Single Malt or Blended? A Study in Multilingual Parser Optimization

Johan Hall[*] **Jens Nilsson**[*] **Joakim Nivre**[*†]
Gülşen Eryiğit[‡] **Beáta Megyesi**[†] **Mattias Nilsson**[†] **Markus Saers**[†]

[*]Växjö University, School of Mathematics and Systems Engineering
E-mail: *firstname.lastname*@vxu.se

[†]Uppsala University, Dept. of Linguistics and Philology
E-mail: *firstname.lastname*@lingfil.uu.se

[‡]Istanbul Technical University, Computer Engineering Dept.
E-mail: gulsen.cebiroglu@itu.edu.tr

Abstract

We describe a two-stage optimization of the MaltParser system for the ten languages in the multilingual track of the CoNLL 2007 shared task on dependency parsing. The first stage consists in tuning a single-parser system for each language by optimizing parameters of the parsing algorithm, the feature model, and the learning algorithm. The second stage consists in building an ensemble system that combines six different parsing strategies, extrapolating from the optimal parameters settings for each language. When evaluated on the official test sets, the ensemble system significantly outperforms the single-parser system and achieves the highest average labeled attachment score.

1 Introduction

In the multilingual track of the CoNLL 2007 shared task on dependency parsing, a single parser must be trained to handle data from ten different languages: Arabic (Hajič et al., 2004), Basque (Aduriz et al., 2003), Catalan, (Martí et al., 2007), Chinese (Chen et al., 2003), Czech (Böhmová et al., 2003), English (Marcus et al., 1993; Johansson and Nugues, 2007), Greek (Prokopidis et al., 2005), Hungarian (Csendes et al., 2005), Italian (Montemagni et al., 2003), and Turkish (Oflazer et al., 2003).[1] Our contribution is a study in multilingual parser optimization using the freely available MaltParser system, which performs deterministic, classifier-based parsing with history-based feature models and discriminative learning, and which was one of the top performing systems in the CoNLL 2006 shared task (Nivre et al., 2006).

In order to maximize parsing accuracy, optimization has been carried out in two stages, leading to two different, but related parsers. The first of these is a single-parser system, similar to the one described in Nivre et al. (2006), which parses a sentence deterministically in a single left-to-right pass, with post-processing to recover non-projective dependencies, and where the parameters of the MaltParser system have been tuned for each language separately. We call this system Single Malt, to emphasize the fact that it consists of a single instance of MaltParser. The second parser is an ensemble system, which combines the output of six deterministic parsers, each of which is a variation of the Single Malt parser with parameter settings extrapolated from the first stage of optimization. It seems very natural to call this system Blended.

Section 2 summarizes the work done to optimize the Single Malt parser, while section 3 explains how the Blended parser was constructed from the Single Malt parser. Section 4 gives a brief analysis of the experimental results, and section 5 concludes.

2 The Single Malt Parser

The parameters available in the MaltParser system can be divided into three groups: parsing algorithm parameters, feature model parameters, and learning algorithm parameters.[2] Our overall optimization

[1]For more information about the task and the data sets, see Nivre et al. (2007).

[2]For a complete documentation of these parameters, see http://w3.msi.vxu.se/users/nivre/research/MaltParser.html.

Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007, pp. 933–939,
Prague, June 2007.

strategy for the Single Malt parser was as follows:

1. Define a good baseline system with the same parameter settings for all languages.
2. Tune parsing algorithm parameters once and for all for each language (with baseline settings for feature model and learning algorithm parameters).
3. Optimize feature model and learning algorithm parameters in an interleaved fashion for each language.

We used nine-fold cross-validation on 90% of the training data for all languages with a training set size smaller than 300,000 tokens and an 80%–10% train-devtest split for the remaining languages (Catalan, Chinese, Czech, English). The remaining 10% of the data was in both cases saved for a final dry run, where the parser was trained on 90% of the data for each language and tested on the remaining (fresh) 10%. We consistently used the labeled attachment score (LAS) as the single optimization criterion.

