Co-Parsing with Competitive Models

Lidia Khmylko
Natural Language Systems Group
University of Hamburg, Germany
khmylko@informatik.uni-hamburg.de

Kilian A. Foth
smartSpeed GmbH & Co. KG
Hamburg, Germany
kilian.foth@smartspeed.com

Wolfgang Menzel
Natural Language Systems Group
University of Hamburg, Germany
menzel@informatik.uni-hamburg.de

Abstract

We present an asymmetric approach to a run-time combination of two parsers where one component serves as a predictor to the other one. Predictions are integrated by means of weighted constraints and therefore are subject to preferential decisions. Previously, the same architecture has been successfully used with predictors providing partial or inferior information about the parsing problem. It has now been applied to a situation where the predictor produces exactly the same type of information at a fully competitive quality level. Results show that the combined system outperforms its individual components, even though their performance in isolation is already fairly high.

1 Introduction

Machine learning techniques for automatically acquiring processing models from a data collection and traditional methods of eliciting linguistic knowledge from human experts are usually considered as two alternative roadmaps towards natural language processing solutions. Since the resulting components exhibit quite different performance characteristics with respect to coverage, robustness and output quality, they might be able to provide some kind of complementary information, which could even lead to a notable degree of synergy between them when combined within a single system solution.

For the task of dependency parsing, the high potential for such a synergy has indeed been demonstrated already (e.g. Zeman and Žabokrtský (2005), Foth and Menzel (2006)).

A popular approach for combining alternative decision procedures is voting (Zeman and Žabokrtský, 2005). It makes use of a symmetric architecture, where a meta component chooses from among the available candidate hypotheses by means of a (weighted) voting scheme. Such an approach not only requires the target structures of all components to be of the same kind, but in case of complex structures like parse trees also requires sophisticated decision procedures which are able to select the optimal hypotheses with respect to additional global constraints (e.g. the tree property). Since this optimization problem has to be solved by the individual parser anyhow, an asymmetric architecture suggests itself as an alternative.

In asymmetric architectures, a master component, i.e. a full fledged parser, is solely in charge of deciding on the target structure, whilst the others (so called helper or predictor components) provide additional evidence which is integrated into the global decision by suitable means. Such a scheme has been extensively investigated for the Weighted Constraint Dependency Grammar, WCDG (Foth, 2006). External evidence from the predictor components is integrated by means of constraints, which check for compatibility between a local structure and a prediction, and penalize this hypothesis in case of a conflict. So far, however, all the additional information sources which have been considered in this research differed considerably from the master component: They either focused on particular aspects of the parsing problem (e.g. POS tagging, chunking, PP attachment), or used a simplified scheme for structural annotation (e.g. projective instead of non-projective trees).

This paper takes one step further by investigating the same architecture under the additional condition that (1) the helper component provides the

Proceedings of the 11th International Conference on Parsing Technologies (IWPT), pages 99–107,
Paris, October 2009.

very same kind of target structure as the master, and (2) the quality levels of each of the components in isolation are considered.

As a helper component MSTParser (McDonald, 2006), a state-of-the-art dependency parser for non-projective structures based on a discriminative learning paradigm, is considered. The accuracy of MSTParser differs insignificatly from that of WCDG with all the previously used helper components active.

Section two introduces WCDG with a special emphasis on the soft integration of external evidence while section three describes MSTParser which is used as a new predictor component. Since parsing results for these systems have been reported in quite different experimental settings we first evaluate them under comparable conditions and provide the results of using MSTParser as a guiding predictor for WCDG in section four and discuss whether the expected synergies have really materialized. Section five concentrates on a comparative error analysis.

2 WCDG

The formalism of a Constraint Dependency Grammar was first introduced by Maruyama (1990) and suggests modeling natural language with the help of constraints. Schröder (2002) has extended the approach to Weighted Constraint Dependency Grammar, WCDG, where weights are used to further disambiguate between competing structural alternatives. A WCDG models natural language as labeled dependency trees and is entirely declarative. It has no derivation rules — instead, constraints license well-formed tree structures. The reference implementation of WCDG for the German language used for the experiments described below contains about $1,000$ manually compiled constraints.[1]

Every constraint of the WCDG carries a *weight*, also referred to as a *penalty*, in the interval from zero to one, a lower value of the weight reflects its greater importance. Constraints having zero weights are referred to as *hard* and are used for prohibitive rules. Constraints with a weight greater than zero, also called *defeasible*, may express universal principles or vague preferences for language phenomena.

