ:tinnadu' name='raamudu' chunkId='NP' chunkType='head:NP'>
2. oka DEM<fs drel='DEM:pamdu' name='oka' chunkId='NP2' chunkType='child:NP2'>
3. pamdu NN <fs drel='OBJ:tinnadu' name='pamdu' chunkId='NP2' chunkType='head:NP2'>
4. tinnadu VM <fs name='tinnadu' chunkId='VGF' chunkType='head:VGF'>

The SSF above depicts the relationships between chunk heads. The dependencies are listed in the SSF's fourth field. A unique id is assigned to the head via an attribute value pair. This can be seen in node 3 above, where 'name' is given. In addition to dependency information, other morphological information can be entered in the *4th* column. The initial node in the SSF with both morphological information and head computation for the preceding sentence would appear as follows

```
2.      (( NP<fs af='raamudu,n,m,s,3,,,0,du' drel='SUBJ:VGF'>
2.1     raamudu NN <fs af='raamudu,n,m,s,3,,,0,du'>
        ))
```

'af' above stands for abbreviated feature structure and represents the following info.

```
raamudu, n, m,  s,  3,  ,  0,  du
   |     |  |   |   |   |  |   |
 root   cat gen num per cas vib/tam suf
  (1)   (2) (3) (4) (5)  (6)  (7)  (8)
```

1. Root: The word's root form
2. Course-grained POS tag as a category
3. Gender: Masculine, Feminine, or Neutral
4. Singular/plural number
5. Person: first/s
6. **3.** econd/third
7. Case: Oblique/Direct
8. TAM/Vibhakti (prefixes/postpositions, etc) (tense, aspect, and modality)
9. Suffix: A word suffix

2.2 CoNLL format

An automated script converts the Telugu treebank to CoNLL format. CoNLL format is the standard format for dependency parsing in CoNLL Shared Tasks. This layout has ten columns. Table 2 contains a brief description of these columns.

Table 2 : Columns in Conll format

Field No.	Description	Name
1	Word form or punctuation symbol	FORM
2	Head of the current token, a value of ID or zero ('0')	HEAD
3	Fine-grained POS tag	POSTAG
4	Lemma or stem of word form	LEMMA
5	Coarse-grained POS tag	CPOSTAG
6	Dependency relation to the HEAD	DEPREL
7	An unordered set of syntactic and/or morphological features, separated by a vertical bar (|)	FEATS
8	Dependency relation to the PHEAD	PDEPREL
9	Token counter, starting at 1 for each new sentence	ID
10	Projective head of current token, which is either a value of ID or zero ('0')	PHEAD

The CoNLL format for the above example sentence (1) will be:

```
1  raamudu raamu   n    NNP   gen-m|num-s|per-3|cas-$|vib-0|suf-du$  4  SUBJ  -
2  oka     oka     dem  DEM   gen-$|num-s|per-$|cas-$|vib-0|suf-$    3  DEM   -
3  pamdu   pamdu   n    NN    gen-m|num-s|per-3|cas-$|vib-0jsuf-$     4  OBJ   -|
4  tinnadu tinu    v    VM    gen-m|num-s|per-3|cas-$|vib-nadu|suf-du 0  ROOT
```

Each row/line represents a sentence node. A blank line separates each statement. Unicode characters are supported by the format. But, in order to do tests, we transformed the Unicode data into wx-notation. The columns ID, FORM, CPOSTAG, POSTAG, HEAD, and DEPREL are required, while the others are optional. In the case of optional columns, an underscore indicates that the information is not accessible[20-22]. The other columns are fixed except for the FEATS column. But, in the FEATS column, we may store any important information that is not encoded in the fixed columns. Examining the fourth row of the CoNLL format displayed above, we can see that the node's ID is '4'. The term 'Tinnadu' is present, and the root form of the word is 'tinu'. The coarse-grained and fine-grained POS tags are represented by 'v' and 'VM', respectively. As this node serves as the root of the sentence, the HEAD and DEPREL are designated as '0' and 'ROOT', respectively, which are both fixed columns. The FEATS column can be utilized to convey additional information such as gender, number, person, case, vibhakti, and suffix.

3. Results and discussion

Table.3 shows the results of our step-by-step investigation of the influence of various characteristics on Telugu dependency parsing. We found that the liblinear learner and arc-eager parsing algorithms consistently outperformed other methods. In the table, we present the evaluation metrics, including Unlabeled Attachment Score (UAS) and Labeled Attachment Score (LAS), for each experiment.

Table. 3: Results of various features on parsing Telugu using MaltParser

Features	UAS	LS	LAS
Exp1:current FORM,POSTAG	74.1%	51.1%	48.1%
Exp2:Exp1+context FORM,POSTAG	86.1%	61.1%	59.4%
Exp3:Exp2+LEMMA	86.3%	63.3%	61.3%
Exp4:Exp3+CPOSTAG	87.7%	62.8%	61.4%
Exp5:Exp4+FEATS	88.7%	69.1%	66.8%
Exp6:Exp5+DEPREL	90.5%	70.5%	68.3%
Exp7:Exp6+partial Tree features	90.7%	71.8%	69.6%
Exp8:Ex7+Bi-gram features	**91.8%**	**72.3%**	**70.0%**

In our first experiment, we utilized the FORM and POSTAG of the current word as features, resulting in an accuracy of 74.1% in UAS and 48.1% in LAS. However, incorporating the FORM and POSTAG of context words (Exp2) increased both UAS and LAS by approximately 12%, demonstrating the value of context in parsing. Exp3 and Exp4 involved incorporating LEMMA and CPOSTAG features, respectively, which resulted in a 1.6% increase in UAS and a 2% improvement in LAS. Exp5 incorporated FEATS, which provide morphological information and led to a 1% improvement in UAS and a 5.4% increase in LAS. Given Telugu's rich morphology, morphological information is likely essential in parsing, particularly in determining appropriate dependence labels. In Exp6, we added dependency relations (DEPREL) to partially constructed trees, which resulted in a 1.8% increase in UAS and a 1.5% improvement in LAS. Incorporating partial tree (Exp7) and bi-gram (Exp8) features improved UAS by 1.3% and LAS by 1.7%, respectively. Overall, these studies culminated in a UAS performance of 91.8% and a LAS performance of 70.0%.

4. Conclusion

We began this article by discussing the linguistic features of the Telugu language. The Telugu dependency tree bank was then thoroughly defined. In our work, we use the CoNLL format. The data supplied includes both fine-grained and coarse-grained dependence designations. For our tests, we employed a fine-grained version. With MaltParser, we got cutting-edge performance of 91.8% in UAS and 70.0% in LAS.

Conflict of Interest

The authors declare no conflict of interest

References

[1] R. Begum, S. Husain, A. Dhwaj, D. Misra Sharma, L. Bai, and R. Sangal “Dependency annotation scheme for Indian languages” *In proceedings of the third international joint conference on natural language processing*, Hyderabad, India, 2008

[2] B. V. S. Kumari and R. R. Rao “Improving Telugu dependency parsing using combinatory

categorial grammar supertags", *ACM Transactions on Asian and Low-Resource Language Information Processing*, Vol. 14, No. 1, Article No. 3, 2015.

[3] B. V. S. Kumari and R. R. Rao "A Hybrid Dependency Parsing Approach for Telugu Language", *International Journal of Advanced Research in Computer Science and Software Engineering*, Vol. 5, No. 9, pp: 664-668, 2015.

[4] A. Bharati, R. Sangal, D. M. Sharma and L. Bai "AnnCorra: Annotating Corpora Guidelines for POS and Chunk Annotation for Indian Languages" In Technical Report (TR-LTRC-31), LTRC, IIIT-Hyderabad, 2006.

[5] A. Bharati, R. Sangal and D. M. Sharma SSF: Shakti Standard Format Guide. In Technical Report (TRLTRC-33), LTRC, IIIT-Hyderabad, 2007.

[6] B. V. S. Kumari and R. R. Rao "Improving Indian Language Dependency Parsing by Combining Transition-based and Graph-based Parsers", *International Journal of Computer Applications* Vol. 115, No. 5, pp. 13-17, 2015.

