Analyzing the Effect of Global Learning and Beam-search on Transition-based Dependency Parsing

Yue Zhang[1] *Joakim Nivre*[2]
(1) Singapore University of Technology and Design
(2) Uppsala University, Sweden
yue_zhang@sutd.edu.sg, joakim.nivre@lingfil.uu.se

Abstract

Beam-search and global models have been applied to transition-based dependency parsing, leading to state-of-the-art accuracies that are comparable to the best graph-based parsers. In this paper, we analyze the effects of global learning and beam-search on the overall accuracy and error distribution of a transition-based dependency parser. First, we show that global learning and beam-search must be jointly applied to give improvements over greedy, locally trained parsing. We then show that in addition to the reduction of error propagation, an important advantage of the combination of global learning and beam-search is that it accommodates more powerful parsing models without overfitting. Finally, we characterize the errors of a global, beam-search, transition-based parser, relating it to the classic contrast between "local, greedy, transition-based parsing" and "global, exhaustive, graph-based parsing".

Title and Abstract in Chinese

分析全局模型和柱搜索对基于转移依存分析器的影响

柱搜索和全局模型被应用于基于转移的依存分析，可以取得与最好的基于图的依存分析器同一水平的精度。我们分析全局学习和柱搜索对基于转移的依存分析器的精度与错误分布的影响。首先，全局学习和柱搜索需要同时使用才能达到显著优于局部学习和贪婪搜索的效果。此外，全局学习和柱搜索的联合使用不仅可以减少错误蔓延，还可以支持更为复杂的模型训练而不过拟合。最后，我们对应用了全局学习和柱搜索的基于转移的依存分析器进行错误分析，且将此分析与对MaltParser与MSTParser的错误对比相比较。

Keywords: Dependency parsing, error analysis, ZPar, MaltParser, MSTParser.

Keywords in Chinese: 依存分析, 错误分析, ZPar, MaltParser, MSTParser

Proceedings of COLING 2012: Posters, pages 1391–1400,
COLING 2012, Mumbai, December 2012.

1 Introduction

Beam-search has been applied to transition-based dependency parsing in recent studies (Zhang and Clark, 2008; Huang and Sagae, 2010; Hatori et al., 2011). In addition to reducing search errors compared to greedy search, it also enables the use of global models that accommodate richer non-local features without overfitting, leading to recent state-of-the-art accuracies of transition-based dependency parsing (Zhang and Nivre, 2011; Bohnet and Kuhn, 2012; Bohnet and Nivre, 2012) that are competitive with the best graph-based dependency parsers.

It has been known that a transition-based parser using global learning, beam-search and rich features gives significantly higher accuracies than one with local learning and greedy search. However, the effects of global learning, beam-search and rich features have not been separately studied. Apart from the natural conclusion that beam-search reduces error propagation compared to greedy search, exactly how these techniques help to improve parsing has not been discussed, and many interesting questions remain unanswered. For example, the contribution of global learning in improving the accuracies has not been separately studied. It has not been shown how global learning affects the accuracies, or whether it is important at all. For another example, it would be interesting to know whether a local, greedy, transition-based parser can be equipped with the rich features of Zhang and Nivre (2011) to improve its accuracy, and in particular whether MaltParser (Nivre et al., 2006) can achieve the same level of accuracies as ZPar (Zhang and Nivre, 2011) by using the same range of rich feature definitions.

In this paper, we answer the above questions empirically. First, we separate out global learning and beam-search, and study the effect of each technique by comparison with a local greedy baseline. Our results show that significant improvements are achieved only when the two are jointly applied. Second, we show that the accuracies of a local, greedy transition-based parser cannot be improved by adding the rich features of Zhang and Nivre (2011). Our result suggests that global learning with beam-search accommodates more complex models with richer features than a local model with greedy search and therefore enables higher accuracies.

