Maximum Spanning Malt: Hiring World's Leading Dependency Parsers to Plant Indian Trees

Daniel Zeman
Univerzita Karlova v Praze, Ústav formální a aplikované lingvistiky
Malostranské náměstí 25, CZ-11800, Praha, Czechia
zeman@ufal.mff.cuni.cz

Abstract

We present our system used for participation in the ICON 2009 NLP Tools Contest: dependency parsing of Hindi, Bangla and Telugu. The system consists of three existing, freely available dependency parsers, two of which (MST and Malt) have been known to produce state-of-the-art structures on data sets for other languages. Various settings of the parsers are explored in order to adjust them for the three Indian languages, and a voting approach is used to combine them into a superparser. Since there is nothing novel about the approach used, substantial part of the paper is devoted to the analysis of errors the system makes on the given data sets.

1 Introduction

Dependency parsing, i.e. sentence analysis that outputs tree of word-on-word dependencies (as opposed to constituent trees of context-free derivations), gained growing attention and popularity recently. There are data-driven dependency parsers that can be trained on syntactically annotated corpora (treebanks) and new, previously unseen material can be parsed very efficiently (Nivre, 2009).

Most of the successful parsers employ discriminative learning techniques to sort out vast sets of potentially useful features observed in the input text. Thus, for every new training treebank, smart feature engineering is the key to getting the most out of the existing parsers, regardless how well they performed on other data sets and languages. Now that there are new treebanks available for two Indo-Aryan and one Dravidian language, we took three existing dependency parsers and explored the possibilities of tuning them for the new training data. Both parser configuration and data preprocessing are relevant approaches to the tuning. In addition, we used parser combination to further improve the results.

Throughout the paper we focus mainly on the unlabeled attachment score. Although the parsers produce labeled dependencies, we do not optimize the system towards label accuracy.

The rest of the paper is organized as follows: In Section 2, we describe the respective parsers and the combined parsing system. In Section 3, we report on the experiments we performed, discuss various results on the development set and analyze the errors. In Section 4 we present the official results on the test data. We conclude by summarizing the best configuration we were able to find, and future implications.

2 System Description

Several good trainable dependency parsers have emerged during the past five years. The CoNLL-X (Buchholz and Marsi, 2006) and CoNLL 2007 (Nivre et al., 2007a) shared tasks in multilingual dependency parsing have greatly contributed to the development of the parsers. Some of the parsers are now freely available on the web, some are even open-source. We selected three of the publicly available parsers for our experiments:

2.1 MST Parser

The Maximum Spanning Tree (MST) parser (McDonald et al., 2005) views the sentence as a directed complete graph with edges weighted by a feature scoring function. It finds for the graph the spanning tree that maximizes the weights of the edges. A multi-class classification algorithm called MIRA is used to compute the scoring function.

MST Parser achieved the best unlabeled attachment scores (UAS) for 9 out of the 13 languages of CoNLL-X, and second best scores in two others. Parsing is fast but training the parser takes many hours on large treebanks. On small data however,

Proceedings of ICON-2009: 7th International Conference on Natural Language Processing, Macmillan Publishers, India. Also accessible from http://ltrc.iiit.ac.in/proceedings/ICON-2009

multiple quick experiments with different settings are still doable. The parser is implemented in Java and freely available for download.[1]

2.2 Malt Parser

The Malt Parser (Nivre et al., 2007b) is a deterministic shift-reduce parser where input words can be either put to the stack or taken from the stack and combined to form a dependency. The decision which operation to perform is made by an oracle based on various features of the words in the input buffer and the stack. The default machine learning algorithm used to train the oracle is a sort of SVN (support vector machine) classifier (Cristianini and Shawe-Taylor, 2000).

Malt Parser has participated in both CoNLL-X and CoNLL 2007 shared tasks, and although it achieved the best UAS in three languages only, it usually scored among the five best parsers, sometimes with statistically insignificant difference from the winner. Malt Parser is really fast and its new Java implementation is open-source, freely available for download.[2]

2.3 DZ Parser

In order to combine the two above parsers, we needed a third parser. We picked DZ Parser (Zeman, 2004), which is also reasonably fast and freely available.[3] Although its accuracy, if compared to MST or Malt, is worse by a wide margin, this parser proved useful because its only role was to help to form a majority whenever MST and Malt disagreed.

