Journal of King Saud University – Computer and Information Sciences (2017) 29, 134–140

King Saud University

Journal of King Saud University – Computer and Information Sciences

www.ksu.edu.sa
www.sciencedirect.com

Telugu dependency parsing using different statistical parsers

B. Venkata Seshu Kumari [a,*], **Ramisetty Rajeshwara Rao** [b,1]

[a] *JNTUH, Hyderabad, Telangana, India*
[b] *Computer Science & Engineering, JNTU Kakinada, Andhra Pradesh, India*

Received 14 September 2014; revised 21 November 2014; accepted 24 December 2014
Available online 3 November 2015

KEYWORDS

Dependency parsing;
Telugu;
MSTParser;
MaltParser;
TurboParser;
ZPar

Abstract In this paper we explore different statistical dependency parsers for parsing Telugu. We consider five popular dependency parsers namely, MaltParser, MSTParser, TurboParser, ZPar and Easy-First Parser. We experiment with different parser and feature settings and show the impact of different settings. We also provide a detailed analysis of the performance of all the parsers on major dependency labels. We report our results on test data of Telugu dependency treebank provided in the ICON 2010 tools contest on Indian languages dependency parsing. We obtain state-of-the art performance of 91.8% in unlabeled attachment score and 70.0% in labeled attachment score. To the best of our knowledge ours is the only work which explored all the five popular dependency parsers and compared the performance under different feature settings for Telugu.

1. Introduction

Dependency parsing is the task of uncovering the dependency tree of a sentence, which consists of labeled links representing dependency relationships between words. Parsing is useful in major NLP applications like Machine Translation, Dialogue Systems, Question Answering, etc. This led to the development of grammar-driven, data-driven and hybrid parsers. Due to the availability of annotated corpora in recent years, data driven parsing has achieved considerable success. The availability of phrase structure treebank for English (Marcus et al., 1993) has seen the development of many efficient parsers.

Unlike English, many Indian (Hindi, Bangla, Telugu, etc.) languages are free-word-order and are also morphologically rich. It has been suggested that free-word-order languages can be handled better using the dependency based framework than the constituency based one Bharati et al. (1995). Due to the availability of dependency treebanks, there are several recent attempts at building dependency parsers. Two CoNLL shared tasks (Buchholz and Marsi, 2006; Nivre et al., 2007a) were held aiming at building state-of-the-art dependency parsers for different languages. Recently in two ICON tools contests (Husain, 2009; Husain et al., 2010), and Coling 2012 Hindi parsing shared task (Bharati et al., 2012), rule-based, constraint based, statistical and hybrid approaches were explored towards building dependency parsers for three Indian

* Corresponding author. Tel.: +91 9989308242.
E-mail addresses: venkataseshukumari@gmail.com (B.V.S. Kumari), rajaraob4u@gmail.com (R.R. Rao).
[1] Tel.: +91 9959559456.
Peer review under responsibility of King Saud University.

ELSEVIER Production and hosting by Elsevier

http://dx.doi.org/10.1016/j.jksuci.2014.12.006

languages namely, Telugu, Hindi and Bangla. In all these efforts, state-of-the-art accuracies are obtained by the popular data-driven parsers namely, MaltParser (Nivre et al., 2007b) and MSTParser (McDonald, 2006).

Among Indian languages, though there has been significant amount of work on dependency parsing Hindi, there is very little work on parsing Telugu. Most of the work in ICON 2010 tools contest for parsing Telugu used MaltParser. In this paper, we consider five popular dependency parsers, MaltParser, MSTParser, TurboParser, ZPar and Easy-First Parser. We provide related work in Section 2 and details of dependency parsing, Telugu language and the Telugu dependency treebank in Section 3. In Section 4, we experiment with different parser and feature settings and show the impact of different settings. Section 5 provides a detailed analysis of the performance of all the parsers on major dependency labels. We conclude with possible future directions in Section 6. We obtain state-of-the art performance of 91.8% in unlabeled attachment score and 70.0% in labeled attachment score. To the best of our knowledge ours is the only work which explored all the five popular dependency parsers and compared the performance under different feature settings for Telugu.

