Parser Evaluation Using Derivation Trees: A Complement to evalb

Seth Kulick and **Ann Bies** and **Justin Mott**
Linguistic Data Consortium, University of Pennsylvania, Philadelphia, PA 19104
{skulick,bies,jmott}@ldc.upenn.edu

Anthony Kroch and **Mark Liberman** and **Beatrice Santorini**
Department of Linguistics, University of Pennsylvania, Philadelphia, PA 19104
{kroch,myl,beatrice}@ling.upenn.edu

Abstract

This paper introduces a new technique for phrase-structure parser analysis, categorizing possible treebank structures by integrating regular expressions into derivation trees. We analyze the performance of the Berkeley parser on OntoNotes WSJ and the English Web Treebank. This provides some insight into the evalb scores, and the problem of domain adaptation with the web data. We also analyze a "test-on-train" dataset, showing a wide variance in how the parser is generalizing from different structures in the training material.

1 Introduction

Phrase-structure parsing is usually evaluated using evalb (Sekine and Collins, 2008), which provides a score based on matching brackets. While this metric serves a valuable purpose in pushing parser research forward, it has limited utility for understanding what sorts of errors a parser is making. This is the case even if the score is broken down by brackets (NP, VP, etc.), because the brackets can represent different types of structures. We would also like to have answers to such questions as "How does the parser do on non-recursive NPs, separate from NPs resulting from modification? On PP attachment?" etc.

Answering such questions is the goal of this work, which combines two strands of research. First, inspired by the tradition of Tree Adjoining Grammar-based research (Joshi and Schabes, 1997; Bangalore and Joshi, 2010), we use a decomposition of the full trees into "elementary trees" (henceforth "etrees"), with a derivation tree that records how the etrees relate to each other, as in Kulick et al. (2011). In particular, we use the "spinal" structure approach of (Shen et al., 2008; Shen and Joshi, 2008), where etrees are constrained to be unary-branching.

Second, we use a set of regular expressions (henceforth "regexes") that categorize the possible structures in the treebank. These are best thought of as an extension of head-finding rules, which not only find a head but simultaneously identify each parent/children relation as one of a limited number of types of structures (right-modification, etc.).

The crucial step is that we integrate these regexes into the spinal etrees. The derivation trees provide elements of a dependency analysis, which allow us to calculate scores for head identification and attachment for different projections (e.g., PP). The regexes allow us to also provide scores based on spans of different construction types. Together these two aspects break down the evalb brackets into more meaningful categories, and the simultaneous head and span scoring allows us to separate these aspects in the analysis.

After describing in more detail the basic framework, we show some aspects of the resulting analysis of the performance of the Berkeley parser (Petrov et al., 2008) on three datasets: (a) OntoNotes WSJ sections 2-21 (Weischedel et al., 2011)[1], (b) OntoNotes WSJ section 22, and (c) the "Answers" section of the English Web Treebank (Bies et al., 2012). We trained the parser on sections 2-21, and so (a) is "test-on-train". These three results together show how the parser is generalizing from the training data, and what aspects of the "domain adaptation" problem to the web material are particularly important.[2]

2 Framework for analyzing parsing performance

We first describe the use of the regexes in tree decomposition, and then give some examples of in-

[1]We refer only to the WSJ treebank portion of OntoNotes, which is roughly a subset of the Penn Treebank (Marcus et al., 1999) with annotation revisions including the addition of NML nodes.

[2]We parse (c) while training on (a) to follow the procedure in Petrov and McDonald (2012)

corporating these regexes into the derivation trees.

2.1 Use of regular expressions

Decomposing the original phrase-structure tree into the smaller components requires some method of determining the "head" of a nonterminal, from among its children nodes, similar to parsing work such as Collins (1999). As described above, we are also interested in the type of linguistic construction represented by that one-level structure, each of which instantiates one of a few types - recursive coordination, simple head-and-sister, etc. We address both tasks together with the regexes. In contrast to the sort of head rules in (Collins, 1999), these refer as little as possible to specific POS tags. Instead of explicitly listing the POS tags of possible heads, the heads are in most cases determined by their location in the structure.