Below we describe the most important parameters in each group, define baseline settings, and report notable improvements for different languages during development. The improvements for each language from step 1 (baseline) to step 2 (parsing algorithm) and step 3 (feature model and learning algorithm) can be tracked in table 1.[3]

2.1 Parsing Algorithm

MaltParser implements several parsing algorithms, but for the Single Malt system we stick to the one used by Nivre et al. (2006), which performs labeled projective dependency parsing in linear time, using a stack to store partially processed tokens and an input queue of remaining tokens. There are three basic parameters that can be varied for this algorithm:

1. **Arc order:** The baseline algorithm is *arc-eager*, in the sense that right dependents are attached to their head as soon as possible, but there is also an *arc-standard* version, where the attachment of right dependents has to be postponed until they have found all their own dependents. The arc-standard order was found to improve parsing accuracy for Chinese, while the arc-eager order was maintained for all other languages.
2. **Stack initialization:** In the baseline version the parser is initialized with an artificial root node (with token id 0) on the stack, so that arcs originating from the root can be added explicitly during parsing. But it is also possible to initialize the parser with an empty stack, in which case arcs from the root are only added implicitly (to any token that remains a root after parsing is completed). Empty stack initialization (which reduces the amount of nondeterminism in parsing) led to improved accuracy for Catalan, Chinese, Hungarian, Italian and Turkish.[4]
3. **Post-processing:** The baseline parser performs a single left-to-right pass over the input, but it is possible to allow a second pass where only unattached tokens are processed.[5] Such post-processing was found to improve results for Basque, Catalan, Czech, Greek and Hungarian.

Since the parsing algorithm only produces projective dependency graphs, we may use pseudo-projective parsing to recover non-projective dependencies, i.e., projectivize training data and encode information about these transformations in extended arc labels to support deprojectivization of the parser output (Nivre and Nilsson, 2005). Pseudo-projective parsing was found to have a positive effect on overall parsing accuracy only for Basque, Czech, Greek and Turkish. This result can probably be explained in terms of the frequency of non-projective dependencies in the different languages. For Basque, Czech, Greek and Turkish, more than 20% of the sentences have non-projective dependency graphs; for all the remaining languages the corresponding

[3] Complete specifications of all parameter settings for all languages, for both Single Malt and Blended, are available at http://w3.msi.vxu.se/users/jha/conll07/.

[4] For Arabic, Basque, Czech, and Greek, the lack of improvement can be explained by the fact that these data sets allow more than one label for dependencies from the artificial root. With empty stack initialization all such dependencies are assigned a default label, which leads to a drop in labeled attachment score. For English, however, empty stack initialization did not improve accuracy despite the fact that dependencies from the artificial root have a unique label.

[5] This technique is similar to the one used by Yamada and Matsumoto (2003), but with only a single post-processing pass parsing complexity remains linear in string length.

Tokens	Attributes					
	FORM	LEMMA	CPOSTAG	POSTAG	FEATS	DEPREL
S: Top	+	+	+	+	+	+
S: Top-1				+		
I: Next	+	+	+	+	+	
I: Next+1	+			+		
I: Next+2				+		
I: Next+3				+		
G: Head of Top	+					
G: Leftmost dependent of Top						+
G: Rightmost dependent of Top						+
G: Leftmost dependent of Next						+

Figure 1: Baseline feature model (S = Stack, I = Input, G = Graph).

figure is 10% or less.[6]

The cumulative improvement after optimization of parsing algorithm parameters was a modest 0.32 percentage points on average over all ten languages, with a minimum of 0.00 (Arabic, English) and a maximum of 0.83 (Czech) (cf. table 1).

