Transition-based Dependency Parsing with Rich Non-local Features

Yue Zhang
University of Cambridge
Computer Laboratory
yue.zhang@cl.cam.ac.uk

Joakim Nivre
Uppsala University
Department of Linguistics and Philology
joakim.nivre@lingfil.uu.se

Abstract

Transition-based dependency parsers generally use heuristic decoding algorithms but can accommodate arbitrarily rich feature representations. In this paper, we show that we can improve the accuracy of such parsers by considering even richer feature sets than those employed in previous systems. In the standard Penn Treebank setup, our novel features improve attachment score form 91.4% to 92.9%, giving the best results so far for transition-based parsing and rivaling the best results overall. For the Chinese Treebank, they give a signficant improvement of the state of the art. An open source release of our parser is freely available.

1 Introduction

Transition-based dependency parsing (Yamada and Matsumoto, 2003; Nivre et al., 2006b; Zhang and Clark, 2008; Huang and Sagae, 2010) utilize a deterministic shift-reduce process for making structural predictions. Compared to graph-based dependency parsing, it typically offers linear time complexity and the comparative freedom to define non-local features, as exemplified by the comparison between MaltParser and MSTParser (Nivre et al., 2006b; McDonald et al., 2005; McDonald and Nivre, 2007).

Recent research has addressed two potential disadvantages of systems like MaltParser. In the aspect of decoding, beam-search (Johansson and Nugues, 2007; Zhang and Clark, 2008; Huang et al., 2009) and partial dynamic-programming (Huang and Sagae, 2010) have been applied to improve upon greedy one-best search, and positive results were reported. In the aspect of training, global structural learning has been used to replace local learning on each decision (Zhang and Clark, 2008; Huang et al., 2009), although the effect of global learning has not been separated out and studied alone.

In this short paper, we study a third aspect in a statistical system: feature definition. Representing the type of information a statistical system uses to make predictions, feature templates can be one of the most important factors determining parsing accuracy. Various recent attempts have been made to include non-local features into graph-based dependency parsing (Smith and Eisner, 2008; Martins et al., 2009; Koo and Collins, 2010). Transition-based parsing, by contrast, can easily accommodate arbitrarily complex representations involving non-local features. Complex non-local features, such as bracket matching and rhythmic patterns, are used in transition-based constituency parsing (Zhang and Clark, 2009; Wang et al., 2006), and most transition-based dependency parsers incorporate some non-local features, but current practice is nevertheless to use a rather restricted set of features, as exemplified by the default feature models in MaltParser (Nivre et al., 2006a). We explore considerably richer feature representations and show that they improve parsing accuracy significantly.

In standard experiments using the Penn Treebank, our parser gets an unlabeled attachment score of 92.9%, which is the best result achieved with a transition-based parser and comparable to the state of the art. For the Chinese Treebank, our parser gets a score of 86.0%, the best reported result so far.

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics:shortpapers, pages 188–193,
Portland, Oregon, June 19-24, 2011. ©2011 Association for Computational Linguistics

2 The Transition-based Parsing Algorithm

In a typical transition-based parsing process, the input words are put into a queue and partially built structures are organized by a stack. A set of shift-reduce actions are defined, which consume words from the queue and build the output parse. Recent research have focused on action sets that build projective dependency trees in an *arc-eager* (Nivre et al., 2006b; Zhang and Clark, 2008) or *arc-standard* (Yamada and Matsumoto, 2003; Huang and Sagae, 2010) process. We adopt the arc-eager system[1], for which the actions are:

- `Shift`, which removes the front of the queue and pushes it onto the top of the stack;
- `Reduce`, which pops the top item off the stack;
- `LeftArc`, which pops the top item off the stack, and adds it as a modifier to the front of the queue;
- `RightArc`, which removes the front of the queue, pushes it onto the stack and adds it as a modifier to the top of the stack.