Sample regexes are shown in Figure 1. There are 49 regexes used.[3] Regexes ADJP-t and ADVP-t in (a) identify their terminal head to be the rightmost terminal, possibly preceded by some number of terminals or nonterminals, relying on a mapping that maps all terminals (except CC, which is mapped to CONJ) to TAG and all nonterminals (except CONJP and NML) to NT. Structures with a CONJ/CONJP/NML child do not match this rule and are handled by different regexes, which are all mutually exclusive. In some cases, we need to search for particular nonterminal heads, such as with the (b) regexes S-vp and SQ-vp, which identify the rightmost VP among the children of a S or SQ as the head. (c) NP-modr is a regex for a recursive NP with a right modifier. In this case, the NP on the left is identified as the head. (d) VP-crd is also a regex for a recursive structure, in this case for VP coordination, picking out the leftmost conjunct as the head of the structure. The regex names roughly describe their purpose - "mod" for right-modification, "crd" for coordination, etc. The suffix "-t" is for the simple non-recursive case in which the head is a terminal.

2.2 Regexes in the derivation trees

The names of these regexes are incorporated into the etrees themselves, as labels of the nonterminals. This allows an etree to contain information such as "this node represents right modification".

(a)ADJP-t,ADVP-t:
`^(TAG|NT|NML)*(head:TAG) (NT)*$`
(b)S-vp, SQ-vp: `^([^ ]+)*(head:VP)$`
(c)NP-modr:
`^(head:NP)(SBAR|S|VP|ADJP|PP|ADVP|NP)+$`
(d)VP-crd: `^(head:VP) (VP)* CONJ VP$`

Figure 1: Some sample regexes

For example, Figure 2 shows the derivation tree resulting from the decomposition of the tree in Figure 4. Each structure within a circle is one etree, and the derivation as a whole indicates how these etrees are combined. Here we indicate with arrows that point to the relevant regex. For example, the PP-t etree #a6 points to the NP-modr regex, which consists of the NP-t together with the PP-t. The nonterminals of the spinal etrees are the names of the regexes, with the simpler nonterminal labels trivially derivable from the regex names.[4]

The tree in Figure 5 is the parser output corresponding to the gold tree in Figure 4, and in this case gets the PP-t attachment wrong, while everything else is the same as the gold.[5] This is reflected in the derivation tree in Figure 3, in which the NP-modr regex is absent, with the NP-t and PP-t etrees #b5 and #b6 both pointing to the VP-t regex in #b3. We show in Section 2.3 how this derivation tree representation is used to score this attachment error directly, rather than obscuring it as an NP bracket error as evalb would do.

2.3 Scoring

We decompose both the gold and parser output trees into derivation trees with spinal etrees, and score based on the regexes projected by each word. There is a match for a regex if the corresponding words in gold/parser files project to that regex, a precision error if the parser file does but the gold does not, and a recall error if the gold does but the parser file does not.

For example, comparing the trees in Figures 4 and 5 via their derivation trees in Figures 2 and Figures 3, the word "trip" has a match for the regex NP-t, but a recall error for NP-modr. The word

[3]Some among the 49 are duplicates, used for different nonterminals, as with (a) and (b) in Figure 1. We derived the regexes via an iterative process of inspection of tree decomposition on dataset (a), together with taking advantage of the treebanking experience from some of the co-authors.

[4]We do not have space here to discuss the data structure in complete detail, but multiple regex names at a node, such a VP-aux and VP-t at tree a3 in Figure 2, indicate multiple VP nonterminals.

[5]We leave function tags aside for future work. The gold tree is shown without the SBJ function tag.

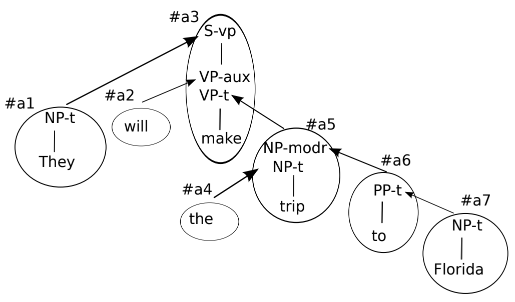

Figure 2: Derivation Tree for Figure 4

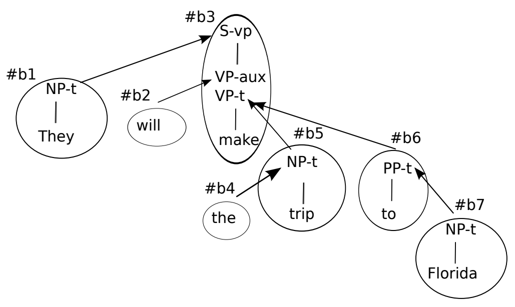

Figure 3: Derivation Tree for Figure 5)

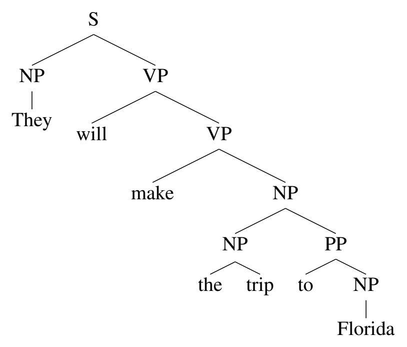

Figure 4: Gold tree

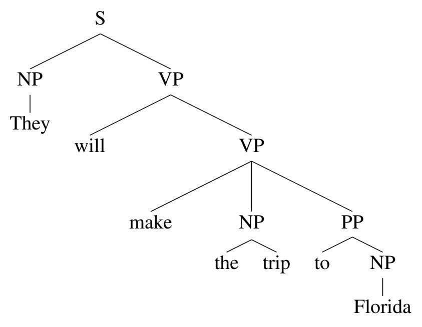

Figure 5: Parser output tree

Corpus	tokens	brackets	coverage	%	evalb
2-21 g	650877	578597	571243	98.7	
p		575744	569480	98.9	93.8
22 g	32092	24819	24532	98.8	
p		24801	24528	98.9	90.1
Ans g	53960	48492	47348	97.6	
p		48750	47423	97.3	80.8

Table 1: Corpus information for gold(g) and parsed(p) sections of each corpus

"make" has a match for the regexes VP-t, VP-aux, and S-vp, and so on. Summing such scores over the corresponding gold/parser trees gives us F-scores for each regex.

There are two modifications/extensions to these F-scores that we also use:
(1) For each regex match, we score whether it matches based on the span as well. For example, "make" is a match for VP-t in Figures 2 and 3, and is also a match for the span as well, since in both derivation trees it includes the words "make. . .Florida". It is this matching for span as well as head that allows us to compare our results to evalb. We call the match just for the head the "F-h" score and the match that also includes the span information the "F-s" score. The F-s score roughly corresponds to the evalb score. However, the F-s score is for separate syntactic constructions (including also head identification), although we can also sum it over all the structures, as done later in Figure 6. The simultaneous F-h and F-s scores let us identify constructions where the parser has the head projection correct, but gets the span wrong.
(2) Since the derivation tree is really a dependency tree with more complex nodes (Rambow and Joshi, 1997; Kulick et al., 2012), we can also score each regex for attachment.[6] For example, while "to" is a match for PP-t, its attachment is not, since in Figure 2 it is a child of the "trip" etree (#a5) and in Figure 3 it is a child of the "make" etree (#b3). Therefore our analysis results in an attachment score for every regex.

2.4 Comparison with previous work

This work is in the same basic line of research as the inter-annotator agreement analysis work in Kulick et al. (2013). However, that work did not utilize regexes, and focused on comparing sequences of identical strings. The current work scores on general categories of structures, without

[6]A regex intermediate in a etree, such as VP-t above, is considered to have a default null attachment. Also, the attachment score is not relevant for regexes that already express a recursive structure, such as NP-modr. In Figure 2, NP-t in etree #a5 is considered as having the attachment to #a3.

