Syntactic Parse Fusion

Do Kook Choe
Brown University
Providence, RI
dc65@cs.brown.edu

David McClosky
IBM Research
Yorktown Heights, NY
dmcclosky@us.ibm.com

Eugene Charniak
Brown University
Providence, RI
ec@cs.brown.edu

Abstract

Model combination techniques have consistently shown state-of-the-art performance across multiple tasks, including syntactic parsing. However, they dramatically increase runtime and can be difficult to employ in practice. We demonstrate that applying constituency model combination techniques to n-best lists instead of n different parsers results in significant parsing accuracy improvements. Parses are weighted by their probabilities and combined using an adapted version of Sagae and Lavie (2006). These accuracy gains come with marginal computational costs and are obtained on top of existing parsing techniques such as discriminative reranking and self-training, resulting in state-of-the-art accuracy: 92.6% on WSJ section 23. On out-of-domain corpora, accuracy is improved by 0.4% on average. We empirically confirm that six well-known n-best parsers benefit from the proposed methods across six domains.

1 Introduction

Researchers have proposed many algorithms to combine parses from multiple parsers into one final parse (Henderson and Brill, 1999; Zeman and Žabokrtskỳ, 2005; Sagae and Lavie, 2006; Nowson and Dale, 2007; Fossum and Knight, 2009; Petrov, 2010; Johnson and Ural, 2010; Huang et al., 2010; McDonald and Nivre, 2011; Shindo et al., 2012; Narayan and Cohen, 2015). These new parses are substantially better than the originals: Zhang et al. (2009) combine outputs from multiple n-best parsers and achieve an F_1 of 92.6% on the WSJ test set, a 0.5% improvement over their best n-best parser. Model combination approaches tend to fall into the following categories: *hybridization*, where multiple parses are combined into a single parse; *switching*, which picks a single parse according to some criteria (usually a form of voting); *grammar merging* where grammars are combined before or during parsing; and *stacking*, where one parser sends its prediction to another at runtime. All of these have at least one of the caveats that (1) overall computation is increased and runtime is determined by the slowest parser and (2) using multiple parsers increases the system complexity, making it more difficult to deploy in practice. In this paper, we describe a simple hybridization extension ("fusion") which obtains much of hybridization's benefits while using only a single n-best parser and minimal extra computation. Our method treats each parse in a single parser's n-best list as a parse from n separate parsers. We then adapt parse combination methods by Henderson and Brill (1999), Sagae and Lavie (2006), and Fossum and Knight (2009) to fuse the constituents from the n parses into a single tree. We empirically show that six n-best parsers benefit from parse fusion across six domains, obtaining state-of-the-art results. These improvements are complementary to other techniques such as reranking and self-training. Our best system obtains an F_1 of 92.6% on WSJ section 23, a score previously obtained only by combining the outputs from multiple parsers. A reference implementation is available as part of BLLIP Parser at `http://github.com/BLLIP/bllip-parser/`

2 Fusion

Henderson and Brill (1999) propose a method to combine trees from m parsers in three steps: populate a chart with constituents along with the number of times they appear in the trees; remove any constituent with count less than $m/2$ from the chart; and finally create a final tree with all the remaining constituents. Intuitively their method

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1360–1366,
Lisbon, Portugal, 17-21 September 2015.

constructs a tree with constituents from the majority of the trees, which boosts precision significantly. Henderson and Brill (1999) show that this process is guaranteed to produce a valid tree. Sagae and Lavie (2006) generalize this work by reparsing the chart populated with constituents whose counts are above a certain threshold. By adjusting the threshold on development data, their generalized method balances precision and recall. Fossum and Knight (2009) further extend this line of work by using n-best lists from multiple parsers and combining productions in addition to constituents. Their model assigns sums of joint probabilities of constituents and parsers to constituents. Surprisingly, exploiting n-best trees does not lead to large improvement over combining 1-best trees in their experiments.

