Language Independent Dependency to Constituent Tree Conversion

Young-Suk Lee
IBM T. J. Watson Research Center
Yorktown Heights, NY 10598
USA
ysuklee@us.ibm.com

Zhiguo Wang
IBM T. J. Watson Research Center
Yorktown Heights, NY 10598
USA
zhigwang@us.ibm.com

Abstract

We present a dependency to constituent tree conversion technique that aims to improve constituent parsing accuracies by leveraging dependency treebanks available in a wide variety in many languages. The technique works in two steps. First, a partial constituent tree is derived from a dependency tree with a *very simple* deterministic algorithm that is both language and dependency type independent. Second, a complete high accuracy constituent tree is derived with a constraint-based parser, which uses the partial constituent tree as external constraints. Evaluated on Section 22 of the WSJ Treebank, the technique achieves the state-of-the-art conversion F-score 95.6. When applied to English Universal Dependency treebank and German CoNLL2006 treebank, the converted treebanks added to the human-annotated constituent parser training corpus improve parsing F-scores significantly for both languages.

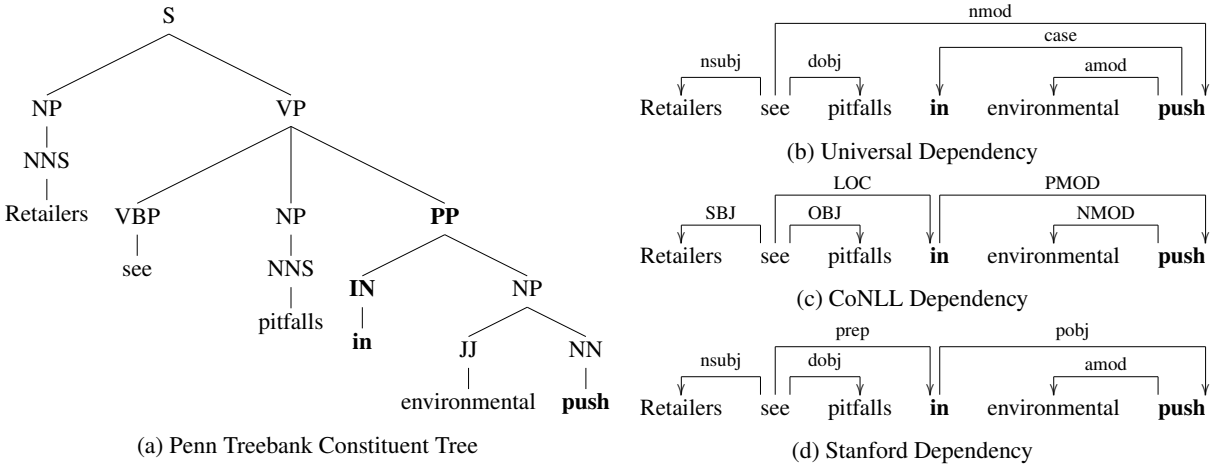
1 Introduction

State-of-the-art parsers require human annotation of a training corpus in a specific representation, e.g. constituent structure in Penn Treebank (Charniak and Johnson, 2005; Petrov and Klein, 2007) or dependency relations in a dependency treebank (Yamada and Matsumoto, 2003; McDonald et al., 2005) . Creation of human-annotated treebanks, however, is knowledge and labor intensive and it is desired that one can improve parsing performance by leveraging treebanks annotated in representations of a wide variety.

While there have been quite a few papers on automatic conversion from dependency to constituent trees and vice versa (Wang et al., 1994; Collins et al., 1999; Forst, 2003; de Marneffe et al., 2006; Johansson and Nugues, 2007; Xia et al., 2008; Hall and Nivre, 2008; Rambow, 2010; Wang and Zong, 2010; Zhang et al., 2013; Simkó et al., 2014; Kong et al., 2015) , very few papers address the issue of whether or not the converted treebank actually improves the performance of the target parser when added to the human-annotated gold treebanks for parser training. In addition, much of the work on dependency to constituency conversion relies on dependency trees automatically derived from the Penn Treebank (Marcus et al., 1993) via head rules and assumes that the head-modifier definitions are consistent between the constituent and dependency trees (Xia et al., 2008). However, such techniques cannot easily generalize to dependencies that diverge from the Penn Treebank in head-modifier definitions and dependency labels, e.g. Universal Dependency (Nivre et al., 2015) in Figure 1(b), and the dependencies of a wide variety available in CoNLL shared tasks.