2.2 Feature Model

MaltParser uses a history-based feature model for predicting the next parsing action. Each feature of this model is an attribute of a token defined relative to the current stack S, input queue I, or partially built dependency graph G, where the attribute can be any of the symbolic input attributes in the CoNLL format: FORM, LEMMA, CPOSTAG, POSTAG and FEATS (split into atomic attributes), as well as the DEPREL attribute of tokens in the graph G. The baseline feature model is depicted in figure 1, where rows denote tokens, columns denote attributes, and each cell containing a plus sign represents a model feature.[7] This model is an extrapolation from many previous experiments on different languages and usually represents a good starting point for further optimization.

The baseline model was tuned for each of the ten languages using both forward and backward feature selection. The total number of features in the tuned models varies from 18 (Turkish) to 56 (Hungarian) but is typically between 20 and 30. This feature selection process constituted the major development effort for the Single Malt parser and also gave the greatest improvements in parsing accuracy, but since feature selection was to some extent interleaved with learning algorithm optimization, we only report the cumulative effect of both together in table 1.

2.3 Learning Algorithm

MaltParser supports several learning algorithms but the best results have so far been obtained with support vector machines, using the LIBSVM package (Chang and Lin, 2001). We use a quadratic kernel $K(x_i, x_j) = (\gamma x_i^T x_j + r)^2$ and LIBSVM's built-in one-versus-one strategy for multi-class classification, converting symbolic features to numerical ones using the standard technique of binarization. As our baseline settings, we used $\gamma = 0.2$ and $r = 0$ for the kernel parameters, $C = 0.5$ for the penalty parameter, and $\epsilon = 1.0$ for the termination criterion. In order to reduce training times during development, we also split the training data for each language into smaller sets and trained separate multi-class classifiers for each set, using the POSTAG of Next as the defining feature for the split.

The time spent on optimizing learning algorithm parameters varies between languages, mainly due to lack of time. For Arabic, Basque, and Catalan, the baseline settings were used also in the dry run and final test. For Chinese, Greek and Hungarian,

[6] In fact, for Arabic, which has about 10% sentences with non-projective dependencies, it was later found that, with an optimized feature model, it is beneficial to projectivize the training data without trying to recover non-projective dependencies in the parser output. This was also the setting that was used for Arabic in the dry run and final test.

[7] The names Top and Next refer to the token on top of the stack S and the first token in the remaining input I, respectively.

Language	Development			Dry Run		Test		Test: UAS	
	Base	**PA**	**F+L**	**SM**	**B**	**SM**	**B**	**SM**	**B**
Arabic	70.31	70.31	71.67	70.93	73.09	74.75	76.52	84.21	85.81
Basque	73.86	74.44	76.99	77.18	80.12	74.97	76.92	80.61	82.84
Catalan	85.43	85.51	86.88	86.65	88.00	87.74	88.70	92.20	93.12
Chinese	83.85	84.39	87.64	87.61	88.61	83.51	84.67	87.60	88.70
Czech	75.00	75.83	77.74	77.91	82.17	77.22	77.98	82.35	83.59
English	85.44	85.44	86.35	86.35	88.74	85.81	88.11	86.77	88.93
Greek	72.67	73.04	74.42	74.89	78.17	74.21	74.65	80.66	81.22
Hungarian	74.62	74.64	77.40	77.81	80.04	78.09	80.27	81.71	83.55
Italian	81.42	81.64	82.50	83.37	85.16	82.48	84.40	86.26	87.77
Turkish	75.12	75.80	76.49	75.87	77.09	79.24	79.79	85.04	85.77
Average	77.78	78.10	79.81	79.86	82.12	79.80	81.20	84.74	86.13

Table 1: Development results for Single Malt (Base = baseline, PA = parsing algorithm, F+L = feature model and learning algorithm); dry run and test results for Single Malt (SM) and Blended (B) (with corrected test scores for Blended on Chinese). All scores are labeled attachment scores (LAS) except the last two columns, which report unlabeled attachment scores (UAS) on the test sets.

slightly better results were obtained by not splitting the training data into smaller sets; for the remaining languages, accuracy was improved by using the CPOSTAG of Next as the defining feature for the split (instead of POSTAG). With respect to the SVM parameters (γ, r, C, and ϵ), Arabic, Basque, Catalan, Greek and Hungarian retain the baseline settings, while the other languages have slightly different values for some parameters.