Our extension takes the n-best trees from a parser as if they are 1-best parses from n parsers, then follows Sagae and Lavie (2006). Parses are weighted by the estimated probabilities from the parser. Given n trees and their weights, the model computes a constituent's weight by summing weights of all trees containing that constituent. Concretely, the weight of a constituent spanning from ith word to jth word with label ℓ is

$$c_\ell(i \to j) = \sum_{k=1}^{n} W(k) C_\ell^k(i \to j) \qquad (1)$$

where $W(k)$ is the weight of kth tree and $C_\ell^k(i \to j)$ is one if a constituent with label ℓ spanning from i to j is in kth tree, zero otherwise. After populating the chart with constituents and their weights, it throws out constituents with weights below a set threshold t. Using the threshold $t = 0.5$ emulates the method of Henderson and Brill (1999) in that it constructs the tree with the constituents in the majority of the trees. The CYK parsing algorithm is applied to the chart to produce the final tree.

Note that populating the chart is linear in the number of words and the chart contains substantially fewer constituents than charts in well-known parsers, making this a fast procedure.

2.1 Score distribution over trees

We assume that n-best parsers provide trees along with some kind of scores (often probabilities or log probabilities). Given these scores, a natural way to obtain weights is to normalize the probabilities. However, parsers do not always provide accurate estimates of parse quality. We may obtain better performance from parse fusion by altering this distribution and passing scores through a nonlinear function, $f(\cdot)$. The kth parse is weighted:

$$W(k) = \frac{f(\text{SCORE}(k))}{\sum_{i=1}^{n} f(\text{SCORE}(i))} \qquad (2)$$

where $\text{SCORE}(i)$ is the score of ith tree.[1] We explore the family of functions $f(x) = x^\beta$ which can smooth or sharpen the score distributions. This includes a tunable parameter, $\beta \in \mathbb{R}_0^+$:

$$W(k) = \frac{\text{SCORE}(k)^\beta}{\sum_{i=1}^{n} \text{SCORE}(i)^\beta} \qquad (3)$$

Employing $\beta < 1$ flattens the score distribution over n-best trees and helps over-confident parsers. On the other hand, having $\beta > 1$ skews the distribution toward parses with higher scores and helps under-confident parsers. Note that setting $\beta = 0$ weights all parses equally and results in majority voting at the constituent level. We leave developing other nonlinear functions for fusion as future work.

3 Experiments

Corpora: Parse fusion is evaluated on British National Corpus (BNC), Brown, GENIA, Question Bank (QB), Switchboard (SB) and Wall Street Journal (WSJ) (Foster and van Genabith, 2008; Francis and Kučera, 1989; Kim et al., 2003; Judge et al., 2006; Godfrey et al., 1992; Marcus et al., 1993). WSJ is used to evaluate in-domain parsing, the remaining five are used for out-of-domain. For divisions, we use tune and test splits from Bacchiani et al. (2006) for Brown, McClosky's test PMIDs[2] for GENIA, Stanford's test splits[3] for QuestionBank, and articles 4000–4153 for Switchboard.

Parsers: The methods are applied to six widely used n-best parsers: Charniak (2000), Stanford (Klein and Manning, 2003), BLLIP (Charniak and Johnson, 2005), Self-trained BLLIP (McClosky et al., 2006)[4], Berkeley (Petrov et al., 2006), and Stanford RNN (Socher et al., 2013). The list of parsers and their accuracies on the WSJ test set is reported in Table 1. We convert to Stanford

[1] For parsers that return log probabilities, we turn these into probabilities first.

[2] `http://nlp.stanford.edu/~mcclosky/biomedical.html`

[3] `http://nlp.stanford.edu/data/QuestionBank-Stanford.shtml`

[4] Using the '`WSJ+Gigaword-v2`' BLLIP model.

Parser	F_1	UAS	LAS
Stanford	85.4	90.0	87.3
Stanford RNN[5]	89.6	92.9	90.4
Berkeley	90.0	93.5	91.2
Charniak	89.7	93.2	90.8
BLLIP	91.5	94.4	92.0
Self-trained BLLIP	92.2	94.7	92.2

Table 1: Six parsers along with their 1-best F_1 scores, unlabeled attachment scores (UAS) and labeled attachment scores (LAS) on WSJ section 23.

Dependencies (basic dependencies, version 3.3.0) and provide dependency metrics (UAS, LAS) as well.

Supervised parsers are trained on the WSJ training set (sections 2–21) and use section 22 or 24 for development. Self-trained BLLIP is self-trained using two million sentences from Gigaword and Stanford RNN uses word embeddings trained from larger corpora.