In this paper, we propose a *very simple* dependency to constituent tree conversion technique which is applicable to any languages and any dependencies, e.g. Universal Dependency (UD), CoNLL dependencies (CoNLL), Stanford dependencies (Stanford), while achieving the state-of-the-art conversion accuracy. The technique works in two steps. We first derive a partial constituent tree from a dependency tree according to a simple deterministic algorithm without any external knowledge sources such as head rules. The partial constituent tree retains the gold part-of-speech tags (POStags) and partial contituent brackets inferred from the dependency tree (in Section 2). We then recover the complete constituent

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 421–428, Osaka, Japan, December 11-17 2016.



(a) Penn Treebank Constituent Tree

(b) Universal Dependency

(c) CoNLL Dependency

(d) Stanford Dependency

Figure 1: Penn Treebank Constituent Tree (a), Universal Dependency (b), CoNLL Dependency (c) and Stanford Dependency (d) representations for *Retailers see pitfalls in environmental push*. The preposition *in* is the head of *push* in the Penn Treebank, CoNLL and Stanford Dependency, whereas it is the modifier of *push* in the Universal Dependency. None of the dependency labels overlap with the Penn Treebank phrase labels. A dependency arrow goes from a head to its modifier.

structure and labels by a constraint-based parsing which uses the gold POStags and partial brackets as parsing constraints (in Section 3).

Evaluated on WSJ-22 for conversion accuracy, the proposed technique achieves the labeled F-score of 95.62 for conversion from the Stanford (de Marneffe et al., 2006) basic dependency (in Section 4). When applied to the English Universal Dependency (UD) treebank and German CoNLL2006 treebank, the converted treebanks added to the human-annotated constituent parser training corpus improve the F-scores of BerkeleyParser [1] (Petrov and Klein, 2007) and Maximum Entropy (MaxEnt) parsers significantly for both languages (in Section 5). While most of the previous work applies dependency to constituent tree conversion on the dependencies automatically derived from the Penn Treebank, the current work applies the technique to human-annotated English UD treebank as well. The constituent parser performance improvement due to the addition of converted treebanks is the first reported for English and German (in Section 6).

Throughout the paper, we use the notation CTree for a constituent tree, DTree for a dependency tree and UDTree for a universal dependency tree. We use the term 'constituent' and 'phrase' interchangeably. Conversion and parsing accuracies are reported in labeled F-scores.

2 Dependency to Partial Constituent Tree Conversion

We first derive a partial constituent tree from the source dependency tree. The partial constituent tree retains all of the human annotated part-of-speech tags and partial constituent brackets inferred from the source dependencies. Figure 2 is the deterministic algorithm that derives a partial CTree from any given DTree, where the dependency span of a word is a consecutive word sequence reachable from the word by head modifier relations.

Note that the algorithm in Figure 2 does not require any external knowledge sources such as head rules learned from the target CTrees. It applies to any DTrees that make a reasonable linguistic assumption on head-modifier relations regardless of languages and dependency types. This *simplicity* sets the current proposal apart from all of the previous proposals that rely on linguistic rules, as in (Xia et al., 2008), statistical model utilizing manually acquired head rules and the phrase labels of the target constituent treebank, as in (Kong et al., 2015), or a scoring function that computes the similarity between the source DTree and the nbest parsing output of the DTree sentences by the target constituent parser, as in (Niu et al., 2009).