DZ Parser builds a model of bigrams of words that occur together in a dependency; most of the time, words are identified by their part of speech tags and morphological features. The parser was originally developed for Czech but it can be retrained for any other language.[4]

2.4 Voting Superparser

The three parsers are combined using a simple weighted-voting approach similar to Zeman and Žabokrtský (2005), except that the output is guaranteed to be cycle-free. We start by evaluating every parser separately on the development data. The UAS of each parser is subsequently used as the weight of that parser's vote. Dependencies are parent-child relations, and for every node there are up to three candidates for its parent (if all three parsers disagree). Candidates get weighted votes – e.g., if parsers with weights $w_1 = 0.8$ and $w_2 = 0.7$ agree on the candidate, the candidate gets 1.5 votes. Since we have only three parsers, in practice this means that the candidate of the best parser looses only if 1. the other two parsers agree on someone else, or 2. if attaching the child to this candidate would create a cycle.

The tree is constructed from the root down. We repeatedly add nodes whose winning parent candidates are already in the tree. If none of the remaining nodes meet this condition, we have to break a cycle. We do so by examining all unattached nodes. At each node we note the votes of its current winning parent. Then we remove the least-scoring winner and go on with adding nodes until all nodes are attached or there is another cycle to break.

3 Experiments

The final test data are blind, any error analysis is therefore impossible. That is why all scores given in this section were measured on the development data. All three treebanks follow the same annotation scheme and each of them is available in two flavors:

- *nomorph* variety contains word forms, chunk labels, dependency links and dependency labels
- *morph* variety is augmented by automatically assigned lemmas, part of speech tags and values of morphological features (gender, number, person, case, postposition and tam – tense+aspect+modality)

3.1 Morphology

Table 1 shows baseline results on the *nomorph* data. Both MST and Malt parsers were invoked in projective mode, Malt with the default Nivre arc-eager algorithm.

There are several ways how to use the additional information from the *morph* data. The easiest way, exploitable by all three parsers, is to combine the chunk label, the POS tag and the features into one tag string. The results (not presented here) are very

[1] `http://sourceforge.net/projects/mstparser/`

[2] `http://maltparser.org/`

[3] `http://ufal.mff.cuni.cz/~zeman/projekty/parser/`

[4] Of course there are other dependency parsers that successfully participated in the CoNLL shared tasks and are available for download. One alternative worth mentioning is the ISBN Parser (Titov and Henderson, 2007) at `http://flake.cs.uiuc.edu/~titov/`.

	MST	Malt	DZ	Vote
hi	80.32	81.84	62.00	**82.48**
bn	82.00	**84.71**	71.02	83.11
te	77.63	**80.89**	70.52	80.59

Table 1: Baseline UAS of the four parsers on *nomorph* development data. Language codes follow ISO 639: hi = Hindi, bn = Bangla, te = Telugu.

poor. Although there may be tagging errors, the most likely cause is data sparseness. In Table 2 we illustrate this by showing the numbers of unique values in the various attributes of treebank words.

	occ	frm	lem	cl	pos	feat
hi	13779	3973	3134	10	33	714
bn	6449	2997	2336	14	30	367
te	5494	2462	1403	12	31	453

Table 2: Size of the training corpora: occ – word occurrences, frm – distinct forms, lem – lemmas, cl – chunk labels, pos – part of speech tags, feat – feature value combinations.

To fight sparseness, we could either restrict the tag to selected information, or split the information into multiple features learnt separately, or both. We first restricted the tag string to selected information. Parsing on other treebanks showed that POS with case is especially useful (Zeman, 2004). The other feature we selected is called *vibhakti*, which partially corresponds to case suffix and partially to postposition.[5]

Table 3 presents the results of restricting the tag string to POS+case+vibhakti. This move especially improves the MST Parser, which now outperforms Malt on Hindi and Bangla. Malt improves on Hindi but drops behind on Bangla and Telugu. DZ Parser also improves on Hindi and deteriorates elsewhere but there is an interesting observation: even though its own scores are worse, the worsened output actually improves the voting results (provided the configurations of MST and Malt are fixed). So it seems that the newly introduced errors are less important (because taken care of by the more powerful parsers) while some difficult parts of the data are now covered better.

Finally, ve returned to *nomorph* with Malt on Bangla and Telugu, where the POS+case+vibhakti tags did not help. The results are given in Table 4.

[5] The Indian treebanks at hand are unusual in that nodes do not always map to words. They represent chunks, with function words such as postpositions hidden in node attributes.

	MST	Malt	DZ	Vote
hi	86.16	85.84	75.12	**87.12**
bn	85.70	77.31	54.38	**85.82**
te	**79.85**	77.78	45.78	79.70

Table 3: UAS on refined *morph* data (POS tag, case and vibhakti concatenated).

