Global Learning of Labelled Dependency Trees

Michael Schiehlen **Kristina Spranger**
Institute for Computational Linguistics
University of Stuttgart
D-70174 Stuttgart
Michael.Schiehlen@ims.uni-stuttgart.de
Kristina.Spranger@ims.uni-stuttgart.de

Abstract

In the paper we describe a dependency parser that uses exact search and global learning (Crammer et al., 2006) to produce labelled dependency trees. Our system integrates the task of learning tree structure and learning labels in one step, using the same set of features for both tasks. During label prediction, the system automatically selects for each feature an appropriate level of smoothing. We report on several experiments that we conducted with our system. In the shared task evaluation, it scored better than average.

1 Introduction

Dependency parsing is a topic that has engendered increasing interest in recent years. One promising approach is based on exact search and structural learning (McDonald et al., 2005; McDonald and Pereira, 2006). In this work we also pursue this approach. Our system makes no provisions for non-projective edges. In contrast to previous work, we aim to learn labelled dependency trees at one fell swoop. This is done by maintaining several copies of feature vectors that capture the features' impact on predicting different dependency relations (deprels). In order to preserve the strength of McDonald et al. (2005)'s approach in terms of unlabelled attachment score, we add feature vectors for generalizations over deprels. We also employ various reversible transformations to reach treebank formats that better match our feature representation and that reduce the complexity of the learning task. The paper first presents the methodology used, goes on to describing experiments and results and finally concludes.

2 Methodology

2.1 Parsing Algorithm

In our approach, we adopt Eisner (1996)'s bottom-up chart-parsing algorithm in McDonald et al. (2005)'s formulation, which finds the best projective dependency tree for an input string $\boldsymbol{x} = \langle x_1, \ldots, x_n \rangle$. We assume that every possible head–dependent pair i, j is described by a feature vector $\boldsymbol{\Phi}_{ij}$ with associated weights $\boldsymbol{w}_{ij}$. Eisner's algorithm achieves optimal tree packing by storing partial structures in two matrices $\triangle$ and $\triangleleft$. First the diagonals of the matrices are initiated with 0; then all other cells are filled according to eqs. (1) and (2) and their symmetric variants.

$$
\begin{aligned}
{}^{i}\!\searrow_{j} &= \boldsymbol{w}_{ij} \cdot \boldsymbol{\Phi}_{ij} \\
{}_{i}\triangle_{j} &= \max_{k:i \le k < j} {}_{i}\triangleleft_{k} + {}_{k+1}\triangle_{j} + {}^{i}\!\searrow_{j} \qquad (1) \\
{}_{i}\triangleleft_{j} &= \max_{k:i < k \le j} {}_{i}\triangle_{k} + {}_{k}\triangleleft_{j} \qquad (2) \\
\text{root} &= \max_{k:1 \le k \le n} {}_{1}\triangle_{k} + {}_{k}\triangleleft_{n} + \boldsymbol{w}_{0k} \cdot \boldsymbol{\Phi}_{0k}
\end{aligned}
$$

This algorithm only accommodates features for single links in the dependency graph. We also investigated an extension, McDonald and Pereira (2006)'s *second-order model*, where more of the parsing history is taken into account, viz. the last dependent k assigned to a head i. In the extended model, $\triangle$ is updated as defined in eq. (3); optimal packing requires a third matrix $\blacktriangle$.

Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007, pp. 1156–1160,
Prague, June 2007. ©2007 Association for Computational Linguistics

$$
\begin{aligned}
{}_i\triangle_j &= \max_{k:i\le k<j} \left\{ \begin{array}{ll} {}_{i+1}\triangle_j & \text{if } k=i \\ {}_i\triangle_k + {}_k\square_j & \text{else} \end{array} \right\} \\
&\quad \boldsymbol{w}_{ijk} \cdot \boldsymbol{\Phi}_{ijk} \qquad (3) \\
{}_i\square_j &= \max_{k:i\le k<j} {}_i\triangle_k + {}_{k+1}\triangle_j
\end{aligned}
$$