[7] A. Bharati, S. Husain, D. M. Sharma and R. Sangal "A Two-Stage Constraint Based Dependency Parser for Free Word Order Languages", *In Proceedings of the COLIPS International Conference on Asian Language Processing*, Chiang Mai, Thailand, 2008.

[8] B. V. S. Kumari, A. G. Prasaad, M. Susmitha, and R. Bhatnagar "Exploring Different Approaches for Parsing Telugu", *In proceedings Advances in Intelligent system and computing*, Vol: 921, pp-546-555, 2019.

[9] A. Bharati, P. Mannem and D. M. Sharma "Hindi Parsing Shared Task", *In Proceedings of COLING-2012, Workshop on Machine Translation and Parsing in Indian Languages*, IIT, Mumbai. India, 2012.

[10] B. V. S. Kumari and R. R. Ramisetty "Hindi Dependency Parsing using a combined model of Malt and MST", *In Proceedings of the Workshop on Machine Translation and Parsing in Indian Languages,* held at IIT, pp. 171-178, Mumbai in 2012.

[11] J. Bos, C. Bosco and A. Mazzei "Converting a Dependency Tree-bank to a Categorial Grammar Treebank for Italian", In M. Passarotti, Adam Przepiorkowski,´ *Proceedings of the Eighth International Workshop on Treebanks and Linguistic Theories*, pp. 27–38, Milan, Italy, 2009.

[12] R. R. Rajeswara and B. V. S. Kumari "Two approaches for incorporating linguistic constraints to improve the usability of Telugu dependency parser", *International Journal of Applied Pattern Recognition*, Vol. 3, No. 2 pp: 135-144, 2016.

[13] S. Buchholz and E. Marsi "CoNLL-X shared task on multilingual dependency parsing", *In Proceedings of the Tenth Conference on Computational Natural Language Learning*, pp.149–164, New York City, New York, 2006.

[14] A. Cahill, B. Gyawali and J. Bruno "Self-training for parsing learner text", *In Proceedings of the First Joint Workshop on Statistical Parsing of Morphologically Rich Languages and Syntactic Analysis of Non-Canonical Languages*, pp: 66–73, Dublin, Ireland, 2014.

[15] B. V. S. Kumari and R. R. Rao "Improving the Usability of Statistical Parsers by Incorporating Linguistic Constraints", *International Journal of Electronics Communication & Computer Engineering*, Vol. 4, No. 6, pp:102-105, 2013.

[16] D. Deepthi and B. V. S. Kumari "Identification and Recognition of Facial expression using CNN Algorithm", *Turkish Journal of Computer and Mathematics Education*, Vol. 12 No. 6, pp. 5496-5504, 2021.

[17] B. R. Ambati, P. Gadde and K. Jindal "Experiments in Indian Language Dependency Parsing", *In Proceedings of ICON09 NLP Tools Contest: Indian Language Dependency Parsing*, Hyderabad, India, 2009.

[18] R. Cakici "Automatic induction of a CCG grammar for Turkish", *In Proceedings of the ACL Student Research Workshop*, pp. 73–78, Ann Arbor, Michigan, 2005.

[19] R. Cakıcı "Parser Models for a Highly Inflected Language", PhD thesis, University of Edinburgh, UK, 2009.

[20] B. V. S. Kumari and R. R. Rao "Telugu dependency parsing using different statistical parsers", *Journal of King Saud University-Computer and Information Sciences*, Vol. 29, No. 1, pp: 134-140, 2017.

[21] S. Clark, and J. R. Curran. "Wide-coverage efficient statistical parsing with CCG and log-linear models." *Computational Linguistics*, Vol. 33,

No. 4, pp: 493-552, 2007.

[22] S. V. Kumar, M. Nagaratna and L. H. Marrivada, "Task scheduling in cloud computing using PSO algorithm. In Smart Intelligent Computing and Applications", *Proceedings of Fifth International Conference on Smart Computing and Informatics*, pp: 541-550, 2021.

***