# Dependency Parsing for Telugu Using Data-driven Parsers

Praveen Gatla

Assistant Professor

Department of Linguistics

Faculty of Arts

Banaras Hindu University

praveengatla@gmail.com

**Abstract**

In this paper, we have developed manually annotated Telugu corpora by following DS guidelines (2009) and experimented our Telugu dependency treebank data on the data-driven parsers like Malt (Nivre et al., 2007a) and MST (McDonald et al. 2006) for parsing Telugu sentences. In the dependency, we link the head and dependents with their dependency relations (drels) by giving *kāraka* and *non-kāraka* relations to them. Telugu annotated data contains token with their morph information, pos, chunk and the drels. We have used our final Telugu treebank data in CONLL format for parsing in malt and MST parsers. We evaluated the labeled attachment score (LAS), unlabeled attachment score (UAS) and labeled accuracy (LA) for both the parsers and also compared their score in case of dependency relation too. Finally, we evaluated the most frequent errors which occurred after parsing the sentences and explained them with relevant examples with appropriate linguistic analysis, so that we can improve the accuracy of parsers in our future research.

**Keywords:** Dependency Parsing, Parser, Data-driven parser, Telugu Dependency Treebank, Malt, MST, kāraka, non-kāraka.

## 1 Introduction

Parsing is the analysis of the grammatical structure of a sentence. **Dependency Parsing** or **Syntactic Parsing** is the task of recognizing a sentence by assigning a syntactic structure to it. There are different types of approaches to develop a parser. They are grammar-driven, data-driven and hybrid parsers. In the recent years, data-driven parsing is able to achieve a greater

amount of success because of the availability of annotated corpora. Data-driven parser needs huge amount of manually annotated data or dependency trees which is called as treebank. It consists of the representation of the dependency relations between words in a given sentence. Dependency trees are useful to develop parsers. Unlike English and other foreign languages, Telugu is a free-word-order language. The free-word-order languages can be handled better by using the dependency based framework other than the constituency based one (cf. Bharati *et al.,* 1995). Many data-driven parsers have been developed recently because of the availability of dependency treebanks in Indian languages (Hindi, Telugu, Bangla). There are efforts to develop parsers for Indian languages. In order to develop the parsers, huge amount of annotated data is developed for Indian languages like Hindi, Telugu, Bangle etc. The relevant works are Caroll (2000), Bharati, A. (2008, 2009, 2010, 2011) Joakam Nivre (2009), Prashanth Mannem (2009), Bharat Ram Ambati (2009), Meher Vijay and Y. Kalyan D. (2009), Sankar D. (2009), Aniruddha G. (2009), Sanjay Chatterji et al. (2009), Bharati,A. and Rajeev, S. (2009), Phani, G. (2010), Samar, H. (2011), Venkat, B. & Kumari, S. (2012), Karan, S. (2012), Raghu Pujitha, G. (2014) etc.

Similarly, we also made an effort to develop the lexical resources for Telugu i.e. treebank data. We have used two most popular data-driven parsers, namely, Malt (Nivre et al., 2007a) and MST (McDonald et al., 2006) to implement developed Telugu dependency treebank. We discuss about data collection in Section 2 and details of Telugu treebank in Section 3, briefly explained about Malt and MST parser experiment and result in Section 4. In Section 5, we discussed about the error analysis of Telugu parsed sentences and we conclude the paper with labeled, unlabeled attachment score and labeled accuracies with future work.

## 2 **Data Collection**

Here, we have used 2424 Telugu treebank data in which the source has been collected from different Telugu grammar books of Ramarao, C. (1975, 1990, 2002), Subrahmanyam, P.S. (1984, 2013), Krishnamurti, Bh. (1985, 1991, 2003, 2009), Ramakrishna Reddy, B. (1986), Subbarao, K.V. (2012), Arora, H (1990), Lakshmi Bai, B. (1990), Rani, Usha (1990), Vijayanarayana, B. (1990), Krishnamurti, Bh. and Sivananda Sarma, P. (2005), Ramanarsimham, P. (2006),

Umamaheshwara Rao, G. (2012), Rajeshwari, Sivuni (2012), Vishwanatham, K. (2007), Srinivas, Addanki (2012) and from Telugu corpus.