[1] https://github.com/slavpetrov/berkeleyparser

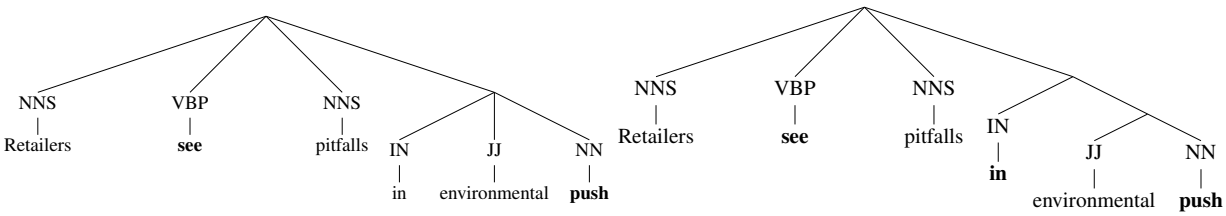
input: DTree (labeled or unlabeled) with *n* input words
output: Unlabeled CTree with gold POStags and partial constituent brackets

Step 1: Identify the dependency span D_i of each word w_i
if *the word* w_i *does not have any dependent* **then**
  D_i is length 1, containing only w_i itself;
else
  D_i subsumes all of its dependents recursively;

Step 2: Convert a dependency span D_i to a constituent C_i
Vertex of C_i dominates the immediate dependents of the head word and the head word itself.

Step 3: Remove all constituent brackets containing only one word.

Figure 2: DTree to unlabeled partial CTree Conversion Algorithm



(a) Partial CTree derived from UDTree in Figure 1(b)

(b) Partial CTree derived from DTrees in Figure 1(c, d)

Figure 3: Partial CTrees derived from the DTrees in Figure 1 according to the algorithm in Figure 2

The UDTree in Figure 1(b) is converted to the partial CTree in Figure 3(a) and the DTrees in Figure 1(c, d) are converted to the partial CTree in Figure 3(b) according to the algorithm in Figure 2. The head word of each constituent is in bold-face. Similarity of the head-modifier definitions between the target CTree and the source DTree is reflected on the partial CTrees. The partial CTree in Figure 3(a) derived from the UDTree leaves more ambiguity within the prepositional phrase covered by *in environmental push* than the one derived from CoNLL or Stanford DTrees. Similarity between a given DTree representation and the Penn Treebank CTree is reflected on the conversion accuracy reported in Section 4.

3 Constraint-based Maximum Entropy Parsing

To derive the fully specified labeled CTree from a partial CTree, we parse the input sentence with a constraint-based constituent parser that utilizes the gold POStags and partial brackets as model external constraints.

We implement the constraint-based parsing algorithm on the maximum entropy parser of (Ratnaparkhi, 1997; Ratnaparkhi, 1999), which works robustly regardless of the grammar coverage of the baseline parsing model and therefore well-suited for constraint-based parsing of partial CTrees derived from out-of-domain as well as in-domain DTrees.

3.1 Baseline Maximum Entropy Parser

The baseline MaxEnt parser takes one of the four actions to parse an input sentence: *tag* , *chunk* , *extend* and *reduce*. Four models corresponding to each action are built separately during training.

The model score in (1) is integrated into the parser scoring function (2). In (1) and (2), a_i is an action from *tag*, *chunk*, *extend* or *reduce*, and b_i is the context for a_i.

$$q(a_i|b_i) = p_{a_i}(a_i|b_i) \quad (1)$$

$$score(T) = \prod_{a_i \in deriv(T)} q(a_i|b_i) \quad (2)$$

deriv(T) in (2) is the derivation of a parse *T*, which may not be complete. Given the scoring function (2), a beam search heuristic attempts to find the best parse T^*, defined in (3) where *trees(S)* are all the complete parses for an input sentence *S*.

input: Input sentence with partial CTree
output: Complete labeled CTree

Parser Initialization;
M = 20 & K = 80 & Q = 0.95;
C = ¡empty heap¿ h_0 = ¡input sentence¿;
while —C— ¡ M **do**
  if *($\forall i$, h_i is empty)* **then**
    break
  else
    i = max $\{i$ — h_i is non-empty$\}$;
    sz = min $(K, |h_i|)$;
    for j = *1 to sz* **do**
      if $\exists$ h_c **then**
        d_c=advance(extract (h_c))
      else
        $d_1 d_p$ = advance(extract (h_i), Q)
      for q = *1 to p* **do**
        if *completed* (d_q) **then**
          insert (d_q, C)
        else
          insert (d_q, h_{i+1})