2. Related work

There has been significant amount of work on dependency parsing in the recent past. Though majority of the work is done on English language, there has been increasing interest in parsing other languages. CoNLL 2006 and 2007 Shared tasks (Buchholz and Marsi, 2006; Nivre et al., 2007a) introduced the task of multi-lingual dependency parsing. Different approaches were explored in these two shared tasks for parsing eighteen different languages: Arabic, Basque, Catalan, Chinese, Czech, Danish, Dutch, English, German, Greek, Hungarian, Italian, Japanese, Portuguese, Slovene, Spanish, Swedish, and Turkish. Three metrices: labeled attachment score (LAS), unlabeled attachment score (UAS) and label accuracy (LA) were used for evaluation. LAS is the percentage of tokens with both correct dependency head and correct dependency label. UAS is the percentage of tokens with correct dependency head and LA is the percentage of tokens with correct dependency label. Different techniques like data-driven vs. hybrid; single step vs. two-stage; transition based vs. graph based were explored. In all these efforts, state-of-the-art accuracies are obtained by MaltParser (Nivre et al., 2007b), a transition based parser and MSTParser (McDonald, 2006) a graph based parser.

Following CoNLL shared tasks, there were two ICON tools contests (Husain, 2009; Husain et al., 2010) aimed at parsing three Indian languages: Hindi, Telugu and Bangla. Different rule-based, constraint based, statistical and hybrid approaches were explored towards building dependency parsers. Kesidi et al. (2010) used a constraint based approach. The scoring function for ranking the base parsers is inspired by a graph based parsing model and labeling. Nivre (2009), Ambati et al. (2009) and Kosaraju et al. (2010) used MaltParser and explored the effectiveness of local morphosyntactic features, chunk features and automatic semantic information. Parser settings in terms of different algorithms and features were also explored. Ambati et al. (2009) explored the usefulness of MSTParser for parsing Indian languages. Zeman (2009) combined various well known dependency parsers forming a super parser by using a voting method.

Recently in Coling 2012 workshop on Machine Translation and Parsing in Indian Languages, Hindi parsing shared task was held with the latest Hindi dependency treebank (Bharati et al., 2012). In this shared task, in addition to experimenting with individual parsers, there were efforts at combining different parsers. McDonald and Nivre (2007) showed that MaltParser and MSTParser make different kinds of errors and combining both the parsers can result in better parsing performance. Following this idea, Kumari and Rao (2012) combined the output of MaltParser and MSTParser in an intuitive manner to extract pros of both the parsers. Kukkadapu et al. (2012) explored voting and blending techniques for parsing Hindi using MaltParser, MSTParser and TurboParser.

In this work, we explore five popular dependency parsers namely MaltParser (Nivre et al., 2007b), MSTParser (McDonald, 2006), TurboParser (Martins et al., 2009), ZPar (Zhang and Clark, 2011) and Easy-First Parser (Goldberg and Elhadad, 2010). MaltParser is a transition based parser whereas MSTParser is a graph based parser. TurboParser is also a graph based parser but uses integer linear programming technique for parsing in contrast to MSTParser which uses maximum spanning tree algorithms. Zpar is a shift-reduce parser similar to MaltParser but uses beam search unlike greedy search used by MaltParser. MaltParser and Zpar parse a sentence from left to right. But, Easy-First Parser use non-directional strategy for parsing where easier dependencies are resolved first and use them as features while resolving harder dependencies.

In addition to standard English Penn treebank data (Marcus et al., 1993), all these parsers were explored for CoNLL shared task data. Though average number of tokens in test data is around 5000 tokens, number of tokens in the training data varied from around 30 thousand tokens (Slovene) to 1.2 million tokens (Czech). Apart from the amount of training data, morphological richness and free word order nature posed greater challenges for the parsers. It has been observed that parsing performance is least for morphologically rich and/or free word order languages like Arabic, Turkish, etc (Buchholz and Marsi, 2006; Nivre et al., 2007a).