One interesting aspect of using a global model with beam-search is that it narrows down the contrast between “local, greedy, transition-based parsing” and “global, exhaustive, graph-based parsing” as exemplified by McDonald and Nivre (2007). On the one hand, global beam-search parsing is more similar to global, exhaustive parsing than local, greedy parsing in the use of global models and non-greedy search. On the other hand, beam-search does not affect the fundamental transition-based parsing process, which allows the use of rich non-local features, and is very different from graph-based parsing.

An interesting question is how such differences in models and algorithms affect empirical errors. McDonald and Nivre (2007) make a comparative analysis of local greedy transition-based MaltParser and global near-exhaustive graph-based MSTParser (McDonald and Pereira, 2006) using the CoNLL-X Shared Task data (Buchholz and Marsi, 2006), showing that the parsers give near identical overall accuracies, but have very different error distributions according to various metrics. While MaltParser is more accurate on frequently occurring short sentences and dependencies, it performs worse on long sentences and dependencies due to search errors.

We present empirical studies of the error distribution of global, beam-search transition-based dependency parsing, using ZPar (Zhang and Nivre, 2011) as a representative system. We follow McDonald and Nivre (2007) and perform a comparative error analysis of ZPar, MSTParser and MaltParser using the CoNLL-X shared task data. Our results show that beam-search improves the precision on long sentences and dependencies compared to greedy search, while the advantage of transition-based parsing on short dependencies is preserved. Under particular measures, such as precision for arcs at different levels of the trees, ZPar shows characteristics surprisingly similar to MSTParser.

2 Analyzing the effect of global learning and beam-search

In this section we study the effects of global learning and beam-search on the accuracies of transition-based dependency parsing. Our experiments are performed using the Penn Treebank (PTB). We follow the standard approach to split PTB3 into training (sections 2–21), development (section 22) and final testing (section 23) sections. Bracketed sentences from the treebank are transformed into dependency structures using the Penn2Malt tool.[1] POS-tags are assigned using a perceptron tagger (Collins, 2002), with an accuracy of 97.3% on a standard Penn Treebank test. We assign automatic POS-tags to the training data using ten-way jacknifing. Accuracies are measured using the *unlabeled attached score* (UAS) metric, which is defined as the percentage of words (excluding punctuation) that are assigned the correct heads.

2.1 The effects of global learning and beam-search

In this subsection, we study the effects of global learning and beam-search separately. Our experiments are performed using ZPar, which uses global learning and beam-search. To make comparisons with local learning under different settings, we make configurations and modifications to ZPar where necessary. Global learning is implemented in the same way as Zhang and Nivre (2011), using the averaged perceptron algorithm (Collins, 2002) and early update (Collins and Roark, 2004). This is a global learning method in the sense that it tries to maximize accuracy over the entire sentence and not on isolated local transitions. Unless explicitly specified, the same beam size is applied for training and testing when beam-search is applied. Local learning is implemented as a multi-class classifier that predicts the next transition action given a parser configuration (i.e. a stack and an incoming queue), trained using the averaged perceptron algorithm. In local learning, each transition is considered in isolation and there is no global view of the transition sequence needed to parse an entire sentence.

Figure 1 shows the UAS of ZPar under different settings, where 'global' refers to a global model trained using the same method as Zhang and Nivre (2011), 'local' refers to a local classifier trained using the averaged perceptron, 'base features' refers to the set of base feature templates in Zhang and Nivre (2011), and 'all features' refers to the set of base and all extended feature templates in Zhang and Nivre (2011).

When the size of the beam is 1, the decoding algorithm is greedy local search. Using base features, a locally trained model gives a UAS of 89.15%, higher than that of a globally trained model (89.04%). Here a global model does not give better accuracies compared to a local model under greedy search.

As the size of the beam increses, the UAS of the global model increases, but the UAS of the local model decreases. Global learning gives significantly better accuracies than local learning under beam-search. There are two ways to explain the reason that beam-search hurts the UAS of a locally trained model. First, the perceptron can be viewed as a large-margin training algorithm that finds a separation margin between the scores of positive examples (gold-standard structures) and negative examples (non-gold structures from the decoder). The online learning

[1] http://w3.msi.vxu.se/ñivre/research/Penn2Malt.html.