Figure 4: Constraint-based parsing algorithm

Constraints	WSJ-22	BOLT-DF
Baseline w/o constraints	88.57	82.43
Gold POStag	89.50	85.09
Gold bracket	98.52	96.88
Gold POStag+bracket	98.74	98.02

Table 1: Impact of model external constraints on parsing F-scores. The constraints Gold POStag, Gold bracket denote the POStags and constituent brackets read off from the human annotated gold CTrees. Combination of gold brackets and gold labels are equivalent to gold CTrees.

$$T^* = arg \max_{T \in trees(S)} score(T) \tag{3}$$

The parser explores the top K scoring parses and terminates when M complete parses are found or all hypotheses are exhausted. Possible actions $a_1 a_n$ on a derivation are sorted according to the model score $q(a_i|b_i)$. Only the actions $a_1 a_m$ with the highest probabilities are considered.

3.2 Constraint-based Maximum Entropy Parsing

In constraint-based parsing, the parser actions are based not only on the trained model scores but also on external constraints, which aim to improve the parsing qualities not achievable by parsing models alone.

The model external constraints include gold (i.e. human annotated) POStags, gold constituent brackets and/or gold labels. We enforce the parser to choose the gold tags, gold constituent brackets and labels over those selections made by the parsing model scores. When gold tags are provided as constraints, the tag action accepts the gold tag as the output. When gold constituent brackets (and labels) are given, the parser $chunk$, $extend$ and $reduce$ actions accept the gold constituent spans and their labels over the highest scoring model hypotheses. Figure 4 shows the constraint-based parsing algorithm.

The parameters M, K are described in Section 3.1. C denotes the heap of completed parses. h_i contains the derivations of length i. h_c contains the derivation with a constraint. Q is the probability pruning threshold. Advance applies relevant actions to a derivation d and returns a list of new derivations $d_1 d_n$. If there is a model external constraint for an action, it returns the derivation with the constraint d_c. Otherwise, it returns the derivations with the highest probabilities until the probability mass of the actions is greater than the threshold Q. Insert inserts a derivation d in heap h. Extract returns a derivation in h. Completed returns true if and only if d is a complete derivation.

Applying the constraint-based parsing algorithm in Figure 4 to the input sentence *Retailers see pitfalls in environmental push* with the partial CTrees in Figure 3 as the constraints, the parser produces the labeled CTree in Figure 1(a). Impact of model external constraints on parsing F-scores is shown in Table 1. The constraints Gold POStag, Gold bracket denote the POStags and constituent brackets read off from the human annotated gold CTrees. Combination of gold brackets and gold labels are equivalent to gold CTrees. Note that gold constituent brackets alone lead to very high F-scores for WSJ-22, 98.52 and BOLT-DF, 96.88. Our proposal capitalizes on the effectiveness of human annotated gold POStags

Techniques	Dependencies	F-score
(Xia et al., 2008)	CoNLL	89.4
(Niu et al., 2009)	Unlabeled	93.8
Current 2-stage	CoNLL	95.5
Current 2-stage	Stanford-v1.6.8	95.6

Table 2: DTree to CTree conversion F-scores on WSJ-22

Dependencies	DevSet	EvalSet
Stanford-v1.6.8	92.88	92.06
CoNLL	92.50	91.74
Universal Dependency	91.22	90.48

Table 3: DTree to CTree conversion F-scores on EWT according to various dependencies

and constituent brackets on parsing even when they are provided only partially, and utilize the partial CTrees derived from human annotated DTrees to recover the complete CTrees.