[1]Freely available from `http://nats-www.informatik.uni-hamburg.de/view/CDG/DownloadPage`

Attempts have been made to compute the weights of a WCDG automatically by observing which weight vectors perform best on a given corpus, but the computations did not bring any significant improvements to the manually assigned scored (Schröder et al., 2001). Empirically, the absolute values of defeasible constraints usually do not matter greatly as long as the relative importance of the rules remains preserved so that typical constructions are preferred, but seldom variations are also allowed. Thus, the values of weights of the WCDG constraints have to be determined by the grammar writer experimentally.

If a set of dependency edges in a parse found by the system violates any of the constraints, it is registered as a *constraint violation* between the structure and the rules of the language. The *score* of an analysis is the product of all the weights for constraint violations occurring in the structure. It becomes possible to differentiate between the quality of different parse results: the analysis with a higher score is considered preferable. Although, under these conditions, an analysis having only a few grave conflicts may be preferred by the system against another one with a great number of smaller constraint violations, but it ensures that an analysis which violates any of the hard constraints always receives the lowest possible score.

The parsing problem is being treated in the WCDG system as a Constraint Satisfaction Problem. While a complete search is intractable for such a problem, *transformation-based solution methods* provide a reliable heuristic alternative. Starting with an initial guess about the optimal tree, changes of labels, subordinations, or lexical variants are applied, with constraint violations used as a control mechanism guiding the transformation process (Foth et al., 2000).

A transformation-based search cannot guarantee to find the best solution to the constraint satisfaction problem. Compared to the resource requirements of a complete search, however, it is not only more efficient, but can also be interrupted at any time. Even if interrupted, it will always return an analysis, together with a list of constraint violations it was not able to remove. The algorithm terminates on its own if no violated constraints with a weight above a predefined threshold remain. Alternatively, a timeout condition can be imposed.

The same kind of constraints that describe grammar rules, can also be used as an interface

to external predictor components. Thus, the formalism turned out to be flexible enough to incorporate other sources of knowledge into the decision process on the optimal structural interpretation. Foth and Menzel (2006) have reported about five additional statistical components that have been successfully integrated into WCDG: POS tagger, chunker, supertagger, PP attacher and a shift-reduce oracle parser. They have also shown that the accuracy improves if multiple components interact and consistent predictions no longer can be guaranteed. Even thought previously integrated predictor components have an accuracy that is mostly — with the exception of the tagger — below that of the parser itself, WCDG not only avoids error propagation successfully, it also improves its results consistently with each component added.

3 MSTParser

MSTParser (McDonald, 2006) is a state-of-the-art language independent data-driven parser. It processes the input in two separate stages. In the first, the dependency structure is determined, labeling is applied to it successively in the second. The reasons of its efficiency lie in the successful combination of discriminative learning with graph-based solution methods for the parsing problem.

In this edge-factored graph-based model, each edge of the dependency graph is assigned a real-valued score by its linear model. The score of the graph is defined as the sum of its edge scores.

If a scoring function for edges is known, the parsing problem becomes equivalent to finding the highest scoring directed spanning tree in the complete graph over the given sentence, and the correct parse can be obtained by searching the space of valid dependency graphs for a tree with a maximum score.

This formalism allows to find efficient solutions for both projective and non-projective trees. When only features over single edges are taken into account, the complexity falls to $O(n^2)$ (McDonald et al., 2005).

Not only a single edge, but also adjacent edges may be included into the scoring function. As a result, intractability problems arise for the non-projective algorithm, but an efficient approximate algorithm based on exhaustive search is provided for this case (McDonald et al., 2006). This algorithm was also used for our experiments.[2]

The parsing model of MSTParser has the advantage that it can be trained globally and eventually be applied with an exact inference algorithm. On the other hand, the parser has only limited access to the history of parsing decisions. To avoid complexity problems, the scores (and the feature representations) are restricted to a single edge or adjacent edges. Outsourcing labeling into a separate stage comes at the price of not being able to combine knowledge about the label and the structure it is attached to. Such combined evidence, however, might be helpful for some disambiguation problems.