Further, we follow Zhang and Clark (2008) and Huang et al. (2009) and use the generalized perceptron (Collins, 2002) for global learning and beam-search for decoding. Unlike both earlier global-learning parsers, which only perform unlabeled parsing, we perform labeled parsing by augmenting the `LeftArc` and `RightArc` actions with the set of dependency labels. Hence our work is in line with Titov and Henderson (2007) in using labeled transitions with global learning. Moreover, we will see that label information can actually improve link accuracy.

3 Feature Templates

At each step during a parsing process, the parser configuration can be represented by a tuple $\langle S, N, A\rangle$, where S is the stack, N is the queue of incoming words, and A is the set of dependency arcs that have been built. Denoting the top of stack with S_0, the front items from the queue with N_0, N_1, and N_2, the head of S_0 (if any) with S_{0h}, the leftmost and rightmost modifiers of S_0 (if any) with S_{0l} and S_{0r}, respectively, and the leftmost modifier of N_0 (if any) with N_{0l}, the baseline features are shown in Table 1. These features are mostly taken from Zhang and Clark (2008) and Huang and Sagae (2010), and our parser reproduces the same accuracies as reported by both papers. In this table, w and p represents the word and POS-tag, respectively. For example, S_0pN_0wp represents the feature template that takes the word and POS-tag of N_0, and combines it with the word of S_0.

from single words
S_0wp; S_0w; S_0p; N_0wp; N_0w; N_0p; N_1wp; N_1w; N_1p; N_2wp; N_2w; N_2p;
from word pairs
S_0wpN_0wp; S_0wpN_0w; S_0wN_0wp; S_0wpN_0p; S_0pN_0wp; S_0wN_0w; S_0pN_0p N_0pN_1p
from three words
$N_0pN_1pN_2p$; $S_0pN_0pN_1p$; $S_{0h}pS_0pN_0p$; $S_0pS_{0l}pN_0p$; $S_0pS_{0r}pN_0p$; $S_0pN_0pN_{0l}p$

Table 1: Baseline feature templates. w – word; p – POS-tag.

distance
S_0wd; S_0pd; N_0wd; N_0pd; S_0wN_0wd; S_0pN_0pd;
valency
S_0wv_r; S_0pv_r; S_0wv_l; S_0pv_l; N_0wv_l; N_0pv_l;
unigrams
$S_{0h}w$; $S_{0h}p$; S_0l; $S_{0l}w$; $S_{0l}p$; $S_{0l}l$; $S_{0r}w$; $S_{0r}p$; $S_{0r}l$;$N_{0l}w$; $N_{0l}p$; $N_{0l}l$;
third-order
$S_{0h2}w$; $S_{0h2}p$; $S_{0h}l$; $S_{0l2}w$; $S_{0l2}p$; $S_{0l2}l$; $S_{0r2}w$; $S_{0r2}p$; $S_{0r2}l$; $N_{0l2}w$; $N_{0l2}p$; $N_{0l2}l$; $S_0pS_{0l}pS_{0l2}p$; $S_0pS_{0r}pS_{0r2}p$; $S_0pS_{0h}pS_{0h2}p$; $N_0pN_{0l}pN_{0l2}p$;
label set
S_0ws_r; S_0ps_r; S_0ws_l; S_0ps_l; N_0ws_l; N_0ps_l;

Table 2: New feature templates. w – word; p – POS-tag; v_l, v_r – valency; l – dependency label, s_l, s_r – labelset.

[1]It is very likely that the type of features explored in this paper would be beneficial also for the arc-standard system, although the exact same feature templates would not be applicable because of differences in the parsing order.

In this short paper, we extend the baseline feature templates with the following:

Distance between S_0 and N_0
Direction and distance between a pair of head and modifier have been used in the standard feature templates for maximum spanning tree parsing (McDonald et al., 2005). Distance information has also been used in the easy-first parser of (Goldberg and Elhadad, 2010). For a transition-based parser, direction information is indirectly included in the `LeftArc` and `RightArc` actions. We add the distance between S_0 and N_0 to the feature set by combining it with the word and POS-tag of S_0 and N_0, as shown in Table 2.