**3 Telugu Trebank**

Treebanking is the process of marking the syntactic or semantic relations between the two words or constituents in a sentence. Once, a text or corpus is annotated with the linguistic information then it is called as a parsed text. In order to build Telugu treebank, we extracted the sentences from various Telugu grammars. We followed DS guidelines (2009d) which are developed by Akshara Bharati. These guidelines are followed based on the pāṇinian grammar formalism. There are two types of relations. They are kāraka relations and non-kāraka relations. Firstly, kāraka relations are like kartā (k1), kartā samānādhikaraṇa (k1s), karma (k2), karma samānādhikaraṇa (k2s), goal or destination (k2p), karaṇa (k3), sampradāna (k4), anubhava kartā(k4a), apādāna (k5), viṣayādhikaraṇa (k7), dēśādhikaraṇa (k7p), kālādhikaraṇa (k7t) etc and non-kā:raka relations are like ṣaṣṭhī (r6), hētu 'cause-effect' (rh), prati 'direction' (rd), noun modifier (nmod), verb modifier (vmod), adverbs (adv), conjunct (ccof) etc. By following the above guidelines, we developed the 2424 Telugu treebank data (sentences). In this Telugu treebank data, we have incorporated linguistic knowledge in the form of morphological features which might improve the accuracy for parsing Telugu treebank. The main goal of the corpus based approach is to encode linguistic knowledge like morphologically rich features of Telugu language, it serves as a strong cue for a sentence to identify the syntactic relations between the words in a sentence.

4 **Experiment and Results**

**4.1 Malt Parser**

Nivre et al., (2007a) says that “Malt parser is a freely available implementation of the parsing models described in Malt parser. It implements the transition-based approach to dependency parsing. It has a transition system for mapping sentences to dependency trees and a classifier for predicting the next transition for every possible system configuration”. Malt parser uses three families of parsing algorithms. They are Nivre, Covington and Stack. So the various parsing algorithms which are provided by Malt are nivre/arc-eager, nivre/arc-standard, stackproj,

stackeager, covington projective, covington non-projective. We tested Telugu treebank with all the algorithms and found **stackproj** gave a better performance for parsing while **liblinear** works better than libsvm for learning**.** Niver et al. (2007a) "Malt parser uses features of the partially built dependency structure together with features of the (tagged) input string". It uses history-based feature models in predicting the next action in the dependency structure.

**4.2 MST Parser**

MST parser is a freely available implementation of the parsing models. McDonald et al. (2005) says that "graph-based parsing system in that core parsing algorithms can be equated to finding directed maximum spanning trees (either projective or non-projective) from a dense graph representation of the sentence". The basic idea of graph based parsing is to draw dependency graphs for a sentence. For non-projective parsing, MST uses Chu-Liu-Edmonds Maximum Spanning Tree algorithm and Eisner's algorithm for projective parsing. There are three different types of features used by MST parser. They are basic, extended, and second-order features. McDonald et al., (2005a) used online large margin learning as the learning algorithm for MST parser.

**4.3 Data Setting**

Here, In this phase, we divided the Telugu treebank into training and testing dataset by applying 5-fold cross validation. It has generated training dataset (9610 tokens) and testing dataset (2364 tokens). The training and testing datasets are having a unique DEPREL (Dependency Relations) label 'root' for tokens where HEAD=0.

**4.4 Evaluation and Result**

We evaluated the performance of our model via the standard Labeled Attachment Score (LAS), Unlabeled Attachment Score (UAS), Labeled Accuracy (LA) metrices and via precision, recall and fscore metrices.