The cumulative improvement after optimization of feature model and learning algorithm parameters was 1.71 percentage points on average over all ten languages, with a minimum of 0.69 (Turkish) and a maximum of 3.25 (Chinese) (cf. table 1).

3 The Blended Parser

The Blended parser is an ensemble system based on the methodology proposed by Sagae and Lavie (2006). Given the output dependency graphs G_i ($1 \leq i \leq m$) of m different parsers for an input sentence x, we construct a new graph containing all the labeled dependency arcs proposed by some parser and weight each arc a by a score $s(a)$ reflecting its popularity among the m parsers. The output of the ensemble system for x is the maximum spanning tree of this graph (rooted at the node 0), which can be extracted using the Chu-Liu-Edmonds algorithm, as shown by McDonald et al. (2005). Following Sagae and Lavie (2006), we let $s(a) = \sum_{i=1}^{m} w_i^c a_i$, where w_i^c is the average labeled attachment score of parser i for the word class c[8] of the dependent of a, and a_i is 1 if $a \in G_i$ and 0 otherwise.

The Blended parser uses six component parsers, with three different parsing algorithms, each of which is used to construct one left-to-right parser and one right-to-left parser. The parsing algorithms used are the arc-eager baseline algorithm, the arc-standard variant of the baseline algorithm, and the incremental, non-projective parsing algorithm first described by Covington (2001) and recently used for deterministic classifier-based parsing by Nivre (2007), all of which are available in MaltParser. Thus, the six component parsers for each language were instances of the following:

1. Arc-eager projective left-to-right
2. Arc-eager projective right-to-left
3. Arc-standard projective left-to-right
4. Arc-standard projective right-to-left
5. Covington non-projective left-to-right
6. Covington non-projective right-to-left

[8] We use CPOSTAG to determine the part of speech.

	root		1		2		3–6		7+	
Parser	R	P	R	P	R	P	R	P	R	P
Single Malt	87.01	80.36	95.08	94.87	86.28	86.67	77.97	80.23	68.98	71.06
Blended	92.09	74.20	95.71	94.92	87.55	88.12	78.66	83.02	65.29	78.14

Table 2: Recall (R) and precision (P) of Single Malt and Blended for dependencies of different length, averaged over all languages (root = dependents of root node, regardless of length).

The final Blended parser was constructed by reusing the tuned Single Malt parser for each language (arc-standard left-to-right for Chinese, arc-eager left-to-right for the remaining languages) and training five additional parsers with the same parameter settings except for the following mechanical adjustments:

1. Pseudo-projective parsing was not used for the two non-projective parsers.
2. Feature models were adjusted with respect to the most obvious differences in parsing strategy (e.g., by deleting features that could never be informative for a given parser).
3. Learning algorithm parameters were adjusted to speed up training (e.g., by always splitting the training data into smaller sets).

Having trained all parsers on 90% of the training data for each language, the weights w_i^c for each parser i and coarse part of speech c was determined by the labeled attachment score on the remaining 10% of the data. This means that the results obtained in the dry run were bound to be overly optimistic for the Blended parser, since it was then evaluated on the same data set that was used to tune the weights.

Finally, we want to emphasize that the time for developing the Blended parser was severely limited, which means that several shortcuts had to be taken, such as optimizing learning algorithm parameters for speed rather than accuracy and using extrapolation, rather than proper tuning, for other important parameters. This probably means that the performance of the Blended system can be improved considerably by optimizing parameters for all six parsers separately.

4 Results and Discussion

Table 1 shows the labeled attachment score results from our internal dry run (training on 90% of the training data, testing on the remaining 10%) and the official test runs for both of our systems. It should be pointed out that the test score for the Blended parser on Chinese is different from the official one (75.82), which was much lower than expected due to a corrupted specification file required by MaltParser. Restoring this file and rerunning the parser on the Chinese test set, without retraining the parser or changing any parameter settings, resulted in the score reported here. This also improved the average score from 80.32 to 81.20, the former being the highest reported official score.