It is worth noticing that the use of distance information in our transition-based model is different from that in a typical graph-based parser such as MSTParser. The distance between S_0 and N_0 will correspond to the distance between a pair of head and modifier when an `LeftArc` action is taken, for example, but not when a `Shift` action is taken.

Valency of S_0 and N_0
The number of modifiers to a given head is used by the graph-based submodel of Zhang and Clark (2008) and the models of Martins et al. (2009) and Sagae and Tsujii (2007). We include similar information in our model. In particular, we calculate the number of left and right modifiers separately, calling them *left valency* and *right valency*, respectively. Left and right valencies are represented by v_l and v_r in Table 2, respectively. They are combined with the word and POS-tag of S_0 and N_0 to form new feature templates.

Again, the use of valency information in our transition-based parser is different from the aforementioned graph-based models. In our case, valency information is put into the context of the shift-reduce process, and used together with each action to give a score to the local decision.

Unigram information for S_{0h}, S_{0l}, S_{0r} and N_{0l}
The head, left/rightmost modifiers of S_0 and the leftmost modifier of N_0 have been used by most arc-eager transition-based parsers we are aware of through the combination of their POS-tag with information from S_0 and N_0. Such use is exemplified by the feature templates "from three words" in Table 1. We further use their word and POS-tag information as "unigram" features in Table 2. Moreover, we include the dependency label information in the unigram features, represented by l in the table. Unigram label information has been used in MaltParser (Nivre et al., 2006a; Nivre, 2006).

Third-order features of S_0 and N_0
Higher-order context features have been used by graph-based dependency parsers to improve accuracies (Carreras, 2007; Koo and Collins, 2010). We include information of third order dependency arcs in our new feature templates, when available. In Table 2, S_{0h2}, S_{0l2}, S_{0r2} and N_{0l2} refer to the head of S_{0h}, the second leftmost modifier and the second rightmost modifier of S_0, and the second leftmost modifier of N_0, respectively. The new templates include unigram word, POS-tag and dependency labels of S_{0h2}, S_{0l2}, S_{0r2} and N_{0l2}, as well as POS-tag combinations with S_0 and N_0.

Set of dependency labels with S_0 and N_0
As a more global feature, we include the set of unique dependency labels from the modifiers of S_0 and N_0. This information is combined with the word and POS-tag of S_0 and N_0 to make feature templates. In Table 2, s_l and s_r stands for the set of labels on the left and right of the head, respectively.

4 Experiments

Our experiments were performed using the Penn Treebank (PTB) and Chinese Treebank (CTB) data. We follow the standard approach to split PTB3, using sections 2 – 21 for training, section 22 for development and 23 for final testing. Bracketed sentences from PTB were transformed into dependency formats using the Penn2Malt tool.[2] Following Huang and Sagae (2010), we assign POS-tags to the training data using ten-way jackknifing. We used our implementation of the Collins (2002) tagger (with 97.3% accuracy on a standard Penn Treebank test) to perform POS-tagging. For all experiments, we set the beam size to 64 for the parser, and report unlabeled and labeled attachment scores (UAS, LAS) and unlabeled exact match (UEM) for evaluation.

[2] http://w3.msi.vxu.se/ nivre/research/Penn2Malt.html

feature	UAS	UEM
baseline	92.18%	45.76%
+distance	92.25%	46.24%
+valency	92.49%	47.65%
+unigrams	92.89%	48.47%
+third-order	93.07%	49.59%
+label set	93.14%	50.12%

Table 3: The effect of new features on the development set for English. UAS = unlabeled attachment score; UEM = unlabeled exact match.