MaltParser and MSTParser are the two parsers which are widely explored in the dependency parsing literature. Though MaltParser is explored extensively for Telugu in ICON shared tasks, there is very little work on experimenting with MSTParser for Telugu. Kukkadapu et al. (2012) adapted TurboParser for Hindi but there is no work on adapting it for Telugu. There has been some work on exploring Zpar and Easy-First Parser for languages other than English (Zhang and Nivre, 2012; Goldberg and Elhadad, 2010). There is no work on adapting these parsers for Indian languages in general and Telugu in particular. So, ours is the first work on exploring TurboParser, Zpar and Easy-First Parser for Telugu. Also, most of the papers compare MaltParser, MSTParser and one of the TurboParser or Zpar or Easy-First parsers but not all of them. To the best of our knowledge, ours is the only work which explored all the five popular dependency parsers and compared their performance for any language in general and Telugu in particular.

3. Dependency Grammar

Dependency Grammar (DG) describes the syntactic structure of a sentence through dependency graphs. A dependency graph represents words and their relationship to syntactic modifiers using directed edges. These edges can be labeled with grammatical relations like Subject, Object, etc.

Dependency trees can either be projective or non-projective. As English is fixed word order language, most English sentences can be analyzed through projective trees. But, in free word order languages, like Czech, Hindi, Telugu, etc. non-projective dependencies are more frequent. Rich inflection systems reduce the demands on word order, leading to non-projective dependencies (McDonald, 2006). Fig. 1 shows the dependency tree for an example Telugu sentence.

In the following section we first describe the morphological and syntactic features of Telugu language. Then we provide the details of Telugu dependency treebank.

3.1. Telugu language

Telugu is one of the official languages of India and the 13th largest language in the world, with over 74 million speakers.[2] It is a morphologically rich and free word order language. It is also an agglutinative language where morphological information is available as suffix to the word rather than a separate lexical item. Fig. 2 shows different Telugu sentences describing the free word order nature and morphological richness of Telugu. Sentence 1 is a simple past sentence 'Ram ate a fruit'. Suffix 'du' of the verb 'tinnadu' (ate) is a masculine marker. In 2nd sentence, as the gender of the subject changed to feminine (Sita), suffix of the verb changed to 'dhi' which is a feminine marker. Suffix of a verb can change when the tense changes. For example, in the 3rd sentence as the tense changed to present continuous, suffix of the verb changed accordingly to 'tunnaadu'. Similar to verbs where suffix depends on different factors like tense, aspect, modality and subject's gender, suffix of the nouns represents case or preposition. For example, in sentence 4, 'Ram gave a fruit to Sita in the school', suffix 'ki' of 'sithaki' (to Sita) is the dative case marker and suffix 'lo' of 'patashalalo' (in School) is the marker for preposition 'in'. Sentence 5 gives an example of free word order nature in Telugu language. Though Subject-Object-Verb is the preferred word order, different word orders are possible in Telugu as in sentence 5, with Object-Subject-Verb order.

3.2. Telugu dependency treebank

Telugu dependency treebank released in ICON 2010 Tools contest (Husain et al., 2010) is used in our work. Data are annotated using the part-of-speech (POS) tagging, chunking and dependency annotation guidelines (Bharati et al., 2006; Bharati et al., 2009). The treebank consists of morphological information (root, coarse pos-tag, gender, number, person, case marking, suffix and TAM (tense, Aspect and Modality marker)), POS, chunk and dependency information. The dependency annotation follows a scheme that can be traced back to Paninian grammar (Bharati et al., 2009), known to be well-suited to modern Indian languages. The dependency

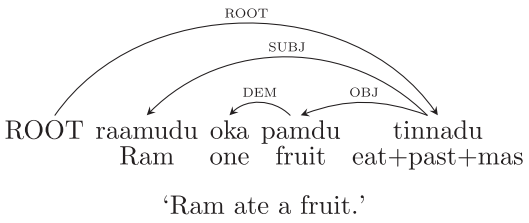

Figure 1 Dependency tree for an example Telugu sentence.

labels are syntactico-semantic in nature (Bharati et al., 1995; Bharati et al., 2009). For example, 'k1' usually corresponds to subject syntactic role and agent semantic role. Similarly, 'k2' corresponds to the syntactic role of object and the semantic role of patient. For the purposes of readability, instead of original treebank dependency labels, their corresponding English labels are used in this paper (SUBJ, OBJ, DEM for k1, k2, nmod__adj respectively).