4 Guiding WCDG by Predictions of MSTParser

MSTParser predictions are integrated into the decision procedure of WCDG by means of two additional constraints, which monitor each dependency hypothesis for being in accord with the prediction and penalize it if a mismatch has been found. One of the constraints checks the attachment point being the same, while the other takes care of the dependency label.

To properly adjust the weights of these constraints, it has to be determined how valuable the information of the predictor is relative to the information already present in the system. This gradation is needed to establish a balance between the influence of the grammar and the predictor. According to the scoring principles of WCDG, a low weight strongly deprecates all deviations from the prediction, thus forcing the system to follow them almost without exception. Higher weights, on the other hand, enable the grammar to override a prediction. This, however, also means that predictions have less guiding effect of the transformation process. Typically for WCDG, the best suitable weights have to be tuned on development data.

To determine the best constraint weights the WCDG grammar has been extended with three additional constraints similar to those used for the shift-reduce predictor in the previous experiments (Foth, 2006). Two of them advise WCDG on the structural information available from the MSTParser result and one fetches the edge label predicted.

As a result of these experiments, the optimum

[2] MSTParser is freely available from `http://sourceforge.net/projects/mstparser`

weight for the attachment predictions has been adjusted to 0.75. Compared to a weight of 0.9 for the shift-reduce parser, this is a rather strong influence, which also reflects the differences in the reliability of these two information sources. With a weight of 0.9, the integration of the label predictions is considerably weaker, which is consistent with their lower degree of accuracy.

Evaluation

The most common general measures for the quality of dependency trees are *structural accuracy* that points out the percentage of words correctly attached to their head word, and *labeled accuracy* which is the ratio of the correctly attached words which also have the correct label. Still, it is difficult to directly compare the results reported for different parsers, as the evaluation results are influenced by the data used during the experiment, the domain of the data, and different annotation guidelines. Moreover, the particular kind of POS information might be relevant, which either can be obtained from the manual annotations or be provided by a real tagger. Even such a condition as the treatment of punctuation has not yet become a standard. Following the evaluation procedure in the CoNLL-X shared task (Buchholz and Marsi, 2006), we will not include punctuation into the performance measures, as was done in previous WCDG experiments (Foth and Menzel, 2006). The source of POS tagging information will need to be specified in each individual case.

All the evaluations were performed on a thousand sentences ($18,602$ – $19,601$) from the NEGRA treebank, the same data set that was previously used in the performance evaluations of WCDG, e.g. in (Foth, 2006). The NEGRA treebank is a collection of newspaper articles; in the original, it stores phrase structure annotations. These have been automatically translated into dependency trees and then manually corrected to bring them in accord with the annotation guidelines of WCDG. The major difference consists in a different treatment of non-projectivity, where WCDG only allows non-projectivity in the attachment of verbal arguments, relative clauses and coordinations, i.e., the cases where it helps to decrease ambiguity. Furthermore, corrections were applied when the annotations of NEGRA itself turned out to be inconsistent (usually in connection with co-ordinated or elliptical structures, adverbs and subclauses).

Unfortunately, these manually corrected data were only available for a small part ($3,000$ sentences) of the NEGRA corpus, which is not sufficient for training MSTParser on WCDG-conforming tree structures. Previous evaluations of the MSTParser have used much larger training sets. E.g., during the CoNLL-X shared task 39,216 sentences from the TIGER Treebank (Brants et al., 2002) were used.

Therefore, we used $20,000$ sentences from the online archive of `www.heise.de` as an alternative training set. They have been manually annotated according to the WCDG guidelines (and are referred to `heiseticker` in the following)[3]. The texts in this corpus are all from roughly the same domain as in NEGRA, and although very many technical terms and proper nouns are used, the sentences have only a slightly longer mean length compared to the NEGRA corpus.