	UAS	UEM	LAS
Z&C08 transition	91.4%	41.8%	—
H&S10	91.4%	—	—
this paper baseline	91.4%	42.5%	90.1%
this paper extended	92.9%	48.0%	91.8%
MSTParser	91.5%	42.5%	—
K08 standard	92.0%	—	—
K&C10 model 1	93.0%	—	—
K&C10 model 2	92.9%	—	—

Table 4: Final test accuracies for English. UAS = unlabeled attachment score; UEM = unlabeled exact match; LAS = labeled attachment score.

4.1 Development Experiments

Table 3 shows the effect of new features on the development test data for English. We start with the baseline features in Table 1, and incrementally add the distance, valency, unigram, third-order and label set feature templates in Table 2. Each group of new feature templates improved the accuracies over the previous system, and the final accuracy with all new features was 93.14% in unlabeled attachment score.

4.2 Final Test Results

Table 4 shows the final test results of our parser for English. We include in the table results from the pure transition-based parser of Zhang and Clark (2008) (row 'Z&C08 transition'), the dynamic-programming arc-standard parser of Huang and Sagae (2010) (row 'H&S10'), and graph-based models including MSTParser (McDonald and Pereira, 2006), the baseline feature parser of Koo et al. (2008) (row 'K08 baeline'), and the two models of Koo and Collins (2010). Our extended parser significantly outperformed the baseline parser, achieving the highest attachment score reported for a transition-based parser, comparable to those of the best graph-based parsers.

	UAS	UEM	LAS
Z&C08 transition	84.3%	32.8%	—
H&S10	85.2%	33.7%	—
this paper extended	86.0%	36.9%	84.4%

Table 5: Final test accuracies for Chinese. UAS = unlabeled attachment score; UEM = unlabeled exact match; LAS = labeled attachment score.

Our experiments were performed on a Linux platform with a 2GHz CPU. The speed of our baseline parser was 50 sentences per second. With all new features added, the speed dropped to 29 sentences per second.

As an alternative to Penn2Malt, bracketed sentences can also be transformed into Stanford dependencies (De Marneffe et al., 2006). Our parser gave 93.5% UAS, 91.9% LAS and 52.1% UEM when trained and evaluated on Stanford *basic* dependencies, which are projective dependency trees. Cer et al. (2010) report results on Stanford *collapsed* dependencies, which allow a word to have multiple heads and therefore cannot be produced by a regular dependency parser. Their results are relevant although not directly comparable with ours.

4.3 Chinese Test Results

Table 5 shows the results of our final parser, the pure transition-based parser of Zhang and Clark (2008), and the parser of Huang and Sagae (2010) on Chinese. We take the standard split of CTB and use gold segmentation and POS-tags for the input. Our scores for this test set are the best reported so far and significantly better than the previous systems.

5 Conclusion

We have shown that enriching the feature representation significantly improves the accuracy of our transition-based dependency parser. The effect of the new features appears to outweigh the effect of combining transition-based and graph-based models, reported by Zhang and Clark (2008), as well as the effect of using dynamic programming, as in Huang and Sagae (2010). This shows that feature definition is a crucial aspect of transition-based pars-

ing. In fact, some of the new feature templates in this paper, such as distance and valency, are among those which are in the graph-based submodel of Zhang and Clark (2008), but not the transition-based submodel. Therefore our new features to some extent achieved the same effect as their model combination. The new features are also hard to use in dynamic programming because they add considerable complexity to the parse items.

Enriched feature representations have been studied as an important factor for improving the accuracies of graph-based dependency parsing also. Recent research including the use of loopy belief network (Smith and Eisner, 2008), integer linear programming (Martins et al., 2009) and an improved dynamic programming algorithm (Koo and Collins, 2010) can be seen as methods to incorporate non-local features into a graph-based model.

An open source release of our parser, together with trained models for English and Chinese, are freely available.[3]

Acknowledgements

We thank the anonymous reviewers for their useful comments. Yue Zhang is supported by the European Union Seventh Framework Programme (FP7-ICT-2009-4) under grant agreement no. 247762.