The treebank is available in SSF (Bharati et al., 2007) and CoNLL[3] formats. We work with the CoNLL format in this paper. In this format, word, root, pos tag, chunk tag and morphological features are available in the FORM, LEMMA, POSTAG, and CPOSTAG and FEATS columns respectively. Data released have both fine-grained and coarse-grained versions of dependency labels. We used fine-grained version for our experiments. Table 1 shows the details of the training, development and the testing data sets of the Telugu dependency treebank. Statistics of sentence count, word count and average sentence length are provided in this table.

4. Experiments and results

We explore MaltParser, MSTParser, TurboParser, ZPar and Easy-First Parser for parsing Telugu. Exploring different feature and parser settings, we build best models for each parser. As the training data size is small, we merged training and development data and did 10-fold cross validation for tuning the parameters of the parsers and for feature selection. Best settings obtained using cross-validated data are applied on test set. We used standard Unlabeled Attachment Score (UAS), Labeled Attachment Score (LAS) and Labeled Score (LS) metrices for our evaluation.

4.1. MaltParser

MaltParser is a freely available implementation of the parsing models described in Nivre et al. (2007b).[4] It is a classifier based shift reduce parser. With MaltParser, parsing can be performed in linear time for projective dependency trees and quadratic time for arbitrary (possibly non-projective) trees. MaltParser provides options for nine deterministic parsing algorithms: Nivre arc-eager, Nivre arc-standard, Covington projective, Covington non-projective, Stack projective, Stack swap-eager, Stack swap-lazy, Planar and 2-planar. It also provides options for libsvm and liblinear learner algorithms. For Telugu dependency parsing liblinear learner and arc-eager parsing algorithm consistently gave better performance.

[2] http://www.ethnologue.com/statistics/size.

[3] http://nextens.uvt.nl/depparse-wiki/DataFormat.

[4] http://www.maltparser.org/.

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'

Figure 2 Telugu example sentences describing free word order nature and morphological richness.

Table 1 Telugu treebank statistics.

Type	Sent count	Word count	Avg. sent_length
Train	1400	7602	5.43
Devel	150	839	5.59
Test	150	836	5.57

We did a step-by-step analysis of the impact of different features on parsing Telugu. Table 2 provides results of these experiments. In Exp1, we provided word FORM and POSTAG of current word as features which gave an accuracy of 74.1% in UAS and 48.1% in LAS. Adding FORM and POSTAG of context words (Exp2) improved both UAS and LAS by around 12% which shows the importance of context in parsing. Adding LEMMA and CPOSTAG features (Exp3 and Exp4) gave a slight improvement of 1.6% in UAS and 2% in LAS. In Exp5, we added FEATS which contain morphological information and this gave 1% improvement in UAS and boosted LAS by 5.4%. As Telugu is a morphologically rich language, it is expected that morphological information plays a crucial role in parsing, especially in identifying correct dependency labels. In Exp6, we provided dependency relations (DEPREL) of the partially formed trees which gave an improvement of 1.8% in UAS and 1.5% in LAS. Adding partial tree (Exp7) and bi-gram (Exp8) features gave further improvements of 1.3% in UAS and 1.7% in LAS. After all these experiments, we achieved a performance of 91.8% in UAS and 70.0% in LAS.

4.2. MSTParser

MSTParser is a freely available implementation of the parsing models described in McDonald (2006).[5] It is a graph-based parsing system in which parsing algorithm is equated to finding directed maximum spanning trees from a dense graph

Table 2 Impact of different features on parsing Telugu using MaltParser.

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

of the sentence. MSTParser uses Chu–Liu–Edmonds maximum spanning tree algorithm for non-projective parsing and Eisner's algorithm for projective parsing. It uses online large margin learning as the learning algorithm (McDonald et al., 2005). It also provides options of 1st order and 2nd order features. 1st order features are the features over the parent and child in the dependency arc. These include different unigram, bigram features of parent node and child node. But, 2nd order features include more global features like grand parent, grand child and sibling features. For example, postag of parent node and child node are 1st order features whereas, postag of grand child and grand parent are second order features.