We compared the accuracies of both Malt and MST parser and mentioned in Table 1. Malt and MST has scored LAS of 79.67% and 73.62% respectively. Malt also scored more score in case of LA. If we see the Unlabeled Attachment Score (UAS), though MALT has scored more but MST

also has given a better performance by scoring 91.44%. Both the parsers has scored more accuracy on the test data for UAS. By looking at the **Table 1,** it is clear that out of both the parsers, Malt has given better performance than MST.

| Parsers | **LAS** | **UAS** | **LA** |
|---|---|---|---|
| **MALT** | **79.67** | **92.35** | **82.45** |
| **MST** | 73.62 | 91.44 | 76.30 |

**Table 1** Result of both Malt and MST on testing dataset

In the below given Table 2 and 3 represents the performance of the kāraka and non-kāraka dependency relations on testing and parsed datasets of Malt and MST parsers respectively.

| **deprel** | **treebankcount** | **correctcounter** | **parsercount** | **recall** | **precision** | **fscore** |
|---|---|---|---|---|---|---|
| k1 | 417 | 369 | 457 | 88.49 | 80.74 | **84.44** |
| k1s | 43 | 32 | 49 | 74.42 | 65.31 | 69.54 |
| k2 | 241 | 203 | 269 | 84.23 | 75.46 | **79.60** |
| k2p | 42 | 35 | 40 | 83.33 | 87.5 | 85.36 |
| k2s | 19 | 7 | 17 | 36.84 | 41.18 | 38.88 |
| k4 | 55 | 42 | 68 | 76.36 | 61.76 | 68.28 |
| k4a | 27 | 6 | 9 | 22.22 | 66.67 | 33.33 |
| k7 | 21 | 9 | 18 | 42.86 | 50 | 32.26 |
| k7p | 42 | 31 | 43 | 73.81 | 72.09 | 72.93 |
| k7t | 78 | 57 | 65 | 73.08 | 87.69 | 79.72 |
| nmod | 40 | 10 | 14 | 25 | 71.43 | 37.03 |
| vmod | 175 | 161 | 194 | 92 | 82.99 | 97.55 |
| adv | 417 | 369 | 457 | 88.49 | 80.74 | **84.44** |
| ccof | 43 | 32 | 49 | 74.42 | 65.31 | 69.54 |
| r6 | 241 | 203 | 269 | 84.23 | 75.46 | **79.60** |
| rh | 42 | 35 | 40 | 83.33 | 87.5 | 85.36 |

**Table 2** Precision and recall DEPREL of 'kāraka and non-kāraka relation' on Malt Parser

| deprel | treebankcount | correctcounter | parsercount | recall | precision | fscore |
|---|---|---|---|---|---|---|
| k1 | 417 | 359 | 513 | 86.09 | 69.98 | **77.20** |
| k1s | 43 | 23 | 38 | 53.49 | 60.53 | 56.79 |
| k2 | 241 | 182 | 278 | 75.52 | 65.47 | **70.13** |
| k2p | 42 | 28 | 35 | 66.67 | 80 | 54.54 |
| k2s | 19 | 6 | 14 | 31.58 | 42.86 | 36.36 |
| k4 | 55 | 30 | 46 | 54.55 | 65.22 | 59.40 |
| k4a | 27 | 6 | 11 | 22.22 | 54.55 | 31.57 |
| k7 | 21 | 4 | 8 | 19.05 | 50 | 27.58 |
| k7p | 42 | 15 | 21 | 35.71 | 71.43 | 47.61 |
| k7t | 78 | 53 | 71 | 67.95 | 74.65 | 71.14 |
| nmod | 40 | 6 | 17 | 15 | 35.29 | 21. 05 |
| vmod | 175 | 165 | 209 | 94.29 | 78.95 | 85.94 |
| adv | 33 | 21 | 34 | 63.64 | 61.76 | 62.68 |
| ccof | 36 | 28 | 33 | 77.78 | 84.85 | 81.16 |
| r6 | 12 | 2 | 7 | 16.67 | 28.57 | 21.05 |
| rh | 27 | 6 | 11 | 22.22 | 54.55 | 31.57 |