For the Single Malt parser, the test results are on average very close to the dry run results, indicating that models have not been overfitted (although there is considerably variation between languages). For the Blended parser, there is a drop of almost one percentage point, which can be explained by the fact that weights could not be tuned on held-out data for the dry run (as explained in section 3).

Comparing the results for different languages, we see a tendency that languages with rich morphology, usually accompanied by flexible word order, get lower scores. Thus, the labeled attachment score is below 80% for Arabic, Basque, Czech, Greek, Hungarian, and Turkish. By comparison, the more configurational languages (Catalan, Chinese, English, and Italian) all have scores above 80%. Linguistic properties thus seem to be more important than, for example, training set size, which can be seen by comparing the results for Italian, with one of the smallest training sets, and Czech, with one of the largest. The development of parsing methods that are better suited for morphologically rich languages with flexible word order appears as one of the most important goals for future research in this area.

Comparing the results of our two systems, we see that the Blended parser outperforms the Single Malt parser for all languages, with an average improvement of 1.40 percentage points, a minimum of 0.44 (Greek) and a maximum of 2.40 (English). As shown by McDonald and Nivre (2007), the Single Malt parser tends to suffer from two problems: error propagation due to the deterministic parsing strategy, typically affecting long dependencies more than short ones, and low precision on dependencies originating in the artificial root node due to fragmented parses.[9] The question is which of these problems is alleviated by the multiple views given by the component parsers in the Blended system. Table 2 throws some light on this by giving the precision and recall for dependencies of different length, treating dependents of the artificial root node as a special case. As expected, the Single Malt parser has lower precision than recall for root dependents, but the Blended parser has even lower precision (and somewhat better recall), indicating that the fragmentation is even more severe in this case.[10] By contrast, we see that precision and recall for other dependencies improve across the board, especially for longer dependencies, which probably means that the effect of error propagation is mitigated by the use of an ensemble system, even if each of the component parsers is deterministic in itself.

5 Conclusion

We have shown that deterministic, classifier-based dependency parsing, with careful optimization, can give highly accurate dependency parsing for a wide range of languages, as illustrated by the performance of the Single Malt parser. We have also demonstrated that an ensemble of deterministic, classifier-based dependency parsers, built on top of a tuned single-parser system, can give even higher accuracy, as shown by the results of the Blended parser, which has the highest labeled attachment score for five languages (Arabic, Basque, Catalan, Hungarian, and Italian), as well as the highest multilingual average score.

[9]A fragmented parse is a dependency forest, rather than a tree, and is automatically converted to a tree by attaching all (other) roots to the artificial root node. Hence, children of the root node in the final output may not have been predicted as such by the treebank-induced classifier.

[10]This conclusion is further supported by the observation that the single most frequent "frame confusion" of the Blended parser, over all languages, is to attach two dependents with the label ROOT to the root node, instead of only one. The frequency of this error is more than twice as high for the Blended parser (180) as for the Single Malt parser (83).

Acknowledgements

We want to thank all treebank providers for making the data available for the shared task and the (other) organizers for their efforts in organizing it. Special thanks to Ryan McDonald, for fruitful discussions and assistance with the error analysis, and to Kenji Sagae, for showing us how to produce a good blend. Thanks also to two reviewers for useful comments.

References

A. Abeillé, editor. 2003. *Treebanks: Building and Using Parsed Corpora*. Kluwer.

I. Aduriz, M. J. Aranzabe, J. M. Arriola, A. Atutxa, A. Diaz de Ilarraza, A. Garmendia, and M. Oronoz. 2003. Construction of a Basque dependency treebank. In *Proc. of the 2nd Workshop on Treebanks and Linguistic Theories (TLT)*, pages 201–204.

A. Böhmová, J. Hajič, E. Hajičová, and B. Hladká. 2003. The PDT: a 3-level annotation scenario. In Abeillé (2003), chapter 7, pages 103–127.