References

Xavier Carreras. 2007. Experiments with a higher-order projective dependency parser. In *Proceedings of the CoNLL Shared Task Session of EMNLP/CoNLL*, pages 957–961, Prague, Czech Republic.

Daniel Cer, Marie-Catherine de Marneffe, Dan Jurafsky, and Chris Manning. 2010. Parsing to stanford dependencies: Trade-offs between speed and accuracy. In *Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC'10)*.

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

Marie-catherine De Marneffe, Bill Maccartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of LREC*.

[3]http://www.sourceforge.net/projects/zpar. version 0.5.

Yoav Goldberg and Michael Elhadad. 2010. An efficient algorithm for easy-first non-directional dependency parsing. In *Porceedings of HLT/NAACL*, pages 742–750, Los Angeles, California, June.

Liang Huang and Kenji Sagae. 2010. Dynamic programming for linear-time incremental parsing. In *Proceedings of ACL*, pages 1077–1086, Uppsala, Sweden, July.

Liang Huang, Wenbin Jiang, and Qun Liu. 2009. Bilingually-constrained (monolingual) shift-reduce parsing. In *Proceedings of EMNLP*, pages 1222–1231, Singapore.

Richard Johansson and Pierre Nugues. 2007. Incremental dependency parsing using online learning. In *Proceedings of CoNLL/EMNLP*, pages 1134–1138, Prague, Czech Republic.

Terry Koo and Michael Collins. 2010. Efficient third-order dependency parsers. In *Proceedings of ACL*, pages 1–11, Uppsala, Sweden, July.

Terry Koo, Xavier Carreras, and Michael Collins. 2008. Simple semi-supervised dependency parsing. In *Proceedings of ACL/HLT*, pages 595–603, Columbus, Ohio, June.

Andre Martins, Noah Smith, and Eric Xing. 2009. Concise integer linear programming formulations for dependency parsing. In *Proceedings of ACL/IJCNLP*, pages 342–350, Suntec, Singapore, August.

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

Ryan McDonald and Fernando Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *Proceedings of EACL*, pages 81–88, Trento, Italy, April.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005. Online large-margin training of dependency parsers. In *Proceedings of ACL*, pages 91–98, Ann Arbor, Michigan, June.

Joakim Nivre, Johan Hall, and Jens Nilsson. 2006a. Maltparser: A data-driven parser-generator for dependency parsing. pages 2216–2219.

Joakim Nivre, Johan Hall, Jens Nilsson, Gülşen Eryiğit, and Svetoslav Marinov. 2006b. Labeled pseudo-projective dependency parsing with support vector machines. In *Proceedings of CoNLL*, pages 221–225, New York, USA.

Joakim Nivre. 2006. *Inductive Dependency Parsing*. Springer.

Kenji Sagae and Jun'ichi Tsujii. 2007. Dependency parsing and domain adaptation with LR models and parser ensembles. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*, pages 1044–1050,

Prague, Czech Republic, June. Association for Computational Linguistics.

David Smith and Jason Eisner. 2008. Dependency parsing by belief propagation. In *Proceedings of EMNLP*, pages 145–156, Honolulu, Hawaii, October.

Ivan Titov and James Henderson. 2007. A latent variable model for generative dependency parsing. In *Proceedings of IWPT*, pages 144–155, Prague, Czech Republic, June.

Xinhao Wang, Xiaojun Lin, Dianhai Yu, Hao Tian, and Xihong Wu. 2006. Chinese word segmentation with maximum entropy and n-gram language model. In *Proceedings of SIGHAN Workshop*, pages 138–141, Sydney, Australia, July.

H Yamada and Y Matsumoto. 2003. Statistical dependency analysis using support vector machines. In *Proceedings of IWPT*, Nancy, France.

Yue Zhang and Stephen Clark. 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.

Yue Zhang and Stephen Clark. 2009. Transition-based parsing of the Chinese Treebank using a global discriminative model. In *Proceedings of IWPT*, Paris, France, October.