Parameter tuning: There are three parameters for our fusion process: the size of the n-best list ($2 < n \leq 50$), the smoothing exponent from Section 2.1 ($\beta \in [0.5, 1.5]$ with 0.1 increments), and the minimum threshold for constituents ($t \in [0.2, 0.7]$ with 0.01 increments). We use grid search to tune these parameters for two separate scenarios. When parsing WSJ (in-domain), we tune parameters on WSJ section 24. For the remaining corpora (out-of-domain), we use the tuning section from Brown. Each parser is tuned separately, resulting in 12 different tuning scenarios. In practice, though, in-domain and out-of-domain tuning regimes tend to pick similar settings within a parser. Across parsers, settings are also fairly similar (n is usually 30 or 40, t is usually between 0.45 and 0.5). While the smoothing exponent varies from 0.5 to 1.3, setting $\beta = 1$ does not significantly hurt accuracy for most parsers.

To study the effects of these parameters, Figure 1 shows three slices of the tuning surface for BLLIP parser on WSJ section 24 around the optimal settings ($n = 30, \beta = 1.1, t = 0.47$). In each graph, one of the parameters is varied while the other is held constant. Increasing n-best size improves accuracy until about $n = 30$ where there seems to be sufficient diversity. For BLLIP, the smoothing exponent (β) is best set around 1.0, with accuracy falling off if the value deviates too much. Finally, the threshold parameter is empirically optimized a little below $t = 0.5$ (the value suggested by Henderson and Brill (1999)). Since score values are normalized, this means that constituents need roughly half the "score mass" in order to be included in the chart. Varying the threshold changes the precision/recall balance since a high threshold adds only the most confident constituents to the chart (Sagae and Lavie, 2006).

[5]Socher et al. (2013) report an F_1 of 90.4%, but this is the result of using an ensemble of two RNNs (p.c.). We use a single RNN in this work.

Parser	WSJ	Brown
BLLIP	90.6	85.7
+ Fusion	91.0	86.0
+ Majority voting ($\beta = 0$)	89.1	83.8
+ Rank-based weighting	89.3	84.1

Table 2: F_1 of a baseline parser, fusion, and baselines on development sections of corpora (WSJ section 24 and Brown tune).

Baselines: Table 2 gives the accuracy of fusion and baselines for BLLIP on the development corpora. Majority voting sets $n = 50, \beta = 0, t = 0.5$ giving all parses equal weight and results in constituent-level majority voting. We explore a rank-based weighting which ignores parse probabilities and weight parses only using the rank: $W_{\text{rank}}(k) = 1/(2^k)$. These show that accurate parse-level scores are critical for good performance.

Final evaluation: Table 3 gives our final results for all parsers across all domains. Results in blue are significant at $p < 0.01$ using a randomized permutation test. Fusion generally improves F_1 for in-domain and out-of-domain parsing by a significant margin. For the self-trained BLLIP parser, in-domain F_1 increases by 0.4% and out-of-domain F_1 increases by 0.4% on average. Berkeley parser obtains the smallest gains from fusion since Berkeley's n-best lists are ordered by factors other than probabilities. As a result, the probabilities from Berkeley can mislead the fusion process.

We also compare against model combination using our reimplementation of Sagae and Lavie (2006). For these results, all six parsers were given equal weight. The threshold was set to 0.42 to optimize model combination F_1 on development data (similar to Setting 2 for constituency parsing in Sagae and Lavie (2006)). Model combination