Using POS tags from the gold annotations, MSTParser achieves 90.5% structural and 87.5% labeled accuracy on the aforementioned NEGRA test set (Table 1). Even a model trained on the inconsistent NEGRA data excluding the test set reaches state-of-the-art 90.5 and 87.3% for structural and labeled accuracy respectively, despite the obvious mismatch between training and test data. This performance is almost the same as the $90.4\%/87.3\%$ reported on the TIGER data during the CoNLL-X 2006 shared task.

Experiment	structural	labeled
MSTParser-h	90.5	87.5
MSTParser-N	90.5	87.3
MSTParser(CoNLL-X)	90.4	87.3
WCDG + MST	92.9	91.3
WCDG + MST + 5P	93.3	92.0

Table 1: Structural/labeled accuracy results with POS tagging from the gold standard. WCDG — no statistical enhancements used. MSTParser-h — MSTParser trained on the `heiseticker`. MSTParser-N — MSTParser trained on NEGRA. 5P — with all five statistical predictors of WCDG.

As is to be expected, if a real POS tagger is used in the experiments with MSTParser, the accuracy is reduced quite expectedly by approximately one

[3]The `heiseticker` dependency treebank is under preparation and will be available soon.

percent to $89.5\%/86.0\%$ (Table 2 (B)). All the results obtained with a real POS tagger are summarized in Table 2. For comparison, under the same evaluation conditions, the performance of WCDG with different predictors is summarized in Table 2 (A).

	Experiment	structural	labeled
(A)	**WCDG**	**88.0**	**86.0**
	CP	88.6	86.5
	PP	89.4	87.3
	ST	90.8	89.2
	SR	90.0	88.4
	PP+SR	90.2	88.6
	ST+SR	91.0	89.4
	ST+PP	90.8	89.2
	5P	**91.3**	**90.0**
(B)	**MSTParser**	**89.5**	**86.0**
(C)	**WCDG + MST**	**92.0**	**90.5**
	PP	92.0	90.6
	CP	92.1	90.6
	SR	92.2	90.6
	ST	92.4	90.9
	CP+SR	92.3	90.7
	CP+ST	92.6	91.0
	ST+SR	92.9	91.4
	PP+CP+ST	92.6	91.1
	PP+ST+SR	92.8	91.3
	CP+ST+SR	92.9	91.4
	5P	**92.9**	**91.4**

Table 2: Structural/labeled accuracy results with a real POS tagger. (A) WCDG experiments with different statistical enhancements (B) MSTParser experiment with a real POS tagger. (C) Combined experiments of WCDG and MSTParser with other statistical enhancements of WCDG. CP — chunker, ST — supertagger, PP — prepositional attacher, SR — shift-reduce oracle parser, 5P — POS + CP + PP + ST + SR.

The combined experiments in which MSTParser was used as a predictor for WCDG have achieved higher accuracy than each of the combined components in isolation: the structural accuracy rises to 92.0% while the labeled accuracy also gets over the 90%-boundary (WCDG + MST experiment in Table 2 (C)) .

Finally, the MSTParser predictor was evaluated in combination with the other predictors available for WCDG. The results of the experiments are shown in Table 2 (C). Every combination of MSTParser with other predictors (first four experiments) improves the accuracy. The increase is highest (0.4%) for the combination with the supertagger. This confirms earlier experiments with WCDG, in which the supertagger also contributed the largest gains.

The experimental results again confirm that WCDG is a reliable platform for information integration. Although the use of multiple predictors does not lead to an accumulation of the individual improvements, the performance of predictor combinations is always higher that using them separately. A maximum performance of $92.9\%/91.4\%$ is reached with all the six available predictors active. For comparison, the same experiment with POS tags from the gold standard has achieved even better results of $93.3\%/92.0\%$ (Table 1).

Unfortunately, the PP attacher brings accuracy reductions when it is working parallel to the shift-reduce predictor (experiment PP + CP + SR in Table 2 (C)). This effect has already been observed in the experiments that combined the two alone (experiment PP + SR in Table 2 (A)). When MST was combined with the PP attacher (experiment PP in Table 2 (C)), the increase of the performance was also below a tenth of a percent. The possible reasons why the use of an additional information source does not improve the performance in this case may be the disadvantages of the PP attacher compared to a full parser.