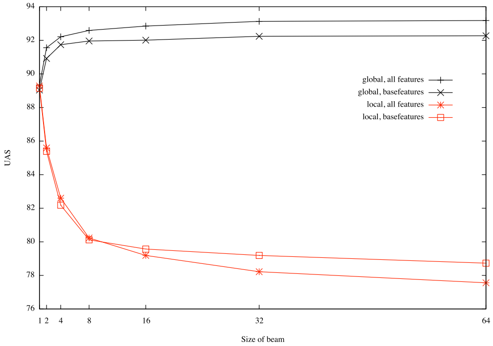

Figure 1: The effect of global learning and beam-search.

training beam	testing beam	UAS
1	1	89.04
1	64	79.34
64	1	87.07
64	64	92.27

Table 1: The effect of different settings between training and testing.

process runs the decoding algorithm to generate a space of negative examples, which is used together with its corresponding positive example space for parameter udpates. If the negative example space during training is different from that during testing, the trained model will not separate the test examples as effectively as when the negative example spaces for training and testing are similar, since there are more unseen negative examples in the model.

To further illustrate this, we conduct an additional set of development experiments by training two global models with different beam sizes. Each of the models is tested using its own training beam size and the training beam size of the other model. The results are shown in Table 1. As can be seen from the table, a global model trained with a size-1 beam gives a higher UAS when tested with a size-1 beam than with a size-64 beam. Similarly, a global model trained with a size-64 beam gives a higher UAS when tested using a size-64 beam than using a size-1 beam. Our observations are consistent with those of Daumé III and Marcu (2005), which show that the accuracies of another online large-margin model are lower when the training and testing beam sizes are different than when they are the same. These results show the negative effect of a mismatch between training and testing negative example spaces, which also happens when a locally trained model is tested using beam-search.

To take a second perspective, a local model is trained to disambiguate different transition actions under the same parser configuration, but not different transitions under different parser configurations. This means that the scores of two sequences of transition actions may not be comparable with each other when they consist of very different parser configuration sequences. This is reminiscent of the label bias problem (Lafferty et al., 2001), and partly explains the performance degradation of the local model when tested with beam-search.

	ZPar	Malt
Baseline	92.18	89.37
+distance	+0.07	−0.14
+valency	+0.24	0.00
+unigrams	+0.40	−0.29
+third-order	+0.18	0.00
+label set	+0.07	+0.06
Extended	93.14	89.00

Table 2: The effect of adding rich non-local features to ZPar and MaltParser. Row 'Baseline' shows the scores of ZPar and MaltParser before extended features are applied. Rows '+distance', '+valency', '+unigrams', '+third-order' and '+label set' show the effect of each group of extended features of Zhang and Nivre (2011), respectively. Row 'Extended' shows the scores with all extended features.

To summarize the above discussion, a global model does not improve over a local model for greedy parsing, and beam-search does not improve the performance of a parser trained locally using the perceptron algorithm. However, the combination of global learning and beam-search can significantly improve the performance compared to a local, greedy transition-based parser.

2.2 Benefits from global learning and beam-search

An additional benefit of global learning and beam-search is the accommodation of rich non-local features. Again in Figure 1, the use of rich non-local features improves the UAS of the global models with all beam sizes, while the improvement brought by rich non-local features also increases with increased size of the beam. With greedy local search, the accuracy improves from 89.04% with base features to 89.35% with all features; with the size of the beam being 64, the accuracy improves form 92.27% with base features to 93.18% with all features. The absolute improvement increased from 0.3% to 0.89%.

The above fact shows that rich non-local features are more effective on a global model with a large beam-size. This is a consequence of the interaction between learning and search: a large beam not only reduces search errors, but also enables a more complex model to be trained without overfitting. In contrast to a globally trained model, a local model cannot benefit as much from the power of rich features. With greedy local search, the UAS of a local model improves from 89.15% with base features to 89.28% with all features. Beam-search does not bring additional improvements.