4 Conversion Accuracy

To compare the performance of the current conversion technique (Current 2-stage) with the previous work, all of which use the DTrees automatically derived from the Penn Treebank as the source dependency, we show the conversion accuracy on WSJ-22 in Table 2. The proposed 2-stage technique achieves the state-of-the-art conversion F-score 95.6 without relying on language and/or target treebank specific head rules. The constraint-based MaxEnt parser is trained on WSJ02-21.[2]

We also show the conversion accuracy of the current technique on English Web Treebank (EWT, LDC2012T13) from three types of dependencies in Table 3: Stanford basic dependency converted from the Penn Treebank by Stanford parser v1.6.8, CoNLL dependency converted from the Penn Treebank by pennconverter.jar[3], and human-annotated UD of (Nivre et al., 2015)[4]. MaxEnt parser for the constraint-based parsing is trained on English Ontonotes-5 treebank. The EWT train/development/evaluation data partitions are the same as those available from the UD.[5] Conversion F-scores are computed with `evalb`, excluding punctuations.

5 Parsing Experimental Results

Our ultimate goal is to improve constituent parsing accuracy by leveraging dependency treebanks available in a wide variety. To achieve this objective, we first convert dependency treebanks into constituent representations using the proposed conversion technique. Then we merge the converted treebanks with the human-annotated constituent treebank to enlarge the training set of constituent parser. We finally re-train the constituent parsers with the enlarged training set. We report the parsing experimental results for English and German.

English parser training and evaluation data sets from Ontonotes-5 (LDC2013T19) and EWT are shown in Table 4. Ontonotes-5 is the biggest constituent treebank available in English and includes sub-corpora from 7 genres. German parser training and evaluation data sets are shown in Table 5.

We experiment with two constituent parsers. The MaxEnt parser which we adapted for the constraint-based parsing and the BerkeleyParser. We measure the labeled F-scores including punctuations so that all sentences are scored correctly even when there is a mismatch of punctuation tags between the reference and machine parses.[6]

5.1 English Results

We train the baseline parser on the Ontonotes-5 training corpus only (Baseline in Tables 6 and 7). UD treebank corresponding to the training portion of EWT is converted to CTrees, using the proposed conversion technique with the constraint-based MaxEnt parser, and the converted treebank is added to the Ontotnotes-5 treebank for parser training (+Converted in Tables 6 and 7). We also train parsers on both Ontonotes-5 treebank and the EWT training corpus (+Gold in Tables 6 and 7).

[2] (Niu et al., 2009) automatically derive their dependencies from the Penn Trees using head percolation table.

[3] Downloaded from http://nlp.cs.lth.se/coftware/treebank-converter

[4] v1.1 downloaded from http://universaldependencies.org

[5] The gold stanford English UD was built over the source material of the EWT. That is, UD and EWT are parallel.

[6] Ontonotes-5 and EWT are quite noisy and quite a few sentences contain punctuation tag mismatches.

	Training Data		Development Data		Evaluation Data	
Genre	sent #	token #	sent #	token #	sent #	token #
WB	~14k	~323k	800	~18k	800	~18k
MZ	~6.7k	~159k	800	~19k	800	~19k
NW	~42k	~1m	800	~20k	800	~19k
BN	~12k	~229k	800	~15k	800	~15k
BC	~15k	~221k	800	~12k	800	~12k
TC	~14k	~109k	800	~6.2k	800	~6k
PT	~24k	~326k	800	~11k	800	~11k
EWT	~125k	~205k	2,002	~25k	2,077	~25k

Table 4: English Ontonotes (WB, MZ, NW, BN, BC, TC, PT) and English Web Treebank (EWT) data partition into baseline parser train (Ontonotes), converted train (EWT), development and evaluation data sets

Data Sets	sent #	token #
Baseline train	~18.5k	~332k
Converted train	~18.5k	~328k
Development	1,061	~18.5k
Evaluation	1,060	~18k

Table 5: German Tiger Treebank data partition into baseline parser train, converted train, development and evaluation data sets

Eval Set	Baseline	+Converted	+Gold
EWT	79.30	80.78	82.22
WB	82.94	83.50	83.67
MZ	85.01	85.64	85.96
NW	86.82	87.01	87.35
BN	87.25	87.31	87.43
BC	82.21	*81.72*	*82.14*
TC	81.45	*81.41*	*81.23*
PT	94.72	94.78	95.11