	Sections 2-21 (Ontonotes)					Section 22 (Ontonotes)					Answers (English Web Treebank)				
regex	%gold	F-h	F-s	att	spanR	%gold	F-h	F-s	att	spanR	%gold	F-h	F-s	att	spanR
NP-t	30.7	98.9	97.6	96.5	99.6	31.1	98.0	95.8	94.4	99.6	28.5	95.4	91.5	90.9	99.3
VP-t	13.5	98.8	94.5	98.4	95.8	13.4	98.1	91.7	97.3	93.7	16.0	96.7	81.7	96.1	85.4
PP-t	12.2	99.2	91.0	90.5	92.0	12.1	98.7	86.4	86.1	88.2	8.4	96.4	80.5	80.7	84.7
S-vp	12.2	97.9	92.8	96.8	96.3	11.9	96.5	89.1	95.4	95.0	14.2	94.1	72.9	88.0	84.1
NP-modr	8.6	88.4	80.3	-	91.5	8.5	82.9	71.8	-	87.9	4.4	69.0	54.2	-	80.5
VP-aux	5.5	97.9	94.0	-	96.1	5.0	96.5	91.0	-	94.6	6.2	94.4	81.7	-	86.7
SBAR-s	3.7	96.1	91.1	91.8	95.3	3.5	94.3	87.2	86.4	93.5	4.0	84.8	68.2	81.9	81.9
ADVP-t	2.7	95.2	93.3	93.9	98.6	3.0	89.6	84.5	88.0	95.9	4.5	84.0	78.2	80.3	96.8
NML-t	2.3	91.6	90.3	97.6	99.8	2.6	85.6	82.2	93.5	99.8	0.7	42.1	37.7	88.8	100.0
ADJP-t	1.9	94.6	88.4	95.5	94.6	1.8	86.8	77.0	93.6	90.7	2.5	84.7	67.0	88.1	84.2
QP-t	1.0	95.3	93.8	98.3	99.6	1.2	91.0	89.0	97.1	100.0	0.2	57.7	57.7	94.4	100.0
NP-crd	0.8	80.3	73.7	-	92.4	0.6	68.6	58.4	-	86.1	0.5	55.3	47.8	-	88.1
VP-crd	0.4	84.3	82.8	-	98.2	0.4	75.3	73.5	-	97.6	0.8	65.5	58.3	-	89.8
S-crd	0.3	83.7	83.2	-	99.6	0.4	70.9	68.6	-	96.7	0.8	68.5	63.0	-	93.4
SQ-v	0.1	88.3	82.0	93.3	97.8	0.1	66.7	66.7	88.9	100.0	0.9	81.9	72.4	93.4	95.8
FRAG-nt	0.1	49.9	48.6	95.4	97.9	0.1	28.6	28.6	100.0	100.0	0.8	22.7	21.3	96.3	96.3

Table 2: Scores for the most frequent categories of brackets in the three datasets of corpora, as determined by the regexes. % gold is the frequency of this regex type compared to all the brackets in the gold. F-h is the score based on matching heads, F-s also incorporates the span information, att is the attachment accuracy for words that match in F-h, and spanR is the span-right accuracy for words that match in F-h.

the reliance on sequences of individual strings.[7]

3 Analysis of parsing results

We worked with the three datasets as described in the introduction. We trained the parser on sections 2-21 of OntoNotes WSJ, and parsed the three datasets with the gold tags, since at present we wish to analyze the parser performance in isolation from Part-of-Speech tagging errors. Table 1 shows the sizes of the three corpora in terms of tokens and brackets, for both the gold and parsed versions, with the evalb scores for the parsed versions. The score is lower for Answers, as also found by Petrov and McDonald (2012).

To facilitate comparison of our analysis with evalb, we used corpora versions with the same bracket deletion (empty yields and most punctuation) as evalb. We ran the gold and parsed versions through our regex decomposition and derivation tree creation. Table 1 shows the number and percentage of brackets handled by our regexes. The high coverage (%) reinforces the point that there is a limited number of core structures in the treebank. In the results below in Table 2 and Figure 6 we combine the nonterminals that are not covered by one of the regexes with the simple non-recursive regex case for that nonterminal.[8]

[7]In future work we will compare our approach to that of Kummerfeld et al. (2012), who also move beyond evalb scores in an effort to provide more meaningful error analysis.

[8]We also combine a few other non-recursive regexes together with NP-t, such as the special one for possessives.

We present the results in two ways. Table 2 lists the most frequent categories in the three datasets, with their percentage of the overall number of brackets (%gold), their score based just on the head identification (F-h), their score based on head identification and (left and right) span (F-s), and the attachment (att) and span-right (spanR) scores for those that match based on the head.[9]

The two graphs in Figure 6 show the cumulative results based on F-h and F-s, respectively. These show the cumulative score in order of the frequency of categories. For example, for sections 2-21, the score for NP-t is shown first, with 30.7% of the brackets, and then together with the VP-t category, they cover 45.2% of the brackets, etc.[10] The benefit of the approach described here is that now we can see the contribution to the evalb score of the particular types of constructions, and within those constructions, how well the parser is doing at getting the same head projection, but failing or

[9]The score for the left edge is almost always very high for every category, and we just list here the right edge score. The attachment score does not apply to the recursive categories, as mentioned above.