5 Error Analysis

A very useful property of WCDG is that it not only can be used as a parser, but also as a diagnostic tool for dependency structures. Applied to a given dependency tree, any constraint violation reported by the constraint solver indicates an inconsistency between the structure and the WCDG constraint grammar.

Among the most frequent hard constraint violations found in the MSTParser results are double subjects, double objects and direct objects in passive, projectivity violations, conjunctions without a clause as well as subordinate clause without conjunction.

These findings are in line with the analysis of

McDonald and Nivre (2007). For example, the errors in distinguishing noun complements of the verb may be due to the fact that MSTParser is more precise for longer dependency arcs and has no access to the parsing history.

In absolute figures, MSTParser commits 1509 attachment errors of which 902 are corrected by WCDG. On the other hand, WCDG adds another 542 errors of its own, so that the final result still contains 1149 errors.

For most labels, accuracy of the predictor combination is higher than in each of the parsers alone. A particularly large gain has been observed for coordinated elements (KON and CJ), subordinate (NEB) and relative (REL) clauses, indirect accusative objects (OBJA), genitive modifiers (GMOD) and apposition (APP). Table 3) summarizes the values of structural precision, the ratio of the number of correct attachment of a given label to the number of all the predictions for that label made by the parser, and label recall, the ratio between the number of correct labeling decisions and desired labeling.

In this respect, the increase in the structural precision of the PP attachment seems worth mentioning. MSTParser attaches 79.3% of PPs correctly on the used test set. Although MSTParser does not use any special PP-attachment resolution mechanisms, it is comparable with the result of WCDG combined with the PP attacher that achieves 78.7% structural precision for PP edges.

If MSTParser is trained on NEGRA excluding the test set — the rest of NEGRA lacking consistence mentioned above — it performs even better, attaching 80.4% of PP-s correctly. Thus, MSTParser as a statistical parser trained on a full corpus becomes a strong competitor for a PP attacher that has been trained on restricted four-tuples input.

As for the errors in the MSTParser output that are most often corrected in the hybrid experiment, this happens for both the structural precision and label recall of most verb complements, such as direct and indirect objects, or clausal objects as well as for subordinate and relative clauses for such subordinate clauses.

It even comes to one case in which the synergy took place in spite of the incorrect predictions. Although MSTParser has predicted possessive modifiers more seldom than WCDG alone (the label recall of MSTParser for possessive modification was over 5% below that of WCDG) its structural precision and label recall in the combined experiment are by around 6% greater than WCDG result.

	(1)		(2)		(3)	
Label	*p*	*r*	*p*	*r*	*p*	*r*
DET	98.4	99.3	98.7	99.5	99.3	99.5
PN	97.4	97.4	98.0	98.0	98.0	98.7
PP	67.6	98.1	78.3	97.4	80.1	98.5
ADV	76.6	94.7	79.4	95.4	82.2	97.2
SUBJ	94.0	90.9	91.3	86.4	95.8	94.0
ATTR	95.2	95.8	97.7	98.2	98.3	98.4
S	89.2	90.1	89.3	90.5	90.5	91.0
AUX	95.9	94.2	98.6	97.8	98.7	97.6
OBJA	87.9	83.9	83.8	72.5	92.5	88.7
APP	85.1	88.5	88.9	90.9	90.9	94.0
KON	78.9	88.1	78.9	88.3	86.0	89.2
CJ	85.6	86.5	90.9	91.4	93.0	93.5
GMOD	90.7	90.7	89.0	85.3	96.3	95.8
KONJ	88.6	91.9	91.9	95.7	95.1	95.7
PRED	90.3	75.0	85.4	60.4	91.7	76.4
NEB	68.9	82.8	73.0	66.4	79.5	90.2
REL	64.8	77.9	59.0	77.0	68.9	86.9

Table 3: Per label structural precision (*p*, $\%$) and label recal (*r*, $\%$) in comparison for the experiments with the real POS tagger (1) WCDG, (2) MSTParser, (3) WCDG combined with MSTParser

Cases in which WCDG performs worse with the predictor than its predictor alone can hardly be found. Still, one may observe many cases in which the predictor has a negative influence on the performance of WCDG, such as for different kinds of objects (indirect objects, object clauses and infinitive objects) and parenthetic matrix clauses. For all, the result of MSTParser was below that of the baseline WCDG with only the POS tagger active. Same can be said about the labeled accuracy for split verb prefixes and nominal time expressions. This worsening effect can be attributed to the lower values of the WCDG constraints for the corresponding labels and edges than for the MSTParser predictor. Thus, the search could not find a decision scoring better than that when the MSTParser prediction has been followed.

