A Constraint Satisfaction Approach to Dependency Parsing

Sander Canisius
ILK / Communication and Information Science
Tilburg University, P.O. Box 90153,
NL-5000 LE Tilburg, The Netherlands
S.V.M.Canisius@uvt.nl

Erik Tjong Kim Sang
ISLA, University of Amsterdam,
Kruislaan 403, NL-1098 SJ Amsterdam,
The Netherlands
erikt@science.uva.nl

Abstract

We present an adaptation of constraint satisfaction inference (Canisius et al., 2006b) for predicting dependency trees. Three different classifiers are trained to predict weighted soft-constraints on parts of the complex output. From these constraints, a standard weighted constraint satisfaction problem can be formed, the solution to which is a valid dependency tree.

1 Introduction

Like the CoNLL-2006 shared task, the 2007 shared task focuses on dependency parsing and aims at comparing state-of-the-art machine learning algorithms applied to this task (Nivre et al., 2007). For our official submission, we used the dependency parser described by Canisius et al. (2006a). In this paper, we present a novel approach to dependency parsing based on constraint satisfaction. The method is an adaptation of earlier work using constraint satisfaction techniques for predicting sequential outputs (Canisius et al., 2006b). We evaluated our approach on all ten data sets of the 2007 shared task[1].

In the remainder of this paper, we will present the new constraint satisfaction method for dependency parsing in Section 2. The method is evaluated in Section 3, in which we will also present a brief error analysis. Finally, Section 4 presents our main conclusions.

[1] 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)

2 Constraint Satisfaction Inference for Dependency Trees

The parsing algorithm we used is an adaptation for dependency trees of the constraint satisfaction inference method for sequential output structures proposed by Canisius et al. (2006b). The technique uses standard classifiers to predict a weighted constraint satisfaction problem, the solution to which is the complete dependency tree. Constraints that are predicted each cover a small part of the complete tree, and overlap between them ensures that global output structure is taken into account, even though the classifiers only make local predictions in isolation of each other.

A weighted constraint satisfaction problem (W-CSP) is a tuple (X, D, C, W). Here, $X = \{x_1, x_2, \ldots, x_n\}$ is a finite set of variables. $D(x)$ is a function that maps each variable to its domain, and C is a set of constraints on the values assigned to the variables. For a traditional (non-weighted) constraint satisfaction problem, a valid solution is an assignment of values to the variables that (1) are a member of the corresponding variable's domain, and (2) satisfy *all* constraints in the set C. Weighted constraint satisfaction, however, relaxes this requirement to satisfy all constraints. Instead, constraints are assigned weights that may be interpreted as reflecting the importance of satisfying that constraint. The optimal solution to a W-CSP is the solution that assigns those values that maximise the sum of the weights of satisfied constraints.

Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007, pp. 1124–1128,
Prague, June 2007.

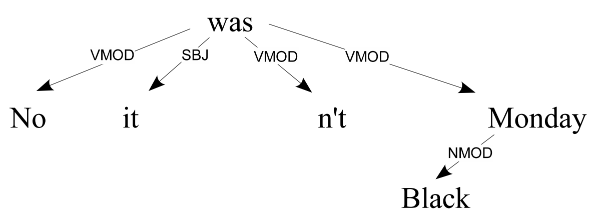

Figure 1: Dependency tree for the sentence *No it wasn't Black Monday*

To adapt this framework to predicting a dependency tree for a sentence, we construct a constraint satisfaction problem by first introducing one variable x_i for each token of the sentence. This variable's value corresponds to the dependency relation that token is the modifier of, i.e. it should specify a relation type and a head token. The constraints of the CSP are predicted by a classifier, where the weight for a constraint corresponds to the classifier's confidence estimate for the prediction.

For the current study, we trained three classifiers to predict three different types of constraints.

1. $C_{dep}(head, modifier, relation)$, i.e. the resulting dependency tree should have a dependency arc from *head* to *modifier* labelled with type *relation*. For the example tree in Figure 1, among others the constraint $C_{dep}(head{=}was, modifier{=}No, relation{=}VMOD)$ should be predicted.

2. $C_{dir}(modifier, direction)$, the relative position (i.e. to its left or to its right) of the head of *modifier*. The tree in Figure 1 will give rise to constraints such as $C_{dir}(modifier{=}Black, direction{=}RIGHT)$.

3. $C_{mod}(head, relation)$, in the dependency tree, *head* should be modified by a relation of type *relation*. The constraints generated for the word *was* in Figure 1 would be $C_{mod}(head{=}was, relations{=}SBJ)$, and $C_{mod}(head{=}was, relations{=}VMOD)$.

Predicting constraints of type C_{dep} is essentially what is done by Canisius et al. (2006a); a classifier is trained to predict a relation label, or a symbol signalling the absence of a relation, for each pair of tokens in a sentence[2]. The training data for this classifier consists of positive examples of constraints to generate, e.g. *was, No, VMOD*, and negative examples, of constraints *not* to generate, e.g. *was, Black, NONE*, but also *No, was, NONE*. In the aforementioned paper, it is shown that downsampling the negative class in the classifier's training data improves the recall for predicted constraints. The fact that improved recall comes at the cost of a reduced precision is compensated for by our choice for the weighted constraint satisfaction framework: an overpredicted constraint may still be left unsatisfied if other, conflicting constraints outweigh its own weight.