For Telugu, 2nd order features and non-projective algorithm gave the best results of 90.0% UAS and 62.6% LAS (Table 3, MSTParser: Baseline). It is difficult to do feature tuning with MSTParser as it doesn't provide nice options similar to MaltParser. We had to modify the code of MSTParser to add new features. Labeling module of the MSTParser is not using FEATS column. Exp5 in Table 2 clearly shows that morphological features in FEATS are very important for labeling in case of Telugu. We explored different features using

[5] http://mstparser.sourceforge.net/.

FEATS columns in the labeling module of the MSTParser and selected the settings which gave best results on 10-fold cross-validation. This gave a huge boost of 4.5% improvement in LAS over the baseline model (MSTParser: Extended in Table 3). With this tuning, we achieved a performance of 90.0% in UAS and 67.1% in LAS.

4.3. TurboParser

TurboParser is a freely available implementation of the parsing models described in Martins et al. (2009).[6] It is a graph based parser which uses integer linear programming technique for parsing. With default settings, we got an UAS of 90.5% and LAS of 67.5%. As the data are low and as the average sentence length is small, using standard model and considering only first order features gave better results. Final best results we could obtain are 91.2% UAS and 68.8% LAS. TurboParser also doesn't provide flexibility like MaltParser to add new features. Also, as the code is highly optimized for speed and performance, it was much harder to add any new features in the code as well. So, we could only explore parser settings, but couldn't do any feature ablation studies similar to MaltParser.

4.4. ZPar

ZPar is a freely available implementation of the parsing models described in Zhang and Clark (2011).[7] We used generic dependency parsing module of ZPar for our experiments. In addition to local features from the nodes in stack and input, it also uses higher order features like valency information, grand child and grand parent information (Zhang and Nivre, 2011). It uses arc-eager algorithm with beam search for parsing. Averaged perceptron (Collins, 2002) is used for learning. With default settings, we got an UAS of 90.0% and LAS of 68.0%. As the features are hard-coded, we modified the code and added features like partial tree features, similar to our experiments with MaltParser. Final best results we obtained after these feature ablation studies are 90.7% UAS and 68.5% LAS.

4.5. Easy-First Parser

Easy-First Parser is a freely available implementation of the parsing models described in Goldberg and Elhadad (2010).[8] Parsing algorithm is a shift-reduce style algorithm. But instead of traditional left to right parsing they employ non-directional strategy for parsing. A variant of structured perceptron (Collins, 2002) is used for learning. This parser only gives unlabeled dependencies. With default settings, we got an UAS of 86.8%. This parser takes feature templates from an input file which is similar to MaltParser. So, it was relatively easier to add new features. Similar to our experiments with MaltParser, we did feature ablation experiments. As this is only an unlabeled dependency parser, we couldn't explore the impact of dependency label features. Final best results we achieved are 88.8% UAS. (see Tables 4–7).

[6] http://www.ark.cs.cmu.edu/TurboParser/.
[7] http://sourceforge.net/projects/zpar/.
[8] http://www.cs.bgu.ac.il/yoavg/software/easyfirst/.

Table 3 Impact of different features on parsing Telugu using MSTParser.

Features	UAS (%)	LAS (%)	LS (%)
MSTParser: Baseline	90.0	62.6	63.9
MSTParser: Extended	90.0	67.1	68.6