Table 6: English MaxEnt parser F-scores

Eval Set	Baseline	+Converted	+Gold
EWT	78.34	79.12	80.21
WB	83.48	*82.94*	*83.15*
MZ	86.10	86.48	86.35
NW	86.17	86.46	86.64
BN	85.49	85.73	86.31
BC	81.67	*81.34*	*81.64*
TC	77.40	*76.65*	*76.12*
PT	91.92	*91.79*	*91.91*

Table 7: English BerkeleyParser F-scores

For both MaxEnt and Berkeley parsers, addition of the converted treebank improves the F-scores of the EWT evaluation data much more than other evaluation data sets from the Ontnotes-5 treebank, as expected. The converted treebank also improves the F-scores of WB, MZ, NW, BN and PT for the MaxEnt parser and MZ, NW and BN for BerkeleyParser. Not surprisingly, addition of the gold EWT improves the parser performance more than addition of the converted treebank. When the addition of the converted treebank hurts the parser performance, we see that the same downward pattern holds even with the addition of the gold EWT, as indicated by italics in Tables 6 and 7.

5.2 German Results

Tiger constituent treebank has the corresponding CoNLL2006 dependency treebank. We split the Tiger treebank training data into two parts, one for the baseline constituent parser training, and the other for conversion from the CoNLL dependency treebank. Experimental results are shown in Table 8. We observe the same pattern of improvement as English in a bigger margin.

6 Related Work and Conclusions

In the family of DTree to CTree conversion technique, the current work is closest in spirit to (Niu et al., 2009). They generate N-best parses of the dependency treebank sentences using the constituent parser and compare the similarity between N-best constituent parses and the source dependencies by converting the N-best parses back to dependencies. They show that addition of converted Chinese dependency treebank to CTB, (Xue et al., 2005), improves the Chinese constituent parsing accuracy modestly. (Xia

training data / parser	Baseline	+ Converted Treebank	+ Gold Treebank
MaxEnt parser	75.74	76.88	78.01
BerkeleyParser	71.88	73.42	76.12

Table 8: Contituent parsing improvement due to the DTree-to-CTree converted treebank and the gold constituent treebank

et al., 2008) propose a rule-based DTree to CTree conversion technique, assuming that the input DTree is identical to a flattened version of the desired CTree. They decompose the input DTree into multiple DTree segments, replacing each segment with the CTree counterparts and glue the CTree segments to form a complete CTree. The idea of utilizing dependency boundaries as constraints on constituent parsing has been explored in (Wang and Zong, 2010).

In the family of bi-directional conversion between CTrees and DTrees, (Hall and Nivre, 2008) present a dependency driven parser that parses both dependency and constituent structures. They automatically transform constituent representations into complex dependency representations so that they can recover the constituent structure. (Kong et al., 2015) propose a statistical model to transform DTrees into CTrees. They first convert CTrees to DTrees, which encode the rich head-modifier and phrase label information from the CTrees.[7] They train a statistical model to restore the CTrees from the feature-rich DTrees. While they report their DTree to CTree conversion accuracy on WSJ-22, their accuracy is not directly comparable to those we report in Tables 2 and 3 since their DTrees encodes head-modifier relations and phrase labels read off from the corresponding gold CTrees. (Fernández-Golzález and Martins, 2015) derive head-ordered DTrees from CTrees, train an off-the-shelf dependency parser on the DTrees, and recover the constituent information from the head-ordered DTrees. These bi-directional techniques practically reduce constituent parsing to dependency parsing and are applied to DTrees that encode the same complex information as the corresponding CTrees in order to easily recover the phrase structures.

We presented a simple DTree to CTree conversion technique that aims to improve constituent parsing accuracies by leveraging dependency treebanks available in a wide variety in many languages. Evaluated on WSJ-22, the technique achieves the state-of-the-art conversion F-score 95.6. When applied to English and German, the converted treebanks added to the constituent parser training corpus improve parsing F-scores significantly for both languages.

Acknowledgements

We would like to acknowledge the anonymous reviewers for their helpful comments.

References

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 173–180.

Michael Collins, Lance Ramshaw, Jan Hajic, and Christoph Tillmann. 1999. A Statistical Parser for Czech. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, pages 505–512.

Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of LREC*.

Daniel Fernández-Golzález and André F. T. Martins. 2015. Parsing as Reduction. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*.

Martin Forst. 2003. Treebank Conversion - Establishing a Testsuite for a Broad-Coverage LFG from the TIGER Treebank. In *Proceedings of LING at EACL*, pages 25–32.

Johan Hall and Joakim Nivre. 2008. A dependency-driven parser for German dependency and constituency representations. In *Proceedings of the Workshop on Parsing German*, pages 47–54.

Richard Johansson and Pierre Nugues. 2007. Extended constituent-to-dependency conversion for English. In *Proceedings of NODALIDA 2007*, pages 105–112, Tartue, Estonia.

Lingpeng Kong, Alexander M. Rush, and Noah A. Smith. 2015. Transforming Dependencies into Phrase Structures. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT)*.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of English: the penn treebank. *Computational Linguistics - Special issue on using large corpora: II, Volume 19 Issue 2, June 1993*, pages 313–330.

[7] https://github.com/ikekonglp/PAD/tree/master/python

Ryan McDonald, F. Pereira, K. Ribarov, and J. Hajic. 2005. Non-Projective Dependency Parsing using Spanning Tree Algorithms. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP) 2005*.

Zheng-Yu Niu, Haifeng Wang, and Hua Wu. 2009. Exploiting Heterogeneous Treebanks for Parsing. In *Proceedings of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing*, pages 46–54.

Joakim Nivre, Cristina Bosco, Jinho Choi, MarieCatherine de Marneffe, Timothy Dozat, Richard Farkas, Jennifer Foster, Filip Ginter, Yoav Goldberg, Jan Hajic, Jenna Kanerva, Veronika Laippala, Alessandro Lenci, Teresa Lynn, Christopher Manning, Ryan McDonald, Anna Missila, Simonetta Montemagni, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Maria Simi, Aaron Smith, Reut Tsarfaty, Veronika Vincze, and Daniel Zeman. 2015. Universal Dependencies 1.0.

Slav Petrov and Dan Klein. 2007. Improved Inference for Unlexicalized Parsing. In *Proceedings of NAACL-HLT*, pages 404–411.

Owen Rambow. 2010. The Simple Truth about Dependency and Phrase Structure Representations: An Opinion Piece. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT)*, pages 337–340.

Adwait Ratnaparkhi. 1997. A Linear Observed Time Statistical Parser Based on Maximum Entropy Models. In *Proceedings of Empirical Methods in Natural Language Processing (EMNLP)*.

Adwait Ratnaparkhi. 1999. Learning to Parse Natural Language with Maximum Entropy Models. *Machine Learning 34*, pages 151–175.

Katalin Ilona Simkó, Veronika Vencze, Zsolt Szánto, and Richárd Farkas. 2014. An Emripical Evaluation of Automatic Conversion from Constituency to Dependency in Hugarian. In *Proceedings of the 25th COLING*, pages 1392–1401.

Zhiguo Wang and Chengqing Zong. 2010. Phrase structure parsing with dependency structure. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, pages 1292–1300. Association for Computational Linguistics.

Jong-Nae Wang, Jing-Shin Chang, and Keh-Yih Su. 1994. An Automatic Treebank Conversion Algorithm for Corpus Sharing. In *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics*, pages 248–254.

Fei Xia, Rajesh Bhatt, Owen Rambow, Martha Palmer, and Dipti Misra Sharma. 2008. Towards a Multi-Representational Treebank. In *Proceedings of the 7th International Workshop on Treebanks and Linguistic Theories*, pages 159–170.

Nianwen Xue, Fei Xia, Fu-Dong Chiou, and Martha Palmer. 2005. The Penn Chinese Treebank: Phrase Structure Annotation of a Large Corpus. *Natural Language Engineering, 11(2)*, pages 207–238.

Hiroyasu Yamada and Yuji Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *Proceedings of International Workshop on Parsing Technologies, Volume 3*.

Yuan Zhang, Regina Barzilay, and Amir Globerson. 2013. Transfer Learning for Constituency-Based Grammars. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 291–301.