2.2 Feature Representation

In deriving features, we used all information given in the treebanks, i.e. words (*w*), fine-grained POS tags (*fp*), combinations of lemmas and coarse-grained POS tags (*lcp*), and whether two tokens agree[1] (*agr* = yes, no, don't know). We essentially employ the same set of features as McDonald et al. (2005): $\boldsymbol{\phi}'_{ij} = \{w_i,\ fp_i,\ lcp_i,\ w_j,\ fp_j,\ lcp_j,\ w_i w_j,\ w_i lcp_j,\ lcp_i w_j,\ lcp_i lcp_j,\ fp_i lcp_j,\ fp_i fp_j,\ fp_i fp_j agr_{ij},\ fp_{i-1} fp_i fp_{j-1} fp_j,\ fp_{i-1} fp_i fp_j fp_{j+1},\ fp_i fp_{i+1} fp_{j-1} fp_j,\ fp_i fp_{i+1} fp_j fp_{j+1}\}$, and token features for root words $\boldsymbol{\phi}_{0r} = \{w_r, fp_r, lcp_r\}$. In the first order model, we recorded the tag of each token m between i and j ($\boldsymbol{\phi}_{ij} = \boldsymbol{\phi}'_{ij} \cup \{fp_i fp_j fp_m\}$); in the second order model, we only conditioned on the previous dependent k ($\boldsymbol{\phi}_{ij} = \boldsymbol{\phi}'_{ij} \cup \{fp_i fp_j fp_k,\ lcp_i fp_j fp_k,\ w_i fp_j fp_k\}$). All features but unary token features were optionally extended with direction of dependency ($i < j$ or $i > j$) and binned token distance ($|i-j| = 1, 2, 3, 4, \ge 5, \ge 10$).

2.3 Structural Learning

For determining feature weights $\boldsymbol{w}$, we used online passive–aggressive learning (OPAL) (Crammer et al., 2006). OPAL iterates repeatedly over all training instances $\boldsymbol{x}$, adapting weights after each parse. It tries to change weights as little as possible (*passiveness*), while ensuring that (1) the correct tree $\boldsymbol{y}$ gets at least as much weight as the best parse tree $\hat{\boldsymbol{y}}$ and (2) the difference in weight between $\boldsymbol{y}$ and $\hat{\boldsymbol{y}}$ rises with the average number of errors in $\hat{\boldsymbol{y}}$ (*aggressiveness*). This optimization problem has a closed–form solution:

$$\boldsymbol{w}^{(t+1)} = \boldsymbol{w}^{(t)} + \tau_t(\boldsymbol{\Phi}(\boldsymbol{x}, \boldsymbol{y}) - \boldsymbol{\Phi}(\boldsymbol{x}, \hat{\boldsymbol{y}}))$$

where

$$\tau_t = \frac{\boldsymbol{w} \cdot \boldsymbol{\Phi}(\boldsymbol{x}, \hat{\boldsymbol{y}}) - \boldsymbol{w} \cdot \boldsymbol{\Phi}(\boldsymbol{x}, \boldsymbol{y}) + \sqrt{1 - \mathrm{LAS}(\boldsymbol{y}, \hat{\boldsymbol{y}})}}{\|\boldsymbol{\Phi}(\boldsymbol{x}, \boldsymbol{y}) - \boldsymbol{\Phi}(\boldsymbol{x}, \hat{\boldsymbol{y}})\|^2}$$

[1] Agreement was computed from morphological features, viz. gender, number and person, and case. In languages with subject–verb agreement, we added a nominative case feature to finite verbs. In Basque, agreement is case-specific (absolutive, dative, ergative, other case).

model order	# of features	min. per iteration	LAS
1	327,743	13.6	78.62
1	601.125	19.5	78.87
1	1,168,609	38.7	79.03
1	12,948,376 (513,611)	120.0 (13.3)	79.53
2	758,433	17.8	78.12
2	1,534,484	25.1	78.40
2	3,257,012 (181,303)	50.0 (9.8)	78.92
2	26,088,102 (582,907)	373.0 (23.5)	79.26

Table 1: Performance on devset of Italian treebank. In parentheses: reduction to non-null features after first iteration.

Having a closed–form solution, OPAL is easier to implement and more efficient than the MIRA algorithm used by McDonald et al. (2005), although it achieves a performance comparable to MIRA's on many problems (Crammer et al., 2006).

2.4 Learning Labels for Dependency Relations

So far, the presented system, which follows closely the approach of McDonald et al. (2005), only predicts unlabelled dependency trees. To derive a labeling, we departed from their approach: We split each feature along the deprel label dimension, so that each deprel d is associated with its own feature vector (cf. eq. (4), where $\otimes$ is the tensor product and Λ^o the orthogonal encoding).

$$\Phi_{ij,\Lambda^o(d)} = \phi_{ij} \otimes \Lambda^o(d) \qquad (4)$$

In parsing, we only consider the best deprel label.

$${}^i\searrow_j = \max_{d \in \mathcal{D}} w_{ij,\Lambda^o(d)} \Phi_{ij,\Lambda^o(d)} \qquad (5)$$