In addition to giving rise to a set of constraints, this classifier differs from the other two in the sense that it is also used to predict the domains of the variables, i.e. any dependency relation not predicted by this classifier will not be considered for inclusion in the output tree.

Whereas the C_{dep} classifier classifies instances for each pair of words, the classifiers for C_{dir} and C_{mod} only classify individual tokens. The features for these classifiers have been kept simple and the same for both classifiers: a 5-slot wide window of both tokens and part-of-speech tags, centred on the token currently being classified. The two classifiers differ in the classes they predict. For C_{dir}, there are only three possible classes: `LEFT`, `RIGHT`, `NONE`. Instances classified as `LEFT`, or `RIGHT` give rise to constraints, whereas `NONE` implies that no C_{dir} constraint is added for that token.

For C_{mod} there is a rather large class space; a class label reflects all modifying relations for the token, e.g. `SBJ+VMOD`. From this label, as many constraints are generated as there are different relation types in the label.

With the above, a weighted constraint satisfaction problem can be formulated that, when solved, describes a dependency tree. As we formulated our problem as a constraint satisfaction problem, any off-the-shelf W-CSP solver could be used to obtain the best dependency parse. However, in general such solvers have a time complexity exponential in the

[2] For reasons of efficiency and to avoid having too many negative instances in the training data, we follow the approach of Canisius et al. (2006a) of limiting the maximum distance between a potential head and modifier.

Language	LAS	'06	UAS	'06
Arabic	60.36	+1.2	78.61	+1.7
Basque	64.23	+1.1	72.24	+2.1
Catalan	77.33	+1.9	84.73	+3.1
Chinese	71.73	+1.3	77.29	+2.5
Czech	57.58	+1.4	75.61	+3.5
English	79.47	+2.2	81.05	+2.8
Greek	62.32	+2.0	76.42	+4.0
Hungarian	66.86	+2.6	72.52	+4.7
Italian	77.04	+1.5	81.24	+2.2
Turkish	67.80	-0.3	75.58	+0.4

Table 1: Performance of the system applied to the test data for each language. The '06 columns show the gain/loss with respect to the parser of Canisius et al. (2006a).

Language	'06	C_{dep}	$C_{dep}^{mod/}$	$C_{dep}^{dir/}$	all
Arabic	59.13	+0.3	+0.9	+0.9	+1.2
Basque	63.17	+0.3	+0.4	+0.9	+1.1
Catalan	75.44	+0.8	+1.2	+1.4	+1.9
Chinese	70.45	+0.4	+1.2	+0.4	+1.3
Czech	56.14	+0.5	+0.5	+1.1	+1.4
English	77.27	+0.4	+1.4	+1.2	+2.2
Greek	60.35	+0.4	+0.6	+1.6	+2.0
Hungarian	64.31	+1.9	+1.3	+2.8	+2.6
Italian	75.57	+0.2	+1.0	+1.1	+1.5
Turkish	68.09	-0.2	-0.3	-0.3	-0.3

Table 2: Performance of the parser by Canisius et al. (2006a) and the performance gain of the constraint satisfaction inference parser with various constraint configurations.

number of variables, and thus in the length of the sentence. As a more efficient alternative we chose to use the CKY algorithm for dependency parsing (Eisner, 2000) for computing the best solution, which has only cubic time complexity, but comes with the disadvantage of only considering projective trees as candidate solutions.

3 Results and discussion

We tested our system on all ten languages of the shared task. The three constraint classifiers have been implemented with memory-based learning. No language-specific parameter optimisation or feature engineering has been performed, but rather the exact same system has been applied to all languages. Labelled and unlabelled attachment scores are listed in Table 1. In addition, we show the increase/decrease in performance when compared with the parser of Canisius et al. (2006a); for all languages but Turkish, there is a consistent increase, mostly somewhere between 1.0 and 2.0 percent in labelled attachment score.

The parser by Canisius et al. (2006a) can be considered a rudimentary implementation of constraint satisfaction inference that only uses C_{dep} constraints. The parser described in this paper elaborates this by adding (1) the C_{mod} and C_{dir} *soft* constraints, and (2) projectivity and acyclicity *hard* constraints, enforced implicitly by the CKY algorithm.

To evaluate the effect of each of these constraints, Table 2 shows the labelled attachment scores for several parser configurations; starting with the 2006 parser, i.e. a parser with only C_{dep} constraints, then the CKY-driven C_{dep} parser, i.e. with acyclicity and projectivity constraints, then with C_{mod}, and C_{dir} separately, and finally, the full parser based on all constraints. It can be seen that supplementing the C_{dep}-only parser with hard constraints for acyclicity and projectivity already gives a small performance improvement. For some languages, such as Italian (+0.2), this improvement is rather small, however for Hungarian 1.9 is gained only by using CKY. The remaining columns show that adding more constraints improves performance, and that for all languages but Turkish and Hungarian, using all constraints works best.