**Table 3** Precision and recall DEPREL of 'kāraka and non-kāraka relation' on MST Parser

In the above mentioned ***Table 2 and 3***, consists of seven columns, among them the first column denotes dependency relation (deprel), second to seventh columns explains about the deprel occurrences and accuracies of both the parsers. The keywords in the first row of table 2 and 3 are mentioned below.

a) deprel- dependency relations

b) treebankcount - Total number of tokens in the test data.

c) correctcounter - The number of tokens were correctly identified in the parsed data

d) parsercount – Total number of tokens in the parsed data.

e) precision: correctcounter / parsercount.

f) recall: correctcounter / treebankcount.

g) fscore: It is the harmonic mean of precision and recall.

fscore = (2 × precision × recall) / ( precision + recall).

In both the parsers, the parser count has increased and decreased than treebank count. For example, treebank count of *k1* is *417* where as parser count has increased to *457* and *369* are *parsed correctly* in Malt similarly *parser count* is increased to *513* and *359* are *parsed correctly* in MST. In the same way, *treebank count* of *k4a* is *27* where as *parser count* has decreased to *9* and *6* are *parsed correctly* in Malt similarly *parser count* is decreased to *11* and 6 are *parsed correctly* in MST.

McDonald and Nivre, (2007) says that "Malt is good at short distance labeling and MST is good at long distance labeling". Telugu treebank has very less non-projective data. Telugu sentences which are having more than 10 words are less in our dataset because we have taken the sentences from different grammar books of Telugu. Because Malt performs better in short distance label, it has secured good score in '*k1*' and '*k2*' dependency label. In other short distance label cases also Malt parser performed better than MST parser. The next section discusses about the error analysis of Telugu Parsed sentences.

**5. Error Analysis of Telugu Parsed Sentences**

This section deals with the error analysis of Telugu parsed sentences. We have developed the Telugu treebank data (2424 sentences) and implemented by using the two data-driven parsers (Malt and MST)(Discussed in Section 4). The Telugu test data is categorized according to the output given by the parsers. The test was done by using the Malt evaluation tool which has an underlying automatic mechanism to extract the most frequent incorrect parsed output which are occurred in the Telugu parsed test dataset. These sentences are divided into two parts viz. correct parsed output and incorrect parsed output. The incorrect parsed output is taken into consideration for the error analysis of the incorrect parsed output. Parsers extracted the most seven frequent errors from the test data (output). They are '*k2*' (karma) instead of '*k1*' (kartā) 31 times, '*k1*' (kartā) instead of '*k2*' (karma) for 27 times, '*k4*' (sampradāna) instead of '*k4a*' (anubhava kartā) for 14 times, '*k1*' (kartā) instead of '*nmod*' (noun modifier) for 7 times, '*k1*' (kartā) instead of '*k7t*' (kālādhikaraṇa) for 7 times, '*k7p*' (dēśādhikaraṇa) instead of '*k7*' (adhikaraṇa) for 6 times,

'*vmod*' (verb modifier) instead of '*rh*' (hētu) for 11 times etc . Among these seven frequent errors, here we discuss the first frequent error and explained with appropriate linguistic explanation. The analysis of the first frequent incorrect parsed outputs is given below.

**5.1 "k1" instead of "k2": 27 times (Passive constructions)**

The example which is illustrated in this section is a passive construction in Telugu. Most of the time, we come across with passive constructions only in written Telugu, where as the same does not occur in spoken Telugu. To make passive constructions in Telugu, one has to use '-*baḍu*' with the verbal root and the postpositions '-*cēta*' for nouns in passive constructions. The examples are discussed below.

1. *bhārataṃ/*NNP vyāsuḍ*i*/NNP *raciṃpabaḍiṃdi/*VM. '*vyāsuḍi* was written by Vyasa'

The POS tag of the each word is separated by a slash(/). Here NNP- Proper Noun, VM- Finite Verb. The following dependency relations of the example1. Here, we marked the dependency relations between two inter chunks in a given sentence. Dependency relations of the example1 are as follows.