[10]The final F-s value is lower than the evalb score - e.g. 92.5 for sections 2-21 (the rightmost point in the graph for sections 2-21 in the F-s graph in Figure 6) compared to the 93.8 evalb score. Space prevents full explanation, but there are two reasons for this. One is that there are cases in which bracket spans match, but the head, as found by our regexes, is different in the gold and parser trees. The other cases is when brackets match, and may even have the same head, but their regex is different. In future work we will provide a full accounting of such cases, but they do not affect the main aspects of the analysis.

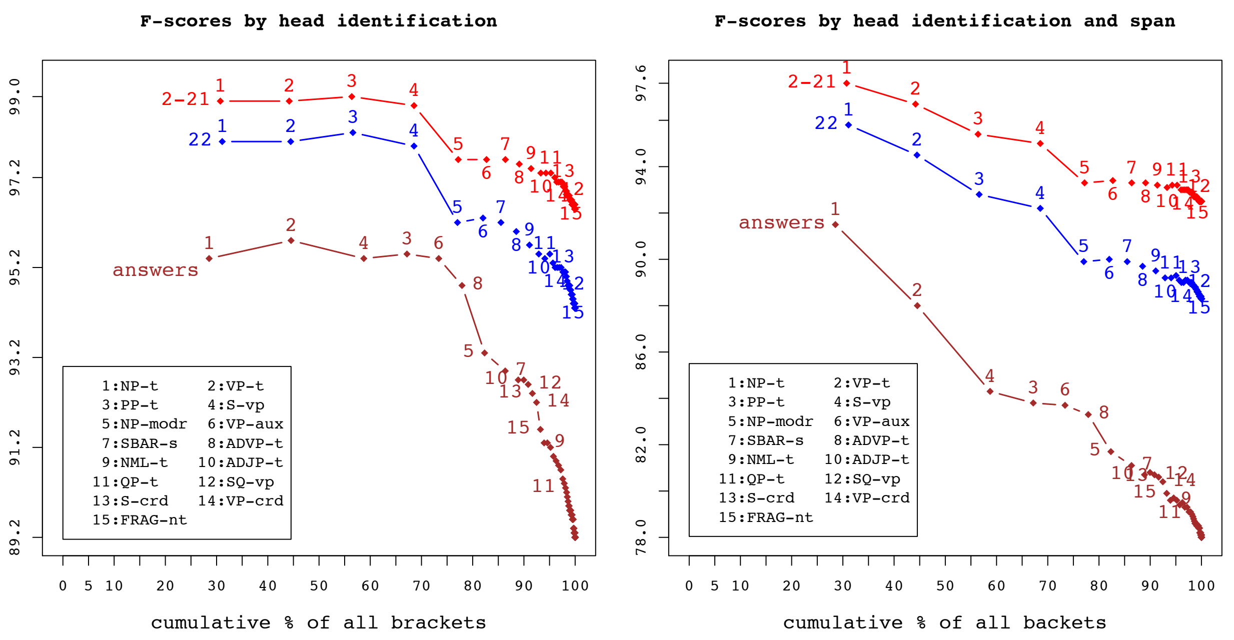

Figure 6: Cumulative scores based on F-h (left) and F-s (right). These graphs are both cumulative in exactly the same way, in that each point represents the total percentage of brackets accounted for so far. So for the 2-21 line, point 1, meaning the NP non-recursive regex, accounts for 30.7% of the brackets, point 2, meaning the VP non-recursive regex, accounts for another 13.5%, so 44.2% cumulatively, etc.

not on the spans.

3.1 Analysis and future work

As this is work-in-progress, the analysis is not yet complete. We highlight a few points here.

(1) The high performance on the OntoNotes WSJ material is in large part due to the score on the non-recursive regexes of NP-t, VP-t, S-vp, and the auxiliaries (points 1, 2, 4, 6 in the graphs). Critical to this is the fact that the parser does well on determining the right edge of verbal structures, which affects the F-s score for VP-t (non-recursive), VP-aux, and S-vp. The spanR score for VP-t is 95.8 for Sections 2-21 and 93.7 for Section 22.

(2) We wouldn't expect the test-on-training evalb score to be 100%, since it has to back off from the training data, but the results for the different categories vary widely, with e.g., the NP-modr F-h score much lower than other frequent regexes. This variance from the test-on-training dataset carries over almost exactly to Section 22.