Around 15% of the sentences in the test set are

not projective. The accuracy of MSTParser on the projective sentences of the test set is higher than that on the non-projective sentences by more than 3 percent (Table 4), although these values cannot be compared directly as the mean length of non-projective sentences is longer (25.0 vs. 15.3 words).

Experiment	Non-proj.	Proj.
MSTParser (POS)	88.2	91.7
WCDG (POS)	87.2	90.2
WCDG (POS + SR)	88.7	92.2
WCDG (POS + MST)	91.3	93.6

Table 4: Structural accuracy, (%), for different parsing runs for non-projective vs. projective sentences.

MSTParser generally tends to find many more non-projective edges than the data has, while the precision remains restricted. The number of non-projective edges was determined by counting how often an edge crosses some other edge. Thus, if a non-projective edge crossed three other edges the number of non-projective edges equals three. For MSTParser experiments with a real POS tagger (MSTParser POS-experiment in Table 5), the non-projective edge recall, the ratio of the non-projective edges found in the experiment to the corresponding value in the gold standard, is at 23% and non-projective edge precision, the ratio of the correctly found non-projective edges to all non-projective edges found, is also only 36% (second column in Table 5).

	Edges		Sentences	
Experiment	*r*	*p*	*r*	*p*
MSTParser (POS)	23	36	35	44
WCDG (POS)	37	53	51	63
WCDG (POS + SR)	41	47	57	55
WCDG (POS + MST)	48	53	61	61

Table 5: Recall (*r*, %) and precision (*p*, %) of the non-projective edges and sentences for different parsing runs.

Precision and recall of non-projective sentences is a less rigid measure. If at least one edge-crossing is correctly identified in a non-projective sentence, it is added to the correctly identified non-projective sentences, even if the identified edge-crossing is not the one annotated in the gold standard and the ratios are calculated respectively (right column of Table 5). Under these relaxed conditions, MSTParser correctly identifies slightly less than a half of the non-projective sentences and over a third of non-projective edges. In fact, WCDG under the same conditions (WCDG POS-experiment in Table 5) has a non-projective sentence precision of 63% and a non-projective edge precision of 53%. Still, WCDG misses a considerable amount of non-projectivities. More importantly, as the present shift-reduce predictor has not been designed for non-projective parsing, its inclusion reduces the non-projective sentence and edge precision of WCDG — to 55% and 47% respectively — WCDG (POS+SR) in Table 5.

The expected benefits for the non-projective sentences have not yet been observed to the full extent. The precision of the combined system to find non-projective sentences and edges remained limited by the performance that WCDG was able to achieve alone (WCDG (POS+MST) in Table 5). While MSTParser in many cases predicts non-projectivity correctly WCDG is seldom capable of accepting this external evidence. On the contrary, WCDG often accepts an incorrect projective solution of the predictor instead of relying on its own cues. In its interaction with external predictors WCDG should typically decide about the alternatives.

6 Related Work

So far, approaches to hybrid parsing have been mainly based on the idea of a post-hoc selection which can be carried out for either complete parses, or individual constituents and dependency edges, respectively. The selection component itself can be based on heuristics, like a majority vote. Alternatively, a second-level classifier is trained to decide which component to trust under which conditions and therefore the approach is often referred to as classifier stacking.

In a series of experiments, Henderson and Brill (1999) combined three constituency-based parsers by a selection mechanism for either complete parsing results (parser switching) or individual constituents (parse hybridization), using both a heuristic decision rule as well as a naïve Bayesian classifier in each case. Among the heuristics considered were majority votes for constituents and a

similarity-based measure for complete trees. Tests on Penn Treebank data showed a clear improvement of the combined results over the best individual parser. Constituent selection outperformed the complete parse selection scheme, and Bayesian selection was slightly superior.