For further evidence, we add rich non-local features in the same increments as Zhang and Nivre (2011) to both ZPar and MaltParser, and evaluate UAS on the same development data set. Original settings are applied to both parsers, with ZPar using global learning and beam-search, and MaltParser using local learning and greedy search. Table 2 shows that while ZPar's accuracy consistently improves with the addition of each new set of features, there is very little impact on MaltParser's accuracy and in some cases the effect is in fact negative, indicating that the locally trained greedy parser cannot benefit from the rich non-local features.

Yet another evidence for the support of more complex models by global learning and beam-search is the work of Bohnet and Nivre (2012), where non-projective parsing using online reordering (Nivre, 2009) and rich features led to significant improvements over greedy search (Nivre, 2009), achieving state-of-the-art on a range of typologically diverse languages.

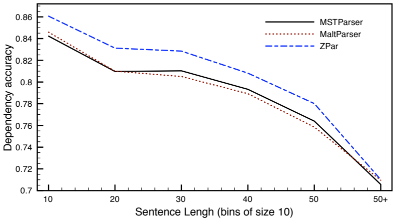

Figure 2: Accuracy relative to sentence length.

3 Characterizing the errors

3.1 The parsers and evaluation data

In this section we study the effect of global learning and beam-search on the error distributions of transition-based dependency parsing. We characterize the errors of ZPar and add it to the error comparison between MaltParser and MSTParser (McDonald and Nivre, 2007).

Following McDonald and Nivre (2007) we evaluate the parsers on the CoNLL-X Shared Task data (Buchholz and Marsi, 2006), which include training and test sentences for 13 different languages. For each parser, we conjoin the outputs for all 13 languages in the same way as McDonald and Nivre (2007), and calculate error distributions over the aggregated output. Accuracies are measured using the *labeled attached score* (LAS) evaluation metric, which is defined as the percentage of words (excluding punctuation) that are assigned both the correct head word and the correct arc label.

To handle non-projectivity, pseudo-projective parsing (Nivre and Nilsson, 2005) is applied to ZPar and MaltParser, transforming non-projective trees into pseudo-projective trees in the training data, and post-processing pseudo-projective outputs by the parser to transform them into non-projective trees. MSTParser produces non-projective trees from projective trees by score-based rearrangements of arcs.

3.2 Error distributions

We take a range of different perspectives to characterize the errors of ZPar, comparing them with those of MaltParser and MSTParser by measuring the accuracies against various types of metrics, including the size of the sentences and dependency arcs, the distance to the root of the dependency tree, and the number of siblings. The parsers show different empirical performances over these measures, demonstrating the comparative advantages and disadvantages of their design discussed in Section 3.1.

Figure 2 shows the accuracy of the parsers relative to sentence length (the number of words in a sentence, in bins of size 10). All three parsers perform comparatively better on short sentences. The performance of MaltParser and MSTParser is very similar, with MaltParser performing better on very short sentences (≤ 20) due to richer feature representations, and worse on longer sentences (20 to 50) due to the propagation of search errors. Because short sentences are much more frequent in the test data, MaltParser and MSTParser give almost identical overall accuracies.

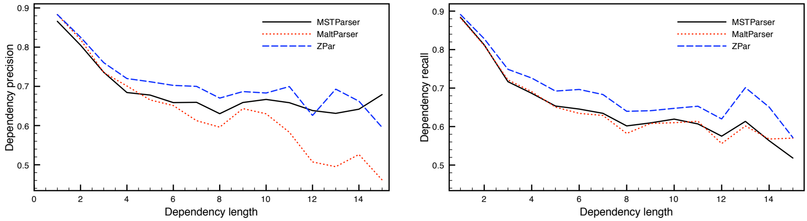

Figure 3: Dependency arc precision/recall relative to predicted/gold dependency length.