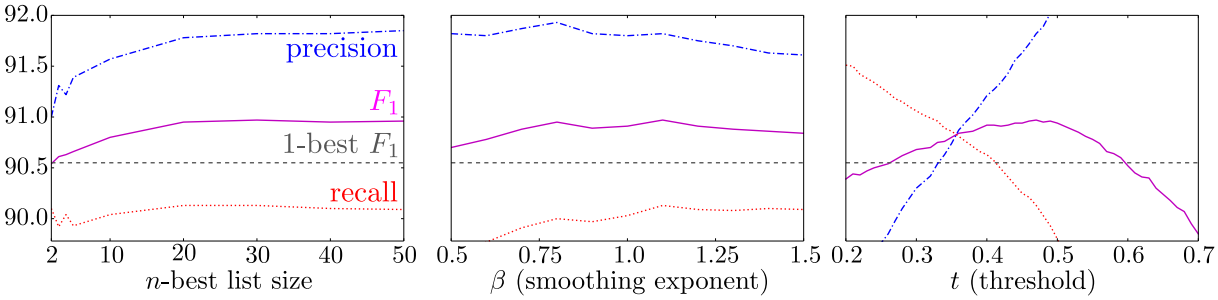

Figure 1: Tuning parameters independently for BLLIP and their impact on F_1 for WSJ section 24 (solid purple line). For each graph, non-tuned parameters were set at the optimal configuration for BLLIP ($n = 30$, $\beta = 1.1$, $t = 0.47$). The dashed grey line represents the 1-best baseline at 90.6% F_1.

Parser	BNC	Brown	GENIA	SB	QB	WSJ
Stanford	78.4 / 79.6	80.7 / 81.6	73.1 / 73.9	67.0 / 67.9	78.6 / 80.0	85.4 / 86.2
Stanford RNN	82.0 / 82.3	84.0 / 84.3	76.0 / 76.2	70.7 / 71.2	82.9 / 83.6	89.6 / 89.7
Berkeley	82.3 / 82.9	84.6 / 84.6	76.4 / 76.6	74.5 / 75.1	86.5 / 85.9	90.0 / 90.3
Charniak	82.5 / 83.0	83.9 / 84.6	74.8 / 75.7	76.8 / 77.6	85.6 / 86.3	89.7 / 90.1
BLLIP	84.1 / 84.7	85.8 / 86.0	76.7 / 77.1	79.2 / 79.5	88.1 / 88.9	91.5 / 91.7
Self-trained BLLIP	85.2 / 85.8	87.4 / 87.7	77.8 / 78.2	80.9 / 81.7	89.5 / 89.5	92.2 / 92.6
Model combination	86.6	87.7	79.4	80.9	89.3	92.5

Table 3: Evaluation of the constituency fusion method on six parsers across six domains. x/y indicates the F_1 from the baseline parser (x) and the baseline parser with fusion (y) respectively. Blue indicates a statistically significant difference between fusion and its baseline parser ($p < 0.01$).

performs better than fusion on BNC and GENIA, but surprisingly fusion outperforms model combination on three of the six domains (not usually not by a significant margin). With further tuning (e.g., specific weights for each constituent-parser pair), the benefits from model combination should increase.

Multilingual evaluation: We evaluate fusion with the Berkeley parser on Arabic (Maamouri et al., 2004; Green and Manning, 2010), French (Abeillé et al., 2003), and German (Brants et al., 2002) from the SPMRL 2014 shared task (Seddah et al., 2014) but did not observe any improvement. We suspect this has to do with the same ranking issues seen in the Berkeley Parser's English results. On the other hand, fusion helps the parser of Narayan and Cohen (2015) on the German NEGRA treebank (Skut et al., 1997) to improve from 80.9% to 82.4%.

Runtime: As discussed in Section 2, fusion's runtime overhead is minimal. Reranking parsers (e.g., BLLIP and Stanford RNN) already need to perform n-best decoding as input for the reranker. Using a somewhat optimized implementation fusion in C++, the overhead over BLLIP parser is less than 1%.

Discussion: Why does fusion help? It is possible that a parser's n-list and its scores act as a weak approximation to the full parse forest. As a result, fusion seems to provide part of the benefits seen in forest reranking (Huang, 2008).

Results from Fossum and Knight (2009) imply that fusion and model combination might not be complementary. Both n-best lists and additional parsers provide syntactic diversity. While additional parsers provide greater diversity, n-best lists from common parsers are varied enough to provide improvements for parse hybridization.

We analyzed how often fusion produces completely novel trees. For BLLIP on WSJ section 24, this only happens about 11% of the time. Fusion picks the 1-best tree 72% of the time. This means that for the remaining 17%, fusion picks an existing parse from the rest of the n-list, acting similar to a reranker. When fusion creates unique trees, they are significantly better than the original 1-best trees (for the 11% subset of WSJ 24, F_1 scores are 85.5% with fusion and 84.1% without, $p < 0.003$). This contrasts with McClosky et al. (2012) where novel predic-

tions from model combination (stacking) were worse than baseline performance. The difference is that novel predictions with fusion better incorporate model confidence whereas when stacking, a novel prediction is less trusted than those produced by one or both of the base parsers.