While in comparison with the system of Canisius et al. (2006a) the addition of extra constraints has clearly shown its use, we expect the C_{dep} classifier still to be the performance bottleneck of the system. This is mainly due to the fact that this classifier is also responsible for defining the domains of the CSP variables, i.e. which dependency relations will be considered for inclusion in the output. For this reason, we performed an error analysis of the output of the C_{dep} classifier and the effect it has on the performance of the complete system.

In our error analysis, we distinguish three types of errors: 1) *label errors*, a correct dependency arc was added to the tree, but its label is incorrect, 2) *recall*

Language	C_{dep} prec.	C_{dep} rec.	prec. %OOD	rec. %OOD
Arabic	54.90	73.66	78.83	77.95
Basque	55.82	74.10	85.05	83.66
Catalan	65.19	87.25	80.29	80.00
Chinese	65.10	76.49	83.79	82.94
Czech	53.64	74.35	81.16	80.27
English	59.37	90.08	67.51	66.63
Greek	53.24	76.29	79.96	79.08
Hungarian	44.71	78.64	69.08	67.45
Italian	71.70	82.57	87.97	87.32
Turkish	64.92	72.79	89.11	88.51

Table 3: Columns two and three: precision and recall on dependency predictions by the C_{dep} classifier. Columns four and five: percentage of dependency arc precision and recall errors caused by out-of-domain errors.

errors, the true dependency tree contains an arc that is missing from the predicted tree, and 3) *precision errors*, the predicted tree contains a dependency arc that is not part of the true dependency parse.

Label errors are always a direct consequence of erroneous C_{dep} predictions. If the correct arc was predicted, but with an incorrect label, then by definition, the correct arc with the correct label cannot have been predicted at the same time. In case of the other two types of errors, the correct constraints may well have been predicted, but afterwards outweighed by other, conflicting constraints. Nevertheless, precision and recall errors may also be caused by the fact that the C_{dep} classifier simply did not predict a dependency arc where it should have. We will refer to those errors as out-of-domain errors, since the domain of at least one of the CSP variables does not contain the correct value. An out-of-domain error is a direct consequence of a recall error made by the C_{dep} classifier. To illustrate these interactions, Table 3 shows for all languages the precision and recall of the C_{dep} classifier, and the percentage of dependency precision and recall errors that are out-of-domain errors.

The table reveals several interesting facts. For English, which is the language for which our system attains its highest score, the percentage of dependency precision and recall errors caused by C_{dep} recall errors is the lowest of all languages. This can directly be related to the 90% recall of the English C_{dep} classifier. Apparently, the weak precision (59%), caused by down-sampling the training data, is compensated for in the subsequent constraint satisfaction process.

For Italian, the percentage of out-of-domain-related errors is much higher than for English. At the same time, the precision and recall of the C_{dep} classifier are much more in balance, i.e. a higher precision, but a lower recall. We tried breaking this balance in favour of a higher recall by applying an even stronger down-sampling of negative instances, and indeed the parser benefits from this. Labelled attachment increases from 77.04% to 78.41%. The precision and recall of this new C_{dep} classifier are 58.65% and 87.15%, respectively.

The lowest C_{dep} precision has been observed for Hungarian (44.71), which unfortunately is not mirrored by a high recall score. Remarkably however, after English, Hungarian has the lowest percentage of dependency errors due to C_{dep} recall errors (69.08 and 67.45). It is therefore hypothesised that not the low recall, but the low precision is the main cause for errors made on Hungarian. With this in mind, we briefly experimented with weaker down-sampling ratios in order to boost precision, but so far we did not manage to attain better results.

4 Concluding remarks

We have presented a novel dependency parsing method based on a standard constraint satisfaction framework. First results on a set of ten different languages have been promising, but so far no extensive optimisation has been performed, which inevitably reflects upon the scores attained by the system. Future work will focus on tuning the many parameters our system has, as well as on experimenting with different types of constraints to supplement or replace one or more of the three types used in this study.

Acknowledgements

The authors wish to thank Antal van den Bosch for discussions and suggestions. This research is funded by NWO, the Netherlands Organization for Scientific Research under the IMIX programme.

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.

S. Canisius, T. Bogers, A. van den Bosch, J. Geertzen, and E. Tjong Kim Sang. 2006a. Dependency parsing by inference over high-recall dependency predictions. In *Proceedings of CoNLL-X*. New York, NY, USA.

S. Canisius, A. van den Bosch, and W. Daelemans. 2006b. Constraint Satisfaction Inference: Non-probabilistic Global Inference for Sequence Labelling. *Proceedings of the EACL 2006 Workshop on Learning Structured Information in Natural Language Applications*, pages 9–16.

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.

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

J. Eisner. 2000. Bilexical grammars and their cubic-time parsing algorithms. *Advances in Probabilistic and Other Parsing Technologies*, pages 29–62.

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/.

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.

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 CoNLL 2007 Shared Task. Joint Conf. on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*.

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.