C.-C. Chang and C.-J. Lin, 2001. *LIBSVM: A Library for Support Vector Machines*. Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm.

K. Chen, C. Luo, M. Chang, F. Chen, C. Chen, C. Huang, and Z. Gao. 2003. Sinica treebank: Design criteria, representational issues and implementation. In Abeillé (2003), chapter 13, pages 231–248.

M. A. Covington. 2001. A fundamental algorithm for dependency parsing. In *Proc. of the 39th Annual ACM Southeast Conf.*, pages 95–102.

D. Csendes, J. Csirik, T. Gyimóthy, and A. Kocsor. 2005. *The Szeged Treebank*. Springer.

J. Hajič, O. Smrž, P. Zemánek, J. Šnaidauf, and E. Beška. 2004. Prague Arabic dependency treebank: Development in data and tools. In *Proc. of the NEMLAR Intern. Conf. on Arabic Language Resources and Tools*, pages 110–117.

R. Johansson and P. Nugues. 2007. Extended constituent-to-dependency conversion for English. In *Proc. of the 16th Nordic Conference on Computational Linguistics (NODALIDA)*.

M. Marcus, B. Santorini, and M. Marcinkiewicz. 1993. Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics*, 19(2):313–330.

M. A. Martí, M. Taulé, L. Màrquez, and M. Bertran. 2007. CESS-ECE: A multilingual and multilevel annotated corpus. Available for download from: http://www.lsi.upc.edu/∼mbertran/cess-ece/.

R. McDonald and J. Nivre. 2007. Characterizing the errors of data-driven dependency parsing models. In *Proc. of the Joint Conf. on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*.

R. McDonald, F. Pereira, K. Ribarov, and J. Hajič. 2005. Non-projective dependency parsing using spanning tree algorithms. In *Proc. of the Human Language Technology Conf. and the Conf. on Empirical Methods in Natural Language Processing (HLT/EMNLP)*, pages 523–530.

S. Montemagni, F. Barsotti, M. Battista, N. Calzolari, O. Corazzari, A. Lenci, A. Zampolli, F. Fanciulli, M. Massetani, R. Raffaelli, R. Basili, M. T. Pazienza, D. Saracino, F. Zanzotto, N. Nana, F. Pianesi, and R. Delmonte. 2003. Building the Italian Syntactic-Semantic Treebank. In Abeillé (2003), chapter 11, pages 189–210.

J. Nivre and J. Nilsson. 2005. Pseudo-projective dependency. In *Proc. of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 99–106.

J. Nivre, J. Hall, J. Nilsson, G. Eryiğit, and S. Marinov. 2006. Labeled pseudo-projective dependency parsing with support vector machines. In *Proc. of the Tenth Conf. on Computational Natural Language Learning (CoNLL)*, pages 221–225.

J. Nivre, J. Hall, S. Kübler, R. McDonald, J. Nilsson, S. Riedel, and D. Yuret. 2007. The CoNLL 2007 shared task on dependency parsing. In *Proc. of the Joint Conf. on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*.

J. Nivre. 2007. Incremental non-projective dependency parsing. In *Human Language Technologies: The Annual Conf. of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT)*, pages 396–403.

K. Oflazer, B. Say, D. Zeynep Hakkani-Tür, and G. Tür. 2003. Building a Turkish treebank. In Abeillé (2003), chapter 15, pages 261–277.

P. Prokopidis, E. Desypri, M. Koutsombogera, H. Papageorgiou, and S. Piperidis. 2005. Theoretical and practical issues in the construction of a Greek dependency treebank. In *Proc. of the 4th Workshop on Treebanks and Linguistic Theories (TLT)*, pages 149–160.

K. Sagae and A. Lavie. 2006. Parser combination by reparsing. In *Proc. of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*, pages 129–132.

H. Yamada and Y. Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *Proc. 8th International Workshop on Parsing Technologies (IWPT)*, pages 195–206.