Preliminary extensions: Here, we summarize two extensions to fusion which have yet to show benefits. The first extension explores applying fusion to dependency parsing. We explored two ways to apply fusion when starting from constituency parses: (1) fuse constituents and then convert them to dependencies and (2) convert to dependencies then fuse the dependencies as in Sagae and Lavie (2006). Approach (1) does not provide any benefit (LAS drops between 0.5% and 2.4%). This may result from fusion's artifacts including unusual unary chains or nodes with a large number of children — it is possible that adjusting unary handling and the precision/recall tradeoff may reduce these issues. Approach (2) provided only modest benefits compared to those from constituency parsing fusion. The largest LAS increase for (2) is 0.6% for the Stanford Parser, though for Berkeley and Self-trained BLLIP, dependency fusion results in small losses (-0.1% LAS). Two possible reasons are that the dependency baseline is higher than its constituency counterpart and some dependency graphs from the n-best list are duplicates which lowers diversity and may need special handling, but this remains an open question.

While fusion helps on top of a self-trained parser, we also explored whether a fused parser can self-train (McClosky et al., 2006). To test this, we (1) parsed two million sentences with BLLIP (trained on WSJ), (2) fused those parses, (3) added the fused parses to the gold training set, and (4) retrained the parser on the expanded training. The resulting model did not perform better than a self-trained parsing model that didn't use fusion.

4 Conclusions

We presented a simple extension to parse hybridization which adapts model combination techniques to operate over a single parser's n-best list instead of across multiple parsers. By weighting each parse by its probability from the n-best parser, we are able to better capture the confidence at the constituent level. Our best configuration obtains state-of-the-art accuracy on WSJ with an F_1 of 92.6%. This is similar to the accuracy obtained from actual model combination techniques but at a fraction of the computational cost. Additionally, improvements are not limited to a single parser or domain. Fusion improves parser accuracy for six n-best parsers both in-domain and out-of-domain.

Future work includes applying fusion to n-best dependency parsers and additional (parser, language) pairs. We also intend to explore how to better apply fusion to converted dependencies from constituency parsers. Lastly, it would be interesting to adapt fusion to other structured prediction tasks where n-best lists are available.

Acknowledgements

We are grateful to Shay Cohen and Shashi Narayan who gave us early access to their parser's German results. We would also like to thank Mohit Bansal, Yoav Goldberg, Siddharth Patwardhan, and Kapil Thadani for helpful discussions and our anonymous reviewers for their insightful comments and suggestions.

References

Anne Abeillé, Lionel Clément, and François Toussenel. 2003. Building a treebank for french. In *Treebanks*, pages 165–187. Springer.

Michiel Bacchiani, Michael Riley, Brian Roark, and Richard Sproat. 2006. Map adaptation of stochastic grammars. *Computer speech & language*, 20(1):41–68.

Sabine Brants, Stefanie Dipper, Silvia Hansen, Wolfgang Lezius, and George Smith. 2002. The tiger treebank. In *Proceedings of the workshop on treebanks and linguistic theories*, volume 168.

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine n-best parsing and MaxEnt discriminative reranking. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 173–180, Ann Arbor, Michigan, June. Association for Computational Linguistics.

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference*, pages 132–139. Association for Computational Linguistics.

Victoria Fossum and Kevin Knight. 2009. Combining constituent parsers. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers*, pages 253–256, Boulder, Colorado, June. Association for Computational Linguistics.

Jennifer Foster and Josef van Genabith. 2008. Parser evaluation and the BNC: Evaluating 4 constituency parsers with 3 metrics. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*. European Language Resources Association (ELRA).

Winthrop Nelson Francis and Henry Kučera. 1989. *Manual of information to accompany a standard corpus of present-day edited American English, for use with digital computers*. Brown University, Department of Linguistics.

John J. Godfrey, Edward C. Holliman, and Jane McDaniel. 1992. Switchboard: Telephone speech corpus for research and development. In *Acoustics, Speech, and Signal Processing, 1992. ICASSP-92., 1992 IEEE International Conference on*, volume 1, pages 517–520. IEEE.

Spence Green and Christopher D Manning. 2010. Better arabic parsing: Baselines, evaluations, and analysis. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 394–402. Association for Computational Linguistics.