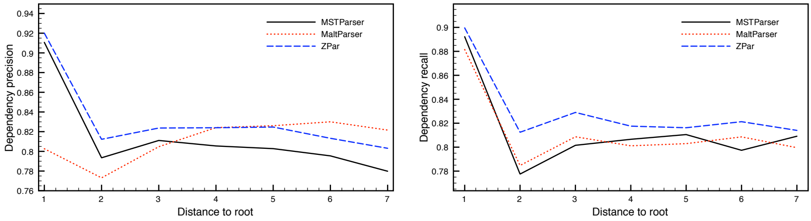

Figure 4: Dependency arc precision/recall relative to predicted/gold distance to root.

ZPar performs better than MaltParser and MSTParser, particularly on short sentences (≤ 30), due to the richest feature representation. For longer sentences (20 to 50), the performance of ZPar drops as quickly as that of MaltParser. One possible reason is that the effect of a fixed-size beam on the reduction of error propagation becomes less obvious when the number of possible parse trees grows exponentially with sentence size. The performance of MSTParser decreases less quickly as the size of the sentence increases, demonstrating the advantage of exact inference. Sentences with 50+ words are relatively rare in the test set.

The three parsers show larger variance in performance when evaluated against specific properties of the dependency tree. Figure 3 shows the precision and recall for each parser relative to the arc lengths in the predicted and gold-standard dependency trees. Here the length of an arc is defined as the absolute difference between the indices of the head and modifier. Precision represents the percentage of predicted arcs with a particular length that are correct, and recall represents the percentage of gold arcs of a particular length that are correctly predicted.

MaltParser gives higher precision than MSTParser for short dependency arcs (≤ 4), but its precision drops rapidly for arcs with increased lengths. These arcs take more shift-reduce actions to build, and are hence more prone to error propagation. The precision of ZPar drops much slower compared to MaltParser, demonstrating the effect of beam-search for the reduction of error propagation. Another important factor is the use of rich non-local features by ZPar, which is a likely reason for its precision to drop slower even than that of MSTParser when the arc size increases from 1 to 8. Interestingly, the precision of ZPar is almost indistinguishable from that of MaltParser for size 1 arcs (arcs between neighbouring words), showing that the wider range of features in ZPar is the most helpful in arcs that take more than one, but not too many shift-reduce actions to build. The recall curves of the three parsers are similar, with ZPar having higher recall than MSTParser and MaltParser, particularly when the dependency size is greater than 2. This shows that particular gold-standard dependencies are hard for all parsers to build, but ZPar is better in recovering hard gold dependencies probably due to its rich features.

To take another perspective, we compare the performance of the three parsers at different levels of a dependency tree by measuring accuracies for arcs relative to their distance to the root. Here the distance of an arc to the root is defined as the number of arcs in the path from the root to the modifier in the arc. Figure 4 shows the precision and recall of each system for arcs of varying distances to the root.

Here the precision of MaltParser and MSTParser is very different, with MaltParser being more precise for arcs nearer to the leaves, but less precise for those nearer to the root. One possible reason is that arcs near the bottom of the tree require comparatively fewer shift-reduce actions to build, and are therefore less prone to the propagation of search errors. Another important reason, as pointed out by McDonald and Nivre (2007), is the default single-root mechanism by MaltParser: all words that have not been attached as a modifier when the shift-reduce process finishes are attached as modifiers to the pseudo-root. Although the vast majority of sentences have only one root-modifier, there is no global control for the number of root-modifiers in the greedy shift-reduce process, and each action is made locally and independently. As a result, MaltParser tends to over-predict root modifiers, leading to the comparatively low precision.

Surprisingly, the precision curve of ZPar is much more similar to that of MSTParser than that of MaltParser, although ZPar is based on the same shift-reduce parsing process, and even has a similar default single-root mechanism as MaltParser. This result is perhaps the most powerful demonstration of the effect of global learning and beam-search compared to local learning and greedy search. The model which scores whole sequences of shift-reduce actions, plus the reduction of search error propagation, lead to significantly reduced over-prediction of root-modifiers. In addition, rich features used by ZPar, such as the valency (number of modifiers for a head) and set of modifier labels for a head, can also be useful in reducing over-prediction of modifiers. Because of these, ZPar effectively pushes the predictions of difficult arcs down the tree, which is exactly the behavior of MSTParser. Interestingly, the recall curve of ZPar is more similar to that of MaltParser than that of MSTParser, showing that arcs at particular levels are harder to recover using the shift-reduce process than a global tree search.