This is also the configuration we used to parse the test set for the official evaluation round 1.

	MST	Malt	DZ	Vote
hi	86.16	85.84	75.12	**87.12**
bn	85.70	84.71	54.38	**86.19**
te	79.85	80.89	45.78	**82.37**

Table 4: UAS on mixed data: MST and DZ use POS+case+vibhakti for all languages, Malt uses that for Hindi only, elsewhere it uses just POS.

3.2 Nonprojectivity

Nonprojectivity is a property of the dependency structure and the word order (Hajičová et al., 2004) that makes parsing more difficult. All three parsers can produce nonprojective structures and all three treebanks are nonprojective. However, except for Hindi, the proportion of nonprojective dependencies is so small that one can hardly imagine that running the parsers in nonprojective mode would bring any improvement. A quick experiment with Malt Parser switched to the nonprojective Stack Eager algorithm revealed that it actually hurts the results even for Hindi.

	Edges	Sentences
hi	01.83	13.93
bn	00.96	05.49
te	00.45	01.31

Table 5: Percentage of nonprojective dependencies, and of sentences containing at least one nonprojectivity.

3.3 Error Patterns

The accuracy of the dependencies is relatively high and it is difficult to trace repetitive error patterns. In Hindi, many wrong attachments seem to be long-distance, and verbs, conjunctions, root and NULL nodes are frequently involved. Frequent words should perhaps be available to the parsers as parts of tag strings: for instance, Hindi कि *(ki)*

"that" or तो *(to)* are wrongly attached because the parser only sees the general CC tag. On a similar note, problems with coordination, also observed e.g. by Zeman (2004), occur here, too: भाई और भाभी *(bhāī aura bhābhī)* "brother and his wife" is correctly recognized as coordination rooted by the conjunction और, however, the conjunction node lacks the information about its noun children and fails to attach as the subject of the verb.

The tag string should contain both the chunk label and the POS tag. So far we wrongly assumed that POS always determines the chunk label. It is often so but not always, as exemplified in the Bangla chunk sequence তবে সুদীপ ওকে একদিন আড়ালে ডেকে বলেছিল কৌতূহল দেখালে তুমি উঁচুতে উঠতে অনিমেষ *(tabe sudīpa oke ekadina āṛāle ḍeke balechila kautūhala dekhāle tumi uṁcute uṭhate animeša)*. The words ডেকে and দেখালে are tagged VGNF|VM while বলেছিল and উঠতে are VGF|VM. The parser gets them wrong and it could be caused by it seeing only VM in the tag.

In Telugu, extraordinal number of sentences follow the SOV order so strongly that the last node (verb) is almost always attached to the root and most other nodes are attached directly to the last node. An example chunk sequence where this rule would lead to 100 % accuracy follows: రాష్ట్రంలో రంగారెడ్డి మెదక్ నిజామాబాద్ జిల్లాలలో పంటను గొప్పా పండిస్తున్నారు *(rāṣṭraṁlo raṁgāreḍḍi medak nijāmābād jillālalo paṁṭanu goppo paṁḍistunnāru)*. In the light of such examples it seems reasonable to provide the parsers with an additional feature telling whether a particular dependency observes the "naïve Telugu" structure. Note however that this will not help with the other two languages. While 73.75 % of Telugu dependncies follow this rule, it is only 39.52 % in Bangla and 35.71 % in Hindi.

3.4 Voting Potential

In order to see how much can be potentially gained from parser combination, we summarized the attachments that at least one of the parsers got correct. This *oracle accuracy* gives an upper limit for the real scores we can achieve. It corresponds to the case that for every word, an oracle correctly tells which parser to ask about the word's parent. Table 6 presents the oracle accuracies together with percentage of unique correct attachments that only one parser delivered. These figures give some idea of how much similar are the errors of the respective parsers to each other. Malt parser has the most unique know-how in all three languages, which could be explained by its focus on local features. Both MST and DZ can reach for global, sentence-wide relations. Note however, that the development data set is small and the percentages correspond to 42 (Malt/Bangla) or less words.

	Oracle	**UqMST**	**UqMalt**	**UqDZ**
hi	93.92	2.96	**3.12**	1.84
bn	94.20	4.32	**5.18**	1.97
te	88.00	2.37	**5.48**	2.07

Table 6: Oracle accuracy for the three languages, and unique correct attachments (%) proposed by a single parser.

4 Official Evaluation

Finally, we present the official evaluation of our voting superparser, as measured by the organizers on the test data. For this purpose, the parsing system has been retrained on both the training data and the development data. The results are shown in Table 7.