5. Analysis

We achieved state-of-the-art performance of 91.8% in UAS and 70.0% in LAS using MaltParser. It has been observed that transition based parasers like MaltParaser are good at short distance dependencies and graph based parsers like MSTParser are good at long distance dependencies (McDonald and Nivre, 2007). As the majority of dependencies in Telugu treebank are short distance, MaltParser outperformed other parsers as it is good at short distance dependencies. As observed in Table 2, features play a crucial role in parsing and in case of MaltPaser, we can provide feature templates in a file as an input to the parser. This flexibility with MaltParser helped in providing different complex features which improved the performance of the parser. As features were hard-coded for other parsers (excluding Easy-First Parser), it was not very easy to explore the impact of different feature settings. This shows the importance of flexibility in providing features to a parser while adapting a parser for new language or treebank. We believe that as the treebank consists of a larger chunk of short distance dependencies graph based parsers like MSTParser, TurboPrser or beam search parsers like ZPar didn't give better improvements over MaltParser. It has been observed previously and we also observed in our experiments with MaltParser that providing dependency labels of the partially formed tree gave huge improvements of around 1.5–2.0% in both UAS and LAS (Exp6 in Table 2). As Easy-First Parser only does unlabeled parsing, we can't provide such information which could be the reason for relatively lower UAS values compared to other parsers.

Though MSTParser and TurboParser are graph based parsers, TurboParser, through integer linear programmaing techniques, efficiently incorporates linguistic constraints like a verb should have only one subject, children of a conjunct should be of similar type. This could be the reason for TurboParser giving second best results. ZPar uses global learning and beam search which is better for learning long distance dependencies. Due to availability of less number of long distance dependencies, global learning didn't give any improvement over greedy local learning of MaltParser. As the training data are very low, and also as Telugu is agglutinative language, LAS for the all the systems is very low. Though we could achieve an UAS of 91.8%, we could only achieve LAS of 70.0%. With more training data and specialized techniques for handling agglutinative languages like Telugu, we can achieve better results in LAS.

Table 8, gives an overview of the performance of the individual parsers for the top six dependencies in Telugu dependency treebank. MAIN, SUBJ, OBJ, COORD, TIME, and VMOD are the dependency labels for sentence root, subject, object, co-ordination, time expression, amd verbal modifier. MaltParser performed better for MAIN, COORD and TIME dependency labels. For SUBJ and VMOD labels TurboParser performed better and for OBJ label MSTParser performed

Table 4 Impact of different features on parsing Telugu using TurboParser.

Features	UAS (%)	LAS (%)	LS (%)
TurboParser: Baseline	90.5	67.5	69.0
TurboParser: Extended	91.2	68.8	70.1

Table 5 Impact of different features on parsing Telugu using ZPar.

Features	UAS (%)	LAS (%)	LS (%)
ZPar: Baseline	90.0	68.0	69.5
ZPar: Extended	90.7	68.5	70.3

Table 6 Impact of different features on parsing Telugu using Easy-First Parser.

Features	UAS (%)	LAS	LS
ZPar: Baseline	86.8	–	–
ZPar: Extended	88.8	–	–

Table 7 Performance of different parsers on Telugu dependency treebank test data.

Parser	UAS (%)	LAS (%)	LS (%)
MaltParser	**91.8**	**70.0**	**72.3**
MSTParser	90.0	67.1	68.6
TurboParser	91.2	68.8	70.1
ZPar	90.7	68.5	70.3
Easy-First	88.8	–	–

Best parser result is marked in bold.

Table 8 Performance of different approaches on top six dependency labels.

Labels	MaltParser	MSTParser	TurboParser	ZPar
MAIN	**97.0**	95.3	96.3	95.3
SUBJ	63.0	59.4	**64.1**	63.0
OBJ	58.8	**62.7**	59.9	61.1
COORD	**83.1**	74.4	77.6	77.5
TIME	**61.2**	60.5	57.8	46.5
VMOD	62.7	65.1	**66.7**	63.3

Best parser result is marked in bold.

better. TurboParser can handle linguistic constraints better which could be the reason for better handling of subject (SUBJ) and verbal modifier (VMOD) labels. As MaltParser is a greedy transition based parser, it is subject to error propagation. But MSTParser doesn't have this problem which could be the reason for better handling of OBJ label.

6. Conclusion and future work

Experimenting with different settings, we built best parsing models for Telugu using five popular dependency parsers namely MaltParser, MSTParser, TurboParser, ZPar and Easy-First Parser. We studied the impact of different features for parsing Telugu. We also provided a detailed analysis of the performance of all the parsers on major dependency labels. For morphologically rich languages like Telugu, our experiments showed that providing morphological features can significantly improve the parsing performance. We obtained state-of-the art performance of 91.8% in unlabeled attachment score and 70.0% in labeled attachment score. Our experiments pointed out two important things which should be considered while creating annotated data and building parsers. Data sets should be representative of real world sentences in that language. Real world Telugu sentences will have a good mixture of short distance and long distance dependencies and not just contain short distance dependencies. From engineering point of view, while developing a parser, it is better to provide flexibility to add or remove features. This would help in adapting parsers to new data sets.

There has been increasing interest on parsing morphologically rich languages. Different approaches are being explored in the Statistical Parsing of Morphologically Rich Languages (SPMRL) workshops (Seddah et al., 2010; Seddah et al., 2011; Apidianaki et al., 2012; Goldberg et al., 2013; Goldberg et al., 2014). Great deal of work is done on incorporating morphological features into different kinds of parsers. There has been some work on exploring the usefulness of large un-annotated data using self-training and co-training techniques (Goutam and Ambati, 2012; Cahill et al., 2014) and using word vectors (Seddah et al., 2014). Though the size of Telugu treebank is small, there is lot of un-annotated Telugu text available. It would be an interesting direction to explore these techniques for Telugu.

There has also been some interesting work going on improving the greedy parsers like MaltParser by incorporating better features from other grammatical frameworks (Ambati et al., 2014) and exploring better learning and parsing algorithms (Sartorio et al., 2013; Goldberg and Nivre, 2013). We would like to see the impact of such approaches for parsing Telugu.

References

Ambati, B.R., Deoskar, T., Steedman, M., 2014. Improving dependency parsers using combinatory categorial grammar. In: Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics, Vol. 2, Short Papers. Gothenburg, Sweden, pp. 159–163.

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.

Apidianaki, M., Dagan, I., Foster, J., Marton, Y., Seddah, D., Tsarfaty, R. (Eds.), 2012. Proceedings of the ACL 2012 Joint Workshop on Statistical Parsing and Semantic Processing of Morphologically Rich Languages. JEJU, Republic of Korea.

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

Bharati, A., Mannem, P., Sharma, D.M., 2012. Hindi parsing shared task. In: Proceedings of Coling Workshop on Machine Translation and Parsing in Indian Languages. Kharagpur, India.

Bharati, A., Sangal, R., Sharma, D.M., 2007. SSF: shakti standard format guide. In: Technical Report (TR-LTRC-33), LTRC, IIIT-Hyderabad.

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>

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

Cahill, A., Gyawali, B., Bruno, J., 2014. 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. Dublin City University, Dublin, Ireland, pp. 66–73.

Collins, M., 2002. Discriminative training methods for hidden markov models: theory and experiments with perceptron algorithms. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing. EMNLP '02. pp. 1–8.

Goldberg, Y., Elhadad, M., 2010a. Easy first dependency parsing of modern hebrew. In: Proceedings of the NAACL HLT 2010 First Workshop on Statistical Parsing of Morphologically-Rich Languages. SPMRL '10. Association for Computational Linguistics, Stroudsburg, PA, USA, pp. 103–107.

Goldberg, Y., Elhadad, M., 2010b. An efficient algorithm for easy-first nondirectional dependency parsing. In: Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics. Los Angeles, California.

Goldberg, Y., Marton, Y., Rehbein, I., Versley, Y. (Eds.), 2013. Proceedings of the Fourth Workshop on Statistical Parsing of Morphologically-Rich Languages. Seattle, Washington, USA.

Goldberg, Y., Marton, Y., Rehbein, I., Versley, Y., Özlem Çetinoğlu, Tetreault, J., (Eds.), 2014. Proceedings of the First Joint Workshop on Statistical Parsing of Morphologically Rich Languages and Syntactic Analysis of Non-Canonical Languages. Dublin, Ireland.

Goldberg, Y., Nivre, J., 2013. Training deterministic parsers with non-deterministic oracles. In: Transactions of the Association for Computational Linguistics.

Goutam, R., Ambati, B., 2012. Exploring self-training and co-training for dependency parsing. In: Proceedings of the 13th International Conference on Intelligent Text Processing and Computational Linguistics. New Delhi, India.