On its own, this simple approach led to a severe degradation of performance, so we took a step back by re-introducing features for unlabelled trees. For each set of deprels $\mathcal{D}$, we designed a taxonomy $\mathcal{T}$ with a single maximal element (complete abstraction over deprel labels) and one minimal element for each deprel label. We also included an intermediate layer in $\mathcal{T}$ that collects *classes* of deprels, such as

Language	# tokens			DevTest	# of	min. per
	Train	DevTest	Test	Split	Features	Cycle
Catalan	425,915	4,929	5,016	89–1	3,055,518	575.0
Basque	48,019	2,507	5,390	19–1	1,837,155	37.4
Turkish	61,951	3,231	4,513	19–1	1,412,000	26.1
English	441,333	5,240	5,003	86–1	3,609,671	727.2
Greek	62,137	3,282	4,804	19–1	2,723,891	58.0
Hungarian	123,266	8,533	7,344	15–1	2,583,593	148.2
Czech	427,338	4,958	4,724	88–1	1,971,599	591.6
Chinese	333,148	4,027	5,161	82–1	1,672,360	1,015.2
Italian	67,593	3,606	5,096	19–1	1,534,485	52.0
Arabic	107,804	3,865	5,124	27–1	1,763,063	110.0

Table 2: Figures for Experiments on Treebanks.

complement, adjunct, marker, punctuation, or coordination deprels, and in this way provides for better smoothing. The taxonomy translates to an encoding Λ^t, where $\lambda_i^t(d) = 1$ iff node i in $\mathcal{T}$ is an ancestor of d (Tsochantaridis et al., 2004). Substituting Λ^t for Λ^o leads to a massive amount of features, so we pruned the taxonomy on a feature–to–feature basis by merging all nodes on a level that only encompass deprels that never occur with this feature in the training data.

2.5 Treebank Transformations

Having no explicit feature representation for the information in the morphological features slot (cf. section 2.2), we partially redistributed that information to other slots: Verb form, case[2] to *fp*, semantic classification to an empty lemma slot (Turkish affixes, e.g. "Able", "Ly"). The balance between *fp* and *w* was not always optimal; we used a fine-grained[3] classification in punctuation tags, distinguished between prepositions (e.g. *in*) and preposition–article combinations (e.g. *nel*) in Italian[4] on the basis of number/gender features, and collected definite and indefinite articles under one common *fp* tag.

When distinctions in deprels are recoverable from context, we removed them: The dichotomy between conjunctive and disjunctive coordination in Italian depends in most cases exclusively on the coordinating conjunction. The Greek and Czech treebanks have a generic distinction between ordinary deprels and deprels in a coordination, apposition, and parenthesis construction. In Greek, we got rid of the parenthesis markers on deprels by switching head and dependent, giving the former head (the parenthesis) a unique new deprel. For Czech, we reduced the number of deprels from 46 to 34 by swapping the deprels of conjuncts, appositions, etc. and their heads (coordination or comma). Sometimes, multiple conjuncts take different deprels. We only provided for the clash between "ExD" (ellipsis) and other deprels, in which case we added "ExD", see below.

1	Minimálně	3	AuxZ		
2	dva	3	Atr		
3	stupně	0	ExD		
4	rozlišení	5	Atr_M	⇒	-Apos
5	-	3	Apos	⇒	Atr
6	standard	7	ExD_M	⇒	-Coord
7	a	5	Coord_M	⇒	-Apos:ExD
8	jemně	7	ExD_M	⇒	-Coord
9	.	0	AuxK		

In Basque, agreement is usually between arguments and *auxiliary* verbs, so we re-attached[5] relevant arguments from main verb to auxiliary verb.

The training set for Arabic contains some very long sentences (up to 396 tokens). Since context-free parsing sentences of this length is tedious, we split up all sentences at final punctuation signs

[2]Case was transferred to *fp* only if important for determination of deprel (CA, HU, IT).

[3]Classes of punctuation are e.g. opening and closing brackets, commas and punctuation signalling the end of a sentence.

[4]Prep and PrepArt behave differently syntactically (e.g. an article can only follow a genuine preposition).

[5]Unfortunately, we did not take into account projectivity, so this step resulted in a steep increase of non-projective edges (9.4% of all edges) and a corresponding degradation of our evaluation results in Basque.