(3) The different distribution of structures in Answers hurts performance. For example, the mediocre performance of the parser on SQ-vp barely affects the score with OntoNotes, but has a larger negative effect with Answers, due to its increased frequency in the latter.

(4) While the different distribution of constructions is a problem for Answers, more critical is the poor performance of the parser on determining the right edge of verbal constructions. This is only 85.4 for VP-t in Answers, compared to the OntoNotes results mentioned in (1). Since this affects the F-s scores for VP-t, VP-aux, and S-vp, the negative effect is large. Preliminary investigation shows that this is due in part to incorrect PP and SBAR placement (the PP-t and SBAR-s attachment scores (80.7 and 81.9) are worse for Answers compared to Section 22 (86.1 and 86.4)), and coordinated S-clauses with no conjunction.

In sum, there is a wealth of information from this new type of analysis that we will use in our ongoing work to better understand what the parser is learning and how it works on different genres.

Acknowledgments

This material is based upon work supported by National Science Foundation Grant # BCS-114749 (first, fourth, and sixth authors) and by the Defense Advanced Research Projects Agency (DARPA) under Contract No. HR0011-11-C-0145 (first, second, and third authors). The content does not necessarily reflect the position or the policy of the Government, and no official endorsement should be inferred.

References

Srinivas Bangalore and Aravind K. Joshi, editors. 2010. *Supertagging: Using Complex Lexical Descriptions in Natural Language Processing*. MIT Press.

Ann Bies, Justin Mott, Colin Warner, and Seth Kulick. 2012. English Web Treebank. LDC2012T13. Linguistic Data Consortium.

Michael Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, Department of Computer and Information Sciences, University of Pennsylvania.

A.K. Joshi and Y. Schabes. 1997. Tree-adjoining grammars. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages, Volume 3: Beyond Words*, pages 69–124. Springer, New York.

Seth Kulick, Ann Bies, and Justin Mott. 2011. Using derivation trees for treebank error detection. Association for Computational Linguistics.

Seth Kulick, Ann Bies, and Justin Mott. 2012. Using supertags and encoded annotation principles for improved dependency to phrase structure conversion. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 305–314, Montréal, Canada, June. Association for Computational Linguistics.

Seth Kulick, Ann Bies, Justin Mott, Mohamed Maamouri, Beatrice Santorini, and Anthony Kroch. 2013. Using derivation trees for informative treebank inter-annotator agreement evaluation. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 550–555, Atlanta, Georgia, June. Association for Computational Linguistics.

Jonathan K. Kummerfeld, David Hall, James R. Curran, and Dan Klein. 2012. Parser showdown at the wall street corral: An empirical investigation of error types in parser output. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1048–1059, Jeju Island, Korea, July. Association for Computational Linguistics.

Mitchell P. Marcus, Beatrice Santorini, Mary Ann Marcinkiewicz, and Ann Taylor. 1999. Treebank-3. LDC99T42, Linguistic Data Consortium, Philadelphia.

Slav Petrov and Ryan McDonald. 2012. Overview of the 2012 shared task on parsing the web. In *Proceedings of the First Workshop on Syntactic Analysis of Non-Canonical Language*.

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 2008. The Berkeley Parser. https://code.google.com/p/berkeleyparser/.

Owen Rambow and Aravind Joshi. 1997. A formal look at dependency grammars and phrase-structure grammars, with special consideration of word-order phenomena. In L. Wanner, editor, *Recent Trends in Meaning-Text Theory*, pages 167–190. John Benjamins, Amsterdam and Philadelphia.

Satoshi Sekine and Michael Collins. 2008. Evalb. http://nlp.cs.nyu.edu/evalb/.

Libin Shen and Aravind Joshi. 2008. LTAG dependency parsing with bidirectional incremental construction. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 495–504, Honolulu, Hawaii, October. Association for Computational Linguistics.

Libin Shen, Lucas Champollion, and Aravind Joshi. 2008. LTAG-spinal and the Treebank: A new resource for incremental, dependency and semantic parsing. *Language Resources and Evaluation*, 42(1):1–19.

Ralph Weischedel, Martha Palmer, Mitchell Marcus, Eduard Hovy, Sameer Pradhan, Lance Ramshaw, Nianwen Xue, Ann Taylor, Jeff Kaufman, Michelle Franchini, Mohammed El-Bachouti, Robert Belvin, and Ann Houston. 2011. OntoNotes 4.0. Linguistic Data Consortium LDC2011T03.