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.R., Gadde, P., 2010. The ICON-2010 tools contest on Indian language dependency parsing. In: Proceedings of ICON-2010 Tools Contest on Indian Language Dependency Parsing. Kharagpur, India.

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.

Kosaraju, P., Kesidi, S.R., Ainavolu, V.B.R., Kukkadapu, P., 2010. Experiments on Indian language dependency parsing. In: Proceedings of the ICON-2010 Tools Contest on Indian Language Dependency Parsing.

Kukkadapu, P., Malladi, D., Dara, A., 2012. Ensembling various dependency parsers: adopting turbo parser for Indian languages. In: Proceeding of Coling 2012 Workshop on MT and Parsing in Indian Languages.

Kumari, B.V.S., Rao, R.R., 2012. Hindi dependency parsing using a combined model of Malt and MST. In: Proceeding of Coling 2012 Workshop on MT and Parsing in Indian Languages.

Marcus, M.P., Santorini, B., Marcinkiewicz, M.A., 1993. Building a large annotated corpus of English: the Penn Treebank. Comput. Linguist. 19 (2), 313–330.

Martins, A., Smith, N., Xing, E., 2009. Concise integer linear programming formulations for dependency parsing. In: Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP. Suntec, Singapore, pp. 342–350.

McDonald, R., 2006. Discriminative learning and spanning tree algorithms for dependency parsing (Ph.D. thesis), Philadelphia, PA, USA.

McDonald, R., Crammer, K., Pereira, F., 2005. Online large-margin training of dependency parsers. In: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics. Ann Arbor, Michigan, pp. 91–98.

McDonald, R., Nivre, J., 2007. Characterizing the errors of data-driven dependency parsing models. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing and Natural Language Learning.

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

Nivre, J., Hall, J., Kübler, S., McDonald, R., Nilsson, J., Riedel, S., Yuret, D., 2007a. The CoNLL 2007 shared task on dependency parsing. In: Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007. Prague, Czech Republic, pp. 915–932.

Nivre, J., Hall, J., Nilsson, J., Chanev, A., Eryigit, G., Kübler, S., Marinov, S., Marsi, E., 2007b. Maltparser: a language-independent system for data-driven dependency parsing. Nat. Lang. Eng. 13 (2), 95–135.

Sartorio, F., Satta, G., Nivre, J., 2013. A transition-based dependency parser using a dynamic parsing strategy. In: Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (vol. 1: Long Papers). Sofia, Bulgaria, pp. 135–144.

Seddah, D., Koebler, S., Tsarfaty, R., (Eds.), 2010. Proceedings of the NAACL HLT 2010 First Workshop on Statistical Parsing of Morphologically-Rich Languages. Los Angeles, CA, USA.

Seddah, D., Kübler, S., Tsarfaty, R., 2014. Introducing the spmrl 2014 shared task on parsing morphologically-rich languages. In: Proceedings of the First Joint Workshop on Statistical Parsing of Morphologically Rich Languages and Syntactic Analysis of Non-Canonical Languages. Dublin City University, Dublin, Ireland, pp. 103–109.

Seddah, D., Tsarfaty, R., Foster, J., (Eds.), 2011. Proceedings of the Second Workshop on Statistical Parsing of Morphologically Rich Languages. Dublin, Ireland.

Zeman, D., 2009. Maximum spanning malt: hiring world's leading dependency parsers to plant Indian trees. In: Proceedings of the ICON09 NLP Tools Contest: Indian Language Dependency Parsing.

Zhang, Y., Clark, S., 2011. Syntactic processing using the generalized perceptron and beam search. Comput. Linguist. 37 (1), 105–151.

Zhang, Y., Nivre, J., 2011. Transition-based dependency parsing with rich nonlocal features. In: Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies. Portland, Oregon, USA, pp. 188–193.

Zhang, Y., Nivre, J., 2012. Analyzing the effect of global learning and beam-search on transition-based dependency parsing. In: Proceedings of COLING 2012: Posters. Mumbai, India, pp. 1391–1400. URL: <http://www.aclweb.org/anthology/C12-2136>.