Language	LAS			UAS			LAcc		
	Dev	Test	AV	Dev	Test	AV	Dev	Test	AV
Basque	68.85	66.75	**68.06**	74.59	73.25	**75.15**	78.82	**76.64**	76.06
Greek	73.49	**72.29**	70.22	82.08	**80.47**	77.78	84.19	**83.16**	81.26
Turkish	70.30	72.48	**73.19**	77.97	79.33	**80.33**	81.67	82.18	**82.29**
Italian	78.23	**80.46**	78.06	82.50	**84.54**	82.45	86.30	**87.44**	85.75
Arabic	69.26	**70.08**	68.34	79.61	**81.07**	78.84	82.25	**82.32**	81.79
Hungarian	74.29	**73.90**	71.49	78.69	**78.61**	76.34	87.82	**87.60**	85.89
Chinese	84.06	**80.04**	76.59	88.25	**85.45**	81.98	87.04	**83.28**	80.16
Catalan	85.17	**85.75**	79.85	90.04	**90.79**	87.98	91.13	**91.29**	86.32
Czech	73.26	**73.86**	70.12	81.63	**81.73**	77.56	81.36	**82.03**	79.66
English	86.93	**86.21**	80.95	88.45	**88.91**	82.67	91.97	**90.89**	87.69
Basque (rev.)	72.32	**70.48**	68.06	77.78	**76.72**	75.15	80.57	**78.85**	76.06
Turkish (rev.)	74.50	**76.31**	73.19	81.12	**82.76**	80.33	84.90	**85.46**	82.29

Table 3: Results on DevTest and Test Sets compared with the Average Performance in CoNLL'07. LAS = Labelled Attachment Score, UAS = Unlabelled Attachment Score, LAcc = Label Accuracy, AV = Average score.

(AuxK). With this trick, we pushed down maximal sentence length to 196.

Unfortunately, we overlooked the fact that in Turkish, the ROOT deprel not only designates root nodes but also attaches some punctuation marks. This often leads to non-projective structures, which our parser cannot handle, so our parser scored below average in Turkish. In after–deadline experiments, we took this feature of the Turkish treebank into account and achieved above–average results by re-linking all ROOT-ed punctuation signs to the immediately preceding token.

3 Experiments and Results

All experiments were conducted on the treebanks provided in the shared task (Hajič et al., 2004; Aduriz et al., 2003; Martí et al., 2007; Chen et al., 2003; Böhmová et al., 2003; Marcus et al., 1993; Johansson and Nugues, 2007; Prokopidis et al., 2005; Csendes et al., 2005; Montemagni et al., 2003; Oflazer et al., 2003). For our contribution, we used the second-order algorithm; only afterwards did we also apply the first-order model to the data, with quite good results (cf. Table 1). For testing our approach, we split the treebanks provided into an actual training and a development set (details are in Table 2). From each training set, we extracted at least a million features (not counting the split for deprel labels). The last column in Table 2 shows the average time needed in a training iteration.

For nearly all languages, our approach achieved a performance better than average (see Table 3). Only in Turkish and Basque did we score below average. On closer inspection, we saw that this performance was due to our projectivity assumption and to insufficient exploration of these treebanks. In its bottom part, Table 3 gives results of improved versions of our approach.

4 Conclusion

We presented an approach to dependency parsing that is based on exact search and global learning. Special emphasis is laid on an integrated derivation of labelled and unlabelled dependency trees. We also employed various transformation techniques to reach treebank formats that are better suited to our approach. The approach scores better than average in (nearly) all languages. Nevertheless, it is still a long way from cutting–edge performance. One direction we would like to explore in the future is the integration of dynamic features on deprel labels.

Acknowledgements

We would like to thank the organizing team for making possible again a great shared task at CoNLL!

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 (Abeillé, 2003), chapter 7, pages 103–127.

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 (Abeillé, 2003), chapter 13, pages 231–248.

Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. 2006. Online Passive–Aggressive Algorithms. *Journal of Machine Learning*, 7:551–585.

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

Jason M. Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of the 16th International Conference on Computational Linguistics (COLING '96)*, Copenhagen, Denmark.

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/.

Ryan McDonald and Fernando Pereira. 2006. Online Learning of Approximate Dependency Parsing Algorithms. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL'06)*, Trento, Italy.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005. Online Large-Margin Training of Dependency Parsers. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*.

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 (Abeillé, 2003), chapter 11, pages 189–210.

K. Oflazer, B. Say, D. Zeynep Hakkani-Tür, and G. Tür. 2003. Building a Turkish treebank. In Abeillé (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.

Ioannis Tsochantaridis, Thomas Hofmann, Thorsten Joachims, and Yasemin Altun. 2004. Support Vector Machine Learning for Interdependent and Structured Output Spaces. In *Proceedings of the 21st International Conference on Machine Learning*.