	UAS	**LAA**	**LAS**
hi	88.58 (3:90.31)	72.66 (4:76.38)	68.60 (4:74.48)
bn	86.06 (4:90.32)	71.28 (4:81.27)	66.70 (5:79.81)
te	80.27 (4:86.28)	54.20 (4:61.58)	49.91 (4:60.55)

Table 7: Official scores on the test data: unlabeled attachment score (UAS), label assignment accuracy (LAA) and labeled attachment score (LAS). The numbers in parentheses are the rank of our system and the score of the best system w.r.t. the given metric.

5 Related and Future Work

There is a large body of work on parser combination. A summary can be found in Nivre and McDonald (2008), whose approach is also related to ours w.r.t. the selection of parsers. However, their feature-based integration of MST and Malt parsers is much more sophisticated than our lightweight voting. Further improvement of accuracy can be expected if MST-Malt integration is applied to the Indian treebanks.

Future work, at the time of writing, includes experiments that can be run before the evaluation round 2, and thus their results will appear in the final version of this paper. We definitely intend to test other algorithms of the Malt parser, as well as

to reconfigure its feature pool and let it work with POS, case and vibhakti separately.

Labeling of the dependencies is another problem that deserves more attention. We have concentrated on the unlabeled attachment score so far and for the sake of the official evaluation, we simply pushed the MST labels through. A separate postprocessing classifier would probably produce better results.

6 Conclusion

We have described our system of voting parsers, as applied to the ICON 2009 NLP Tools Contest task. We showed that case and vibhakti are important features at least for parsing Hindi while their usability in Bangla and Telugu is limited by data sparseness. Providing these features to MST and DZ in all languages, and to Malt in Hindi only yielded the best combined parser. We also discussed several error patterns that could lead to further improvements of the parsing system in future.

Acknowledgements

We are enormously grateful to the developers of MST and Malt parsers for making their software available to the research community. The research has been supported by the grant MSM0021620838 (Czech Ministry of Education).

References

Sabine Buchholz and Erwin Marsi. Conll-x shared task on multilingual dependency parsing. In *Proceedings of the Tenth Conference on Computational Natural Language Learning (CoNLL-X)*, pages 149–164, New York City, June 2006. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/W/W06/W06-2920`. 2

Nello Cristianini and John Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods*. Cambridge University Press, Cambridge, UK, 2000. 2.2

Eva Hajičová, Jiří Havelka, Petr Sgall, Kateřina Veselá, and Daniel Zeman. Issues of projectivity in the prague dependency treebank. *The Prague Bulletin of the Mathematical Linguistics*, 81, 2004. 3.2

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 523–530, Vancouver, British Columbia, Canada, October 2005. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/H/H05/H05-1066`. 2.1

Joakim Nivre. Non-projective dependency parsing in expected linear time. 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*, pages 351–359, Suntec, Singapore, August 2009. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/P/P09/P09-1040`. 1

Joakim Nivre and Ryan McDonald. Integrating graph-based and transition-based dependency parsers. In *Proceedings of ACL-08: HLT*, pages 950–958, Columbus, Ohio, June 2008. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/P/P08/P08-1108`. 5

Joakim Nivre, Johan Hall, Sandra Kübler, Ryan McDonald, Jens Nilsson, Sebastian Riedel, and Deniz Yuret. The CoNLL 2007 shared task on dependency parsing. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*, pages 915–932, Praha, Czechia, June 2007a. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/D/D07/D07-1096`. 2

Joakim Nivre, Johan Hall, Jens Nilsson, Atanas Chanev, Gülşen Eryiğit, Sandra Kübler, Svetoslav Marinov, and Erwin Marsi. Maltparser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13(2):95–135, 2007b. 2.2

Ivan Titov and James Henderson. Fast and robust multilingual dependency parsing with a generative latent variable model. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*, pages 947–951, Praha, Czechia, June 2007. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/D/D07/D07-1099`. 4

Daniel Zeman. *Parsing with a Statistical Dependency Model*. PhD thesis, Univerzita Karlova v Praze, Praha, Czechia, 2004. 2.3, 3.1, 3.3

Daniel Zeman and Zdeněk Žabokrtský. Improving parsing accuracy by combining diverse dependency parsers. In *Proceedings of the Ninth International Workshop on Parsing Technologies (IWPT)*, pages 171–178, Vancouver, British Columbia, Canada, 2005. Association for Computational Linguistics. ISBN 1-932432-58-2. 2.4