*k2*(*bhārataṃ*, *raciṃpabaḍiṃdi*)

*k1*(*vyāsuḍi*, *raciṃpabaḍiṃdi*)

*finite* verb(*raciṃpabaḍiṃdi*). Here we renamed *root* as *finite verb* for our convenient.

The parsed output of the above example1 is given by the Malt parser. Here, we highlighted the incorrect Telugu parsed output which is generated by malt evaluation tool.

**k1(*bhārataṃ*, *raciṃpabaḍiṃdi*)**

**k2(*vyāsuḍi*, *raciṃpabaḍiṃdi*)**

finite verb(***raciṃpabaḍiṃdi***).

In the above illustrated example *bhārataṃ vyāsuḍi cēta raciṃpabaḍiṃdi*. 'Bharatam was written by Vyasa'. Whenever the pattern, simple verb + -*baḍu* suffix with *cēta* a postposition occurs the parsed output is marked *vyāsuḍi cēta* as *k2(karma)* where actually should mark as

*k1(kartā)*. This kind of patterns were not trained by the parsers properly. The reasons may be the input data (trained data) might be having few number of such sentences with '***-baḍu'*** and '***-cēta'*** constructions or the parser could not learn this pattern properly. Hence the parser failed to learn this linguistic rule which is why *k1(kartā)* is marked as *k2(karma)*. If the above mentioned linguistic cue is given as training to the parsers, it may mark such kind of examples as *k1* accurately.

*2. aṃduvalla/*RP *nityaṃ/*NN *gāyatri*/NNP *japiṃcu*/VM. 'Because of, always chant Gayatri'

Here **RP-** Particle, **NNP-**Proper Noun, **VM-** Finite Verb. Here we marked the dependency relations between two inter chunks in a given sentence. Dependency relations of the above mentioned example is as follows.

rh*(aṃduvalla, japiṃcu*)

k7t*(nityaM, japiṃcu)*

k2(*gāyatri*, *japiṃcu*)

finite verb(*japiṃcu*)

The parsed output of the above example 2 is given by the Malt parser. Here, we highlighted the incorrect Telugu parsed output which is generated by malt evaluation tool.

**k1*(aṃduvalla, japiṃcu*)**

k7t*(nityaM, japiṃcu)*

**k1(*gāyatri*, *japiṃcu*)**

finite verb(*japiṃcu*)

The example 2 given above is classical Telugu sentence. Usually, an animate object always prefers the *accusative case marker* '***-ni'***. In the present example *aṃduvalla nityaṃ gāyatri japiṃcu, gāyatri* doesn't contain the *accusative case marker* which always comes for the animate object. When we come across such kind of sentences, we mark k-relations by using the syntactico-semantico relations accurately. However, the absence of a case marker (accusative

case marker/vibhakti) has become an issue to the parser to mark k-relations that is why *k2* is marked as *k1*.

**6. Conclusion**

In this paper, we first explored the two data-driven parsers Malt and MST. We have used the same Telugu treebank for both the parsers. The treebank has very less non-projective sentences nearly 1.56%. We developed the best model for Malt as well as for MST. We found overall Malt gives better performance than MST. Malt system secured labeled attachment score (LAS) of 79.67% where MST secured 73.62%. It has proved that Malt Parser performed better than MST in case of short distance label (Cf. McDonald and Nivre, 2007). Our future research will be proceeded to develop a hybrid system by combining both Malt and MST parsed output. We extracted the most 7 frequent incorrect parsed outputs and explained the frequently incorrect tested output (i.e. *k2 <k1*) with proper linguistic input. Apart from this, we also proved that linguistic knowledge will improve the performance of the data-driven parsers.

**Acknowledgements**

I thank Prof. G. Umamaheshwara Rao who encouraged me to build the Telugu parser. I thank Joakam Nivre and McDonald for developing parsers like Malt and MST respectively. Because of this kind of open resources, we could implement our Telugu treebank data to develop a Telugu parser. I convey my sincere thanks to Y. Vishwanath Naidu and Gouri Sahoo for their technical support.

