Multi-lingual Dependency Parsing with Incremental Integer Linear Programming

Sebastian Riedel and **Ruket Çakıcı** and **Ivan Meza-Ruiz**
ICCS
School of Informatics
University of Edinburgh
Edinburgh, EH8 9LW, UK
S.R.Riedel,R.Cakici,I.V.Meza-Ruiz@sms.ed.ac.uk

Abstract

Our approach to dependency parsing is based on the linear model of McDonald et al.(McDonald et al., 2005b). Instead of solving the linear model using the Maximum Spanning Tree algorithm we propose an incremental Integer Linear Programming formulation of the problem that allows us to enforce linguistic constraints. Our results show only marginal improvements over the non-constrained parser. In addition to the fact that many parses did not violate any constraints in the first place this can be attributed to three reasons: 1) the next best solution that fulfils the constraints yields equal or less accuracy, 2) noisy POS tags and 3) occasionally our inference algorithm was too slow and decoding timed out.

1 Introduction

This paper presents our submission for the CoNLL 2006 shared task of multilingual dependency parsing. Our parser is inspired by McDonald et al.(2005a) which treats the task as the search for the highest scoring Maximum Spanning Tree (MST) in a graph. This framework is efficient for both projective and non-projective parsing and provides an online learning algorithm which combined with a rich feature set creates state-of-the-art performance across multiple languages (McDonald and Pereira, 2006).

However, McDonald and Pereira (2006) mention the restrictive nature of this parsing algorithm. In their original framework, features are only defined over single attachment decisions. This leads to cases where basic linguistic constraints are not satisfied (e.g. verbs with two subjects). In this paper we present a novel way to implement the parsing algorithms for projective and non-projective parsing based on a more generic incremental Integer Linear Programming (ILP) approach. This allows us to include additional global constraints that can be used to impose linguistic information.

The rest of the paper is organised in the following way. First we give an overview of the Integer Linear Programming model and how we trained its parameters. We then describe our feature and constraint sets for the 12 different languages of the task (Hajič et al., 2004; Chen et al., 2003; Böhmová et al., 2003; Kromann, 2003; van der Beek et al., 2002; Brants et al., 2002; Kawata and Bartels, 2000; Afonso et al., 2002; Džeroski et al., 2006; Civit Torruella and Martí Antonín, 2002; Nilsson et al., 2005; Oflