Proceedings of the Australasian Language Technology Workshop 2007, pages 75-82

Charting Democracy Across Parsers

Scott Nowson[1] **and Robert Dale**
Centre for Language Technology
Macquarie University
Sydney, Australia
{snowson|rdale}@ics.mq.edu.au

Abstract

Different parsers trained on the same corpus deliver different results, both in terms of overall performance and in terms of the individual analyses they provide. In particular, for any given sentence, one parser may provide a correct analysis, while another will produce an incorrect analysis; but when faced with a different sentence, the first parser may be in error while the second is correct. In this paper, we leverage this observation by exploring how the results of a number of different parsers may be combined to provide a better performance than any single parser. The method involves constructing a chart that contains edges contributed by a collection of parsers, with a simple voting mechanism to choose the most preferred constituents; this provides a significant improvement in performance over any individual parser. More sophisticated voting mechanisms are also discussed.

1 Introduction

Parsers make mistakes. This is perhaps most apparent when a parser trained on a given corpus is applied to data from a domain or genre different to that of the training corpus. One can, of course, retrain the parser on new data that is more representative of the texts to be handled; but annotation is an expensive process, and the literature does not provide a great deal of guidance as to how much annotation is required in order to obtain an acceptable result (but see Reichart and Rappoport (2007a) for some recent interesting results in this area).

Unfortunately, parsers make mistakes even on the corpora on which they are trained. Before we begin to consider how we might adapt a parser to a new domain, we are therefore interested in how we might improve the performance of existing parsers on the corpora used to derive their models.

We make the observation that different parsers have different 'error profiles', by which we mean that different parsers do not necessarily make the same mistakes. Consider the following verb phrase taken from our test corpus:

> *... lock in profits by buying futures when futures prices fall*

Figure 1 shows the analyses provided for this verb phrase by three different parsers, as an illustration of the kinds of disagreements that are common. In the first analysis, *in* is misclassified as a preposition, while in the second and third analyses it is correctly analysed as a particle. However, the second parse contains a misparse of the embedded VP *buying futures*, while this is correctly analysed in the first and third parses.

This leads us to the hypothesis that, if we were able to select for each parser those parts of individual parses that are more likely to be correct, then the overall result would be an improvement upon the analysis of any individual parser. We explore this hypothesis in this paper, by providing a framework within which the analyses of different parsers can be combined, and the overall best parse selected.

[1] Scott Nowson is now at Appen Pty Ltd.

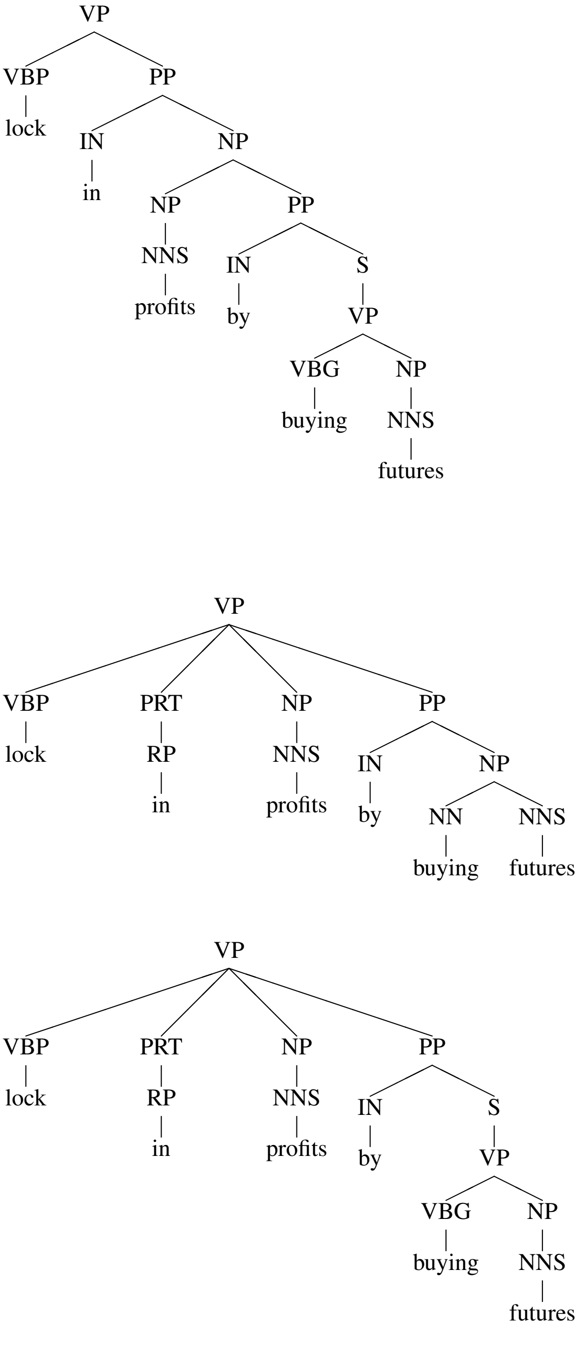

Figure 1: Three analyses by different parsers

The idea of combining the results of different parsers is not in itself new, so in Section 2 we briefly survey related work in this area. In Section 3 we describe our approach, which takes advantage of the central ideas in chart parsing to provide a way of combining parse results, and we describe the parsers used in our experiments. Section 4 describes the results achieved by our method, demonstrating a significant improvement upon the performance achieved by any individual parser. Section 5 discusses how the simple voting mechanism presented here can be made more elaborate, with the prospect of even better improvements in performance, and Section 6 concludes.

2 Background: Combining Parsers

The combination of the results of several different components that carry out the same task—sometimes referred to as the ensemble-based approach—has been employed and shown to be successful in a number of fields such as part-of-speech tagging (Halteren et al., 1998), word sense disambiguation (Pederson, 2000) and question answering (Chu-Carroll et al., 2003).

There are a number of approaches that have been employed for parser combination. Henderson and Brill (1999) describe experiments that fall within two general approaches they label *parse hybridization* and *parse switching*. The most basic form of hybridization is constituent voting, whereby constituents in a parse are included if they can be found in the majority of contributing parses. A second approach is to use a naïve Bayes classifier in order to learn how much each parser should be trusted.

The alternative to this approach is to deal only with complete parses. Henderson and Brill again experimented with two approaches: *similarity switching*, whereby the parse chosen is the one which scores highest when judged for similarity to the remaining parses in the set; and a second naïve Bayes approach to selecting the parse with the highest probability of being the best. All four of Henderson and Brill's approaches produced better results than any of the contributing parsers achieved: their best result was a 30% reduction of precision error rate, a 6% reduction of recall error rate and an absolute F-score increase of 1.58%. These ideas have been extended to take into account more context in adaptation to dependency based parsers with similarly successful results (Zeman and Žabokrtský, 2005).

Henderson and Brill (2000) followed their earlier combination approach with one based on creating an ensemble of complementary parsers. Each parser was based upon the same underlying algorithm, but trained on different data. By using bagging and boosting approaches, their ensemble outperformed all single parsers, with a 0.6% absolute improvement in F-score. In a similar vein, Reichart and Rappoport (2007b) generated a number of parsing models by training one parser on slightly different training corpora. The resulting outputs were compared in order to judge the parse quality: the greater the number of models in agreement, the higher the quality.

Clegg and Shepherd (2005) explored a number of alternative approaches to ensemble parsing when deploying trained parsers in a new domain. Using basic constituent voting based on Brill and Henderson's (1999) method, they report similar improvements, mostly to precision but also to recall. They achieved equally promising results from their variants of parse switching. The first of these was *fallback cascades* in which parsers are stacked in order of decreasing levels of sophistication. When the more complex model fails, the next parser attempts to parse. The bottom parser may be less accurate, but will be the least likely to fail. Their second whole-parse approach they simply termed *parse selection*, though it is similar to Henderson and Brill's similarity switching. Clegg and Shepherd varied this by trying different similarity metrics, such as constituent overlap or lineage similarity.

Sagae and Lavie (2006) apply a notion of reparsing to a two stage parser combination chart-based approach. Once all single parses are complete, the first stage is to store all possible constituents in a chart with a label, start and end positions, and a weighting. Identical constituents from different parses are merged by adding their weights. The second stage of the process is to run a bottom-up parsing algorithm, but rather than use a weighted grammar, the parser is guided by the weighted set of constituents. They experimented with different approaches to setting the initial weights of each non-terminal label. By combining five parsers they were able to achieve a error reduction of 44% for precision and 14% for recall, and an absolute F-score increase of 1.1% (though it is worth noting each of these are *best* improvements, made across a different run made with different settings).

3 Our Approach

3.1 The Basic Idea

Our approach is based on the central idea in chart parsing (Earley 1970; Kay 1980): for any ambiguous string, the constituents derived from multiple parses can be maintained in one data structure, so that subsequent parses can reuse previously derived partial analyses. The insight leveraged here is that the same idea can be applied to multiple parsers: just as the chart can contain multiple analyses for a string as delivered by one parser, it can just as easily contain multiple analyses delivered by several parsers, thus providing a single unified view of all the different analyses, and allowing us to easily determine where parsers agree and where they disagree.

This is a very simple idea, but one which enables the development of a variety of approaches to choosing which edges should be used in building a preferred parse; and, as we demonstrate below, even the simplest methods provide good results.

Our approach to combination is built upon a basic voting strategy, methodologically similar to Henderson and Brill (1999) and Sagae and Lavie (2006), with implementational similarity to the latter. In its purest form, voting is purely democratic: all nominated constituents are considered equally suitable candidates to fill a position in the parse, and the candidate with the most votes—i.e., the candidate proposed by a majority of the parsers—is the winner. The algorithm is simple:

1. Each sentence is parsed by multiple parsers.
2. The resulting Penn Treebank parse strings are converted into a chart representation.
3. Starting with the root node, voting takes place as to what the children of that node should be.
4. Step 3 is then repeated for each successful child node.
5. When the tree is fully populated by terminal nodes, the final chart is returned as a Penn Treebank parse representation for evaluation.

We now describe these stages in more detail.

3.2 Parsing

The first stage in our process is to parse each sentence with the individual contributing parsers. In the experiments reported here, we use three parsers: the Stanford lexicalised parser (Klein and Manning, 2003); Collins generative parsing model number 2 (Collins, 1999) as re-implemented by Bikel (2004); and the OpenNLP parser (Baldridge et al., 2003). These were chosen for two reasons:

- all three parsers output parses in the standard Penn Treebank notation, making conversion to our chart representation the same process for all; and
- all three are provided with Java API functionality making incorporation into one system more straightforward.

This latter advantage also reduces computation times by enabling just a single parser initialisation step before parsing all sentences.

3.3 Chart Representation

Once each sentence has been parsed, the resulting Penn Treebank parse strings are converted into charts. In the standard approach, a chart is a collection of vertices that span sequences of one or more words of the input, with pairs of vertices that are connected by grammatically labeled edges; where a sequence of words is amenable to more than one analysis, each analysis is represented by a separate edge. Subsequent decisions, or some other choice mechanism, then determine which of the multiple analyses should be chosen.

In the implementation used here, vertices are defined in terms of character positions, while edges are defined by the start and end positions and given a grammatical label. As a step towards efficient implementation, each edge contributed by a given parser also indicates the constituent edges—the grammatical children—contained within the span of that edge. By example, consider the sentence *the cat sat*, which is analysed as follows:

> (S (NP (DT the) (NN cat)) (VP (VBD sat)))

and is spanned thus:

```
 t h e   c a t   s a t
0 1 2 3 4 5 6 7 8 9 10 11
```

This analysis would be represented as a chart consisting of edges:

```
(0, 12, S, {(0,7,NP), (8,11,VP)}}
(0, 7, NP, {(0,3,DT), (4,7,NN)})
(0, 3, DT, {(0,3,the)})
(4, 7, NN, {(4,7,cat)})
...
```

3.4 Chart Voting

The fundamental difference between our approach and those of Henderson and Brill (1999) and Sagae and Lavie (2006) described earlier is in the strategy used when selecting constituents. Previous approaches have considered constituents in isolation: Sagae and Lavie's charts contain all possible constituents, each assigned a weight based on their presence across individual parsers, and these are merely used to inform a second stage, bottom-up reparsing. By comparison, our system could be described as a single-stage, top-down process which operates across the prior parses. Similar to Sagae and Lavie, we employ simple voting to determine the choice of constituents, but we consider only the nominated children of each already-decided constituent. Since each such set of children corresponds to a valid parse, we can ensure there will not be any crossing brackets, and that the resulting parse will be grammatically sound.

Each grammatical constituent is defined by an edge within the chart. For each edge that covers the same span of words[1] across the individual parser outputs, the set of potential analyses can be retrieved. Each such solution provides a potential constituent analysis with which to continue the parse, and for whom votes can be tallied. So, in a democratic manner, any child nominated by all contributing parsers is unanimously voted into the parse. Similarly, any constituents that obtain a majority vote also succeed. In the case of a tie, we resort to arbitrarily choosing between the potential solutions. The only restriction is that children are chosen so that the entire span is accounted for and a complete tree is created.

[1] Matching edges are defined by the tuple `<start pos, end pos, label>`.

Of course pure democracy, at least in the case of parser combination, is quite naïve. It treats all candidates as equal and does not take past performance of parsers into consideration; nor does it take into account the possibility that some parsers may perform better in specific situations. Clearly a sensible step forward here is to move towards a more meritocratic approach, as discussed in Section 5 below.

3.5 An Example

As introduced earlier, Figure 1 illustrates how three different parsers can construct parses that differ or are similar in different ways. In this section, we walk through the combination of these parses to provide an example of how our approach works. To save space and to aid clarity, we represent the span of any node simply by listing the words contained in that span.

In our example, we begin part-way through the parse, where the the current node of interest is the VP which spans from *lock* to *futures*. The analyses of this node are retrieved from the charts delivered by the three parsers, and votes are calculated across the children:

VBP ('lock')	3 votes
PRT ('in')	2 votes
NP ('profits')	2 votes
PP ('by ... futures')	2 votes
PP ('in ... futures')	1 vote

Three votes represents a unanimous decision, while two is a majority; so, the decomposition of the VP node that is common to the second and third analyses is chosen.

Note that the PRT daughter of the VP node receives two votes in total, and subsequently so in turn does it's daughter, the RP. However, though the NP node also received just two votes, the NNS node at the next level of analysis receives three votes. This is because in the case of one of these analyses the NP node is buried deeper in the tree. Similarly, though the PP *in buying futures* was voted twice for its position in the tree, it can be found in all three parses at some level.

However, we have a disagreement as to the decomposition of the PP:

IN ('by')	3 votes
S ('buying futures')	2 votes
NP ('buying futures')	1 vote

Parser	P	R	F
Stanford	87.0	85.7	86.4
OpenNLP	88.1	87.7	87.9
Collins	72.9	88.9	80.1
Combined	90.7	89.5	90.1

Table 1: **P**recision, **R**ecall and **F**-score for individual parsers and their combination; sentence length $<=$ 40 words ($n = 2245$).

Parser	P	R	F
Stanford	86.4	85.0	85.7
OpenNLP	87.4	87.0	87.2
Collins	72.7	88.3	79.7
Combined	90.2	88.9	89.5

Table 2: **P**recision, **R**ecall and **F**-score for individual parsers and their combination; all sentences ($n = 2416$).

Consequently, the chosen analysis of the PP is that proposed in the first and third trees.

3.6 Evaluation

We evaluate the results of our approach using the PARSEVAL standard Evalb (Sekine et al., 2006). The input to the system is Section 23 of the Wall Street Journal (WSJ). All sentences are pre-tokenised to ensure standard input, though each parser executes its own part-of-speech tagging. The system outputs four sets of parse strings: one for each of the three constituent parsers, and one for the final combined result. The sets of parses are compared against the gold standard.

4 Results

We report the bracketing precision, recall and F-score for sentences of length less than 40 words in Table 1, and for all sentences in Table 2.

It is clear that the combined system performs the best. Considering all sentences, we have achieved an error reduction of 22% for precision and 5% for recall, along with an absolute F-score increase of 2.3% over the best single contributor. In order to compare our results with those of previous studies, we reproduce the results of Henderson and Brill (1999) and Sagae and Lavie (2006) alongside our own in Table 3. Our results are comparable directly with those of Henderson and Brill; Sagae and Lavie's scores are a compilation of their best scores across three separate systems tuned to maximise each dimension, hence the high increase in precision and recall.

As a further investigation, we employed a simple measure of confidence in a parse as a function of the number of parsers in the system, the total number of edges in the final chart and the total number of votes cast over just those successful edges:

$$\text{confidence} = \frac{\sum \text{votes}}{\sum \text{edges} \times \sum \text{parsers}}$$

Confidence will be highest if all the parsers agreed on each edge (had the same parse throughout) and will be lower the less they agree. Average confidence across our output is 0.88, which suggests that overall there was a high degree of agreement across parsers. The confidence measure also shows a significant correlation ($p < .001$) with the precision and recall scores across all sentences. This suggests that the system is most likely to be wrong when it is least confident in its output, and so the confidence metric is a good one.

5 Discussion

The performance values reported in Tables 1 and 2 show that the combined system produces more accurate results than the original individual parsers, as we had hoped. By simply taking a majority vote on constituents, our system results in more correct constituent analyses than those proposed by the individual parsers. However, the combined result is not a huge improvement over the highest performing of its contributing proposals.

It is of course possible that for the most part, all parsers get the same things wrong — the rare and infrequent syntactic constructions. This would present a simple voting system with no way to select the correct analysis. However, it is likely that systems that get the same things wrong do so in the same way. Such agreement on incorrectness still represents an agreement, which would provide a high level of confidence in the incorrect choice. However, looking at our confidence scores, this incorrect agreement does not appear to be the case: errors appear to follow a lack of confidence — where there is most disagreement.

The biggest weakness in our approach lies in the arbitrary decision-making procedure used in breaking tied situations. In such tied situations, if the wrong result is chosen, then all the constituent analyses below that point have a high likelihood of being incorrect. This is a particular weakness of our top-down approach, in contrast to Sagae and Lavie's bottom-up method.

There are a variety of ways in which the basic model developed here could be extended. Of course, one could extend the mechanism beyond the three parsers that we use to incorporate a larger number of parsers. However, a more interesting direction is to improve the voting mechanism. The greatest number of errors appears to stem from situations of low agreement, when voting is tied.[2]

One approach to resolving deadlocked situations such as these might be to employ a lookahead approach. As illustrated in our example in section 3.5 upon voting across the top level VP, the NP receives two votes. However, this is only because it was directly under the VP in two cases; the NP was in fact still present in the third analysis, but buried further down in the tree. In a tied situation, this fact would have argued for one analysis over the other.

Another approach is to observe that, while democracy is fair other things being equal, parsers are more akin to experts to be consulted. For example, we might think of each parser as having particular areas of expertise, in the sense that its performance on some kinds of constituents might be better than others. If a given parser has a track record of performing well in the analysis of particular kinds of constituents or substructures, then that parser's vote should carry more weight.

There are a number of approaches we might take to developing a more meritocratic decision procedure.

Track Record on This Parse: This is a general if superficial measure of performance. It assumes no external knowledge, and all parsers begin with an equal weighting. Weightings are increased automatically for every successful vote that a parser casts. If all parsers always agree, weights will remain equal. The more others in

[2] Note that these situations are even more likely to occur if the system were to employ an even number of parsers.

	P	*% decrease*	R	*% decrease*	F	*% increase*
Henderson and Brill, best individual	89.6		89.7		89.7	
Henderson and Brill, combination	92.4	*26.9*	90.1	*3.9*	91.3	*1.6*
Sagae and Lavie, best individual	91.3		90.6		91.0	
Sagae and Lavie, combination	95.1	*43.7*	91.9	*13.8*	92.1	*1.1*
Nowson and Dale, best individual	87.5		88.3		87.2	
Nowson and Dale, combination	90.2	*21.6*	88.9	*5.1*	89.5	*2.3*

Table 3: **P**recision, **R**ecall and **F**-score for parsers; The best individual parser from each study, plus the best combined results, and the differences between them.

the system agree with a parser, the more popular it becomes, and the heavier its weighting. The downside, however, is that should one system perform well early on, its weighting may be so much that a local maxima may be reached.

Previous Track Record: This is also a general measure of performance, but is static and relies on external information. Weightings are set based on the prior performance of a parser: those that have previously produced most accurate results will be trusted more and weighted higher. One source for this data would be previously published, preferably comparable, results. However, as we noted at the start of the paper, good performance in one domain or genre does not guarantee similar results in another.

Two other measures, as suggested by Henderson and Brill (1999), take context into account:

Constituent-Level Track Record: The previous approach gives higher weighting to the parsers that have previously performed best overall, but this does not mean they were the best at everything. In this approach, we narrow the focus to performance over individual constituent types: higher weighting is given to a parser's vote, if upon prior evaluation it has proven successful at selecting the specific nominated constituent. The prerequisite to this is that performance analysis must have been carried out at the level of individual constituents. Alternatives might include using machine learning techniques to automatically determine which parsers do best in which situations.

Structural-Level Track Record: The approach above could be further extended to take account of a larger amount of syntactic context; for example, it might be the case that some parers are better at subject NPs but less good at object NPs. Here we would need to compute weights based on past performance on correct annotation of subtrees in an analysis; clearly this could be done at varying levels of granularity, modulo the problem of sparse data.

6 Conclusion

This paper reports work concerned with combining parsers using a chart based representation and voting scheme. It has introduced the methodology we will employ in our future parsing work: the outputs from multiple parsers are transformed into a chart representation; by voting over children these charts are combined into a single chart combining those constituents for which there is the strongest evidence.

The combination process pursued here is based on the simplest interpretation of evidence, where we pursue a purely democratic approach. This approach is most obviously deficient when we have to deal with ties. Nonetheless, the resulting parses prove more accurate than the single nominees that contributed to their creation, and performance compares well to previous studies that employ more complex and sophisticated methods. This suggests our approach has considerable scope for subsequent improvement, some possible directions for which we have outlined in the latter part of this paper.

Acknowledgments

The authors acknowledge the financial support of the Capital Markets Cooperative Research Centre.

References

Jason Baldridge, Tom Morton, and Gann Bierner 2003. *OpenNLP toolkit*. Available from http://opennlp.sourceforge.net/

Daniel M. Bikel. 2004. Distributional Analysis of a Lexicalized Statistical Parsing Model. *Procedings of EMNLP 2004*, NJ.

Jennifer Chu-Carroll, Krzysztof Czuba, John Prager and Abraham Ittycheriah 2003. In Question Answering, Two Heads Are Better Than One. *Proceedings of HLT-NAACL 2003*, 24–31.

Andrew B. Clegg and Adrian J. Shepherd. 2005. Evaluating and integrating treebank parsers on a biomedical corpus. *Proceedings of the 43rd Meeting of the Association for Computational Linguistics*.

Michael Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. PhD Dissertation, University of Pennsylvania.

J. Early. 1986. An efficient context-free parsing algorithm. In B. J. Grosz, K. Sparck-Jones & B. L. Webber (Eds), *Readings in natural language processing*, 25-33. Morgan Kaufmann, San Francisco, CA.

Hans van Halteren, Jakub Zavrel and Walter Daelemans. 1998 Improving data driven wordclass tagging by system combination. *Proceedings of the 17th International Conference on Computational Linguistics*, 491–497.

John C. Henderson and Eric Brill. 1999. Exploiting Diversity in Natural Language Processing: Combining Parsers. *Proceedings of EMNLP 1999*, 187–194.

John C. Henderson and Eric Brill. 2000. Bagging and Boosting a Treebank Parser. *Proceedings of NAACL 2000*, 34–41.

M. Kay. 1986. Algorithm schemata and data structures in syntactic processing. In B. J. Grosz, K. Sparck-Jones & B. L. Webber (Eds), *Readings in natural language processing*, 35-70. Morgan Kaufmann, San Francisco, CA.

Dan Klein and Christopher D. Manning. 2003. Accurate Unlexicalized Parsing. *Proceedings of the 41st Meeting of the Association for Computational Linguistics*.

Ted Pedersen. 2000. A Simple Approach to Building Ensembles of Naive Bayesian Classifiers for Word Sense Disambigusation. *Proceedings of NAACL 2000*, 63–69.

Roi Reichart and Ari Rappoport. 2007. Self-Training for Enhancement and Domain Adaptation of Statistical Parsers Trained on Small Datasets. *Proceedings of the 45th Meeting of the Association for Computational Linguistics*, 616–623.

Roi Reichart and Ari Rappoport. 2007. An Ensemble Method for Selection of High Quality Parses. *Proceedings of the 45th Meeting of the Association for Computational Linguistics*, 408–415.

Kenji Sagae and Alon Lavie. 2006. Parser Combination by Reparsing. *Proceedings of HLT-NAACL 2006*, 129–133.

Satoshi Sekine, Michael J. Collins, David Brooks and David Ellis 2006. *Evalb*. Available from http://nlp.cs.nyu.edu/evalb/

Daniel Zeman and Zdeněk Žabokrtský 2006. Improving Parsing Accuracy by Combining Diverse Dependency Parser. *Proceedings of the Ninth International Workshop on Parsing Technology*, 171–178.