**References**

Ambati, B.R., Gadde, P., Jindal, K., 2009. Experiments in Indian language dependency parsing. In: Proceedings of the ICON09 NLP Tools Contest: Indian Language Dependency Parsing. pp. 32–37.

Bharati, A., Chaitanya, V., Sangal, R., 1995. Natural Language Processing: A Paninian Perspective. Prentice-Hall of India, 65–106.

Bharati, A., Sangal, R., Sharma, D.M., Bai, L., 2006. AnnCorra: annotating corpora guidelines for POS and Chunk Annotation for Indian languages. In: Technical Report (TR-LTRC-31), LTRC, IIIT-Hyderabad.

Bharati, A., Sharma, D.M., Husain, S., Bai, L., Begum, R., Sangal, R., 2009. AnnCorra: TreeBanks for Indian Languages, Guidelines for Annotating Hindi TreeBank (version 2.0). <http://ltrc.iiit.ac.in/MachineTrans/research/tb/DS-guidelines/DS-guidelines-ver2-28-05-09.pdf>

Garapati, U.R., Koppaka, R. and Addanki, S., 2012. Dative case in Telugu: a parsing perspective. In *Proceedings of the Workshop on Machine Translation and Parsing in Indian Languages* (pp. 123-132).

G, U.Rao. 2012. Telugu Bhasha-Sanganam. Hyderabad: Potti Sriramulu Telugu University.

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

Kumari, B.V.S. and Rao, R.R., 2017. Telugu dependency parsing using different statistical parsers. Journal of King Saud University-Computer and Information Sciences, 29(1), pp.134-140.

Kesidi, S.R., Kosaraju, P., Vijay, M., Husain, S., 2010. A two stage constraint based hybrid dependency parser for Telugu. In: Proceedings of the ICON-2010 Tools Contest on Indian Language Dependency Parsing.

Krishnamurti, Bh. 2009. Studies in Telugu Linguistics. Hyderabad: C.P. Brown Academy.

Krishnamurti, Bh. and Gwyn, J.P.L. 1985. A Grammar of Modern Telugu. Delhi: Oxford University Press.

Krishnamurti, Bh. and Sarma, Sivananda. 2005 . A Basic Course in Modern Telugu. Hyderabad: Telugu Akademi.

McDonald, R., Pereira, F., Ribarov, K. and Hajič, J., 2005. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing* (pp. 523-530). Association for Computational Linguistics.

McDonald, R. and Pereira, F., 2006. Online learning of approximate dependency parsing algorithms. In *11th Conference of the European Chapter of the Association for Computational Linguistics*.

Nivre, J., et al. 2007b. Maltparser: a language-independent system for data-driven dependency parsing. Nat. Lang. Eng. 13 (2), 95–135.

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

Ramakrishna Reddy, B. 1986. Localist Studies in Telugu Syntax. Hyderabad: Osmania Campus.

Ramanarsimham, P. 2006. Adhunika Bhasalo 'ku', Bhasha. Pg 42-56. Hyderabad: Telugu Linguists' Forum.

Ramarao, C. 2011. Telugu Vakyam. Secunderabad: Kavya Publishing House.

Ramarao, C. 2002. Quest of Subject in Telugu Case for Language Studies edited by Swarya Lakshmi. pg-153-156. Hyderabad: Booklinks Corporation.

Srinivas, A. 2012. Telugu Bhasha-Vyakaranam. Hyderabad: Akruthi Offset Printers.

Subramanyam, P.S. 2002. Ba:lavya:karnamu of Paravastu Cinnaya Suri. Thiruvananthapuram : Dravidian Linguistics Association.

Subbarao, K.V. 2012. Lexical Anaphors in South Asian Languages, Coalescence. Bangalore: Mudranik Technologies Pvt. Ltd.