4 Conclusion

We studied empirically the effect of global learning and beam-search on the overall accuracies and error distributions of transition-based dependency parsing. We first analyzed the ways in which global learning and beam-search improved parsing accuracies over local learning and greedy search, showing that they allow more complex parsing models without overfitting, including the use of rich non-local features and online reordering for non-projective parsing, which result in state-of-the-art accuracies (Zhang and Nivre, 2011; Zhang and Clark, 2011; Bohnet and Nivre, 2012). We also showed that the effects result from the interaction between global learning and beam-search, and that applying either of the techniques by itself does not lead to improvements over local learning and greedy search. We then performed a detailed error analysis of a global, beam-search transition-based dependency parser, relating it to the classic comparison of local greedy transition-based and global near-exhaustive graph-based parsing (McDonald and Nivre, 2007). Our results might serve to inspire further parser developments by providing more insights into these techniques.

References

Bohnet, B. and Kuhn, J. (2012). The best of bothworlds – a graph-based completion model for transition-based parsers. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 77–87, Avignon, France. Association for Computational Linguistics.

Bohnet, B. and Nivre, J. (2012). A transition-based system for joint part-of-speech tagging and labeled non-projective dependency parsing. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1455–1465, Jeju Island, Korea. Association for Computational Linguistics.

Buchholz, S. and Marsi, E. (2006). Conll-X shared task on multilingual dependency parsing. In *Proceedings of CoNLL*, pages 149–164, New York City.

Collins, M. (2002). Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms. In *Proceedings of EMNLP*, pages 1–8, Philadelphia, USA.

Collins, M. and Roark, B. (2004). Incremental parsing with the perceptron algorithm. In *Proceedings of ACL*, pages 111–118, Barcelona, Spain.

Daumé III, H. and Marcu, D. (2005). Learning as search optimization: approximate large margin methods for structured prediction. In *ICML*, pages 169–176.

Hatori, J., Matsuzaki, T., Miyao, Y., and Tsujii, J. (2011). Incremental joint POS tagging and dependency parsing in chinese. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 1216–1224, Chiang Mai, Thailand. Asian Federation of Natural Language Processing.

Huang, L. and Sagae, K. (2010). Dynamic programming for linear-time incremental parsing. In *Proceedings of ACL*, pages 1077–1086, Uppsala, Sweden.

Lafferty, J., McCallum, A., and Pereira, F. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of ICML*, pages 282–289, Massachusetts, USA.

McDonald, R. and Nivre, J. (2007). Characterizing the errors of data-driven dependency parsing models. In *Proceedings of EMNLP/CoNLL*, pages 122–131, Prague, Czech Republic.

McDonald, R. and Pereira, F. (2006). Online learning of approximate dependency parsing algorithms. In *Proceedings of EACL*, pages 81–88, Trento, Italy.

Nivre, J. (2009). 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. Association for Computational Linguistics.

Nivre, J., Hall, J., Nilsson, J., Eryiğit, G., and Marinov, S. (2006). Labeled pseudo-projective dependency parsing with support vector machines. In *Proceedings of CoNLL*, pages 221–225, New York, USA.

Nivre, J. and Nilsson, J. (2005). Pseudo-projective dependency parsing. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 99–106, Ann Arbor, Michigan. Association for Computational Linguistics.

Zhang, Y. and Clark, S. (2008). A tale of two parsers: investigating and combining graph-based and transition-based dependency parsing using beam-search. In *Proceedings of EMNLP*, Hawaii, USA.

Zhang, Y. and Clark, S. (2011). Syntactic processing using the generalized perceptron and beam search. *Computational Linguistics*, 37(1):105–151.

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