John C. Henderson and Eric Brill. 1999. Exploiting diversity in natural language processing: Combining parsers. In *Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, pages 187–194, College Park, MD, June. Association for Computational Linguistics.

Zhongqiang Huang, Mary Harper, and Slav Petrov. 2010. Self-training with products of latent variable grammars. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 12–22. Association for Computational Linguistics.

Liang Huang. 2008. Forest reranking: Discriminative parsing with non-local features. In *Proceedings of ACL-08: HLT*, pages 586–594, Columbus, Ohio, June. Association for Computational Linguistics.

Mark Johnson and Ahmet Engin Ural. 2010. Reranking the berkeley and brown parsers. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 665–668. Association for Computational Linguistics.

John Judge, Aoife Cahill, and Josef van Genabith. 2006. Questionbank: Creating a corpus of parse-annotated questions. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 497–504, Sydney, Australia, July. Association for Computational Linguistics.

J-D Kim, Tomoko Ohta, Yuka Tateisi, and Junichi Tsujii. 2003. Genia corpus—a semantically annotated corpus for bio-textmining. *Bioinformatics*, 19(suppl 1):i180–i182.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 423–430, Sapporo, Japan, July. Association for Computational Linguistics.

Mohamed Maamouri, Ann Bies, Tim Buckwalter, and Wigdan Mekki. 2004. The penn arabic treebank: Building a large-scale annotated arabic corpus. In *NEMLAR conference on Arabic language resources and tools*, pages 102–109.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: The penn treebank. *Computational linguistics*, 19(2):313–330.

David McClosky, Eugene Charniak, and Mark Johnson. 2006. Effective self-training for parsing. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 152–159, New York City, USA, June. Association for Computational Linguistics.

David McClosky, Sebastian Riedel, Mihai Surdeanu, Andrew McCallum, and Christopher D. Manning. 2012. Combining joint models for biomedical event extraction. *BMC Bioinformatics*, 13(Suppl 11):S9.

Ryan McDonald and Joakim Nivre. 2011. Analyzing and integrating dependency parsers. *Computational Linguistics*, 37(1):197–230.

Shashi Narayan and Shay B. Cohen. 2015. Diversity in spectral learning for natural language parsing. In *Proceedings of EMNLP*.

Scott Nowson and Robert Dale. 2007. Charting democracy across parsers. In *Proceedings of the Australasian Language Technology Workshop*, pages 75–82.

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 433–440, Sydney, Australia, July. Association for Computational Linguistics.

Slav Petrov. 2010. Products of random latent variable grammars. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 19–27. Association for Computational Linguistics.

Kenji Sagae and Alon Lavie. 2006. Parser combination by reparsing. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*, pages 129–132, New York City, USA, June. Association for Computational Linguistics.

Djamé Seddah, Sandra Kübler, and Reut Tsarfaty. 2014. Introducing the spmrl 2014 shared task on parsing morphologically-rich languages. In *Proceedings of the First Joint Workshop on Statistical Parsing of Morphologically Rich Languages and Syntactic Analysis of Non-Canonical Languages*, pages 103–109, Dublin, Ireland, August. Dublin City University.

Hiroyuki Shindo, Yusuke Miyao, Akinori Fujino, and Masaaki Nagata. 2012. Bayesian symbol-refined tree substitution grammars for syntactic parsing. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 440–448. Association for Computational Linguistics.

Wojciech Skut, Brigitte Krenn, Thorsten Brants, and Hans Uszkoreit. 1997. An annotation scheme for free word order languages. In *Proceedings of the fifth conference on Applied natural language processing*, pages 88–95. Association for Computational Linguistics.

Richard Socher, John Bauer, Christopher D. Manning, and Ng Andrew Y. 2013. Parsing with compositional vector grammars. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 455–465, Sofia, Bulgaria, August. Association for Computational Linguistics.

Daniel Zeman and Zdeněk Žabokrtskỳ. 2005. Improving parsing accuracy by combining diverse dependency parsers. In *Proceedings of the Ninth International Workshop on Parsing Technology*, pages 171–178. Association for Computational Linguistics.

Hui Zhang, Min Zhang, Chew Lim Tan, and Haizhou Li. 2009. K-best combination of syntactic parsers. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 1552–1560, Singapore, August. Association for Computational Linguistics.