Instead of coupling different data-driven parsers which all provide comparable analyses for complete sentences, Rupp et al. (2000) combined differently elaborated structural descriptions (namely chunks and phrase structure trees) obtained by data-driven components with the output of a HPSG-parser. Driven by the requirements of the particular application (speech-to-speech translation), the focus was not only on parse selection, but also on combining incomplete results. However, no quantitative evaluation of the results has been published.

Zeman and Žabokrtský (2005) applied the selection idea to dependency structures and extended it by using more context features. They combined seven different parsers for Czech, among them also a system based on a manually compiled rule set. Some of the individual parsers had a fairly poor performance, but even a simple voting scheme on single edges contributed a significant improvement while the best results have been obtained for a combination that did not include the worst components. Alternatively the authors experimented with a trained selection component which not only had access to the alternative local parsing results, but also to their structural context. Neither a memory-based approach nor a model based on decision trees did result in further gains.

In two separate experiments, Sagae and Lavie (2006) combined a number of dependency and constituent parsers, respectively. They created a new weighted search space from the results of the individual component parsers using different weighting schemes for the candidates. They then reparsed this search space and found a consistent improvement for the dependency structures, but not for the constituent-based ones.

While all these approaches attempt to integrate the available evidence at parse time, Nivre and McDonald (2008) pursued an alternative architecture, where integration is achieved already at training time. They combined the two state-of-the-art data-driven dependency parsers, MaltParser (Nivre et al., 2006) and MSTParser (McDonald et al., 2006), by integrating the features of each of the classifiers into the parsing model of the other one at training time. Since the two parsers are based on quite different model types (namely a history-based vs. a structure-based one), they exhibit a remarkable complementary behavior (McDonald and Nivre, 2007). Accordingly, significant mutual benefits have been observed. Note, however, that one of the major benefits of MaltParser, its incremental left-to-right processing, is sacrificed under such a combination scheme.

Martins et al. (2008) use stacked learning to overcome the restriction to the single-edge features in both MaltParser and MSTParser. They suggest an architecture with two layers, where the output of a standard parser in the first level provides new features for a parser in the subsequent level. During the training phase, the second parser learns to correct mistakes made by the first one. It allows to involve higher-order predicted edges to simulate non-local features in the second parser. The results are competitive with McDonald and Nivre (2007) while $O(n^2)$ runtime of the spanning tree algorithm is preserved.

7 Conclusion

Integrating MSTParser as a full predictor with WCDG is beneficial for both of them. Since these systems take their decisions based on completely different sources of knowledge, combining both helps avoid many mistakes each of them commits in isolation. Altogether, with a real POS tagger, an accuracy level of 92.9%/91.3% has been reached (the last row in Table 2 (C)), which is higher than what any of the parsers achieved alone. With POS tagging from the gold standard, the accuracy has been at 93.3%/92.0% (the last row in Table 1). To the knowledge of the authors, these accuracy values are also better than any previous parsing results on the NEGRA test set.

WCDG can profit from the combination not only with ancillary predictors for specific parsing subtasks, but also with another full parser. This result was achieved even though the second parser is very similar to WCDG with respect to both the richness and the accuracy of its target structures. The probable reason lies in the considerable difference in the error profiles of both systems as regards specific linguistic phenomena. WCDG was also used as a diagnostic tool for the errors of MSTParser.

Possibly, a higher degree of synergy could be

achieved if a stronger coupling of the components were established by also using the scores of MSTParser as additional information for WCDG, reflecting the intuitive notion of preference or plausibility of the predictions. This could be done for the optimal parse tree alone as well as for the complete hypothesis space. Alternatively, the output of MSTParser can be used as a initial state for the transformation procedure of WCDG. Vice versa, MSTParser could be enriched with additional features based on the output of WCDG, similar to the feature-based integration of data-driven parsers evaluated by Nivre and McDonald (2008).

At the moment, the integration constraints treats all attachment and label predictions as being uniformly reliable. To individualize them with respect to their type or origin could not only make the system sensitive to qualitative differences between predictions (for instance, with respect to different labels). It would also allow the parser to accommodate multiple oracle predictors and to carefully distinguish between typical configurations in which one prediction should be preferred over an alternative one. MaltParser (Nivre et al., 2006) is certainly a good candidate for carrying out such experiments.

References

Sabine Brants, Stefanie Dipper, Silvia Hansen, Wolfgang Lezius, and George Smith. 2002. The TIGER treebank. In: *Proceedings of the First Workshop on Treebanks and Linguistic Theories (TLT)*, pages 24–41.

Sabine Buchholz and Erwin Marsi. 2006. CoNLL-X shared task on multilingual dependency parsing. In *Proc. CoNLL*, pages 149 – 164.

Kilian A. Foth and Wolfgang Menzel. 2006. Hybrid parsing: using probabilistic models as predictors for a symbolic parser. In *Proc. 21st Int. Conference on Computational Linguistics and ACL-44*, pages 321–328.

Kilian A. Foth, Wolfgang Menzel, and Ingo Schröder. 2000. A Transformation-based Parsing Technique with Anytime Properties. In *4th Int. Workshop on Parsing Technologies, IWPT-2000*, pages 89 – 100.

Kilian A. Foth. 2006. *Hybrid Methods of Natural Language Analysis*. Doctoral thesis, Hamburg University.

John C. Henderson and Eric Brill. 1999. Exploiting diversity in natural language processing: Combining parsers. In *Proceedings 4th Conference on Empirical Methods in Natural Language Processing*, pages 187–194.

André F. T. Martins, Dipanjan Das, Noah A. Smith, and Eric P. Xing. 2008. Stacking Dependency Parsers. In *Proc. of the 2008 Conf. on Empirical Methods in Natural Language Processing*, pages 157 – 166.

Hiroshi Maruyama. 1990. Structural disambiguation with constraint propagation. In *Proc. 28th Annual Meeting of the ACL (ACL-90)*, pages 31–38.

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

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič. 2005. Non-projective dependency parsing using spanning tree algorithms. In *Proc. HLT/EMNLP*, pages 523 – 530.

Ryan McDonald, Kevin Lerman, and Fernando Pereira. 2006. Multilingual dependency analysis with a two-stage discriminative parser. In *Proc. CoNLL*, pages 216 – 220.

Ryan McDonald. 2006. *Discriminative Learning and Spanning Tree Algorithms for Dependency Parsing*. PhD dissertation, University of Pennsylvania.

Joakim Nivre and Ryan McDonald. 2008. Integrating graph-based and transition-based dependency parsers. In *Proc. ACL-08: HLT*, pages 950–958.

Joakim Nivre, Johan Hall, Jens Nilsson, Gülsen Eryiğit, and Svetoslav Marinov. 2006. Labelled pseudo-projective dependency parsing with support vector machines. In *Proc. CoNLL-2006*, pages 221–225.

Christopher G. Rupp, Jörg Spilker, Martin Klarner, and Karsten L. Worm. 2000. Combining analyses from various parsers. In Wolfgang Wahlster, editor, *Verbmobil: Foundations of Speech-to-Speech Translation*, pages 311–320. Springer-Verlag, Berlin etc.

Kenji Sagae and Alon Lavie. 2006. Parser combinations by reparsing. In *Proc. HLT/NAACL*, pages 129–132.

Ingo Schröder. 2002. *Natural Language Parsing with Graded Constraints*. Ph.D. thesis, Dept. of Computer Science, University of Hamburg, Germany.

Ingo Schröder, Horia F. Pop, Wolfgang Menzel, and Kilian A. Foth. 2001. *Learning grammar weights using genetic algorithms*. In *Proc. Euroconference Recent Advances in Natural Language Processing*, pages 235 – 239.

Daniel Zeman and Zdeněk Žabokrtský. 2005. Improving parsing accuracy by combining diverse dependency parsers. In *Proc. 9th International Workshop on Parsing Technologies (IWPT-2005)*, pages 171–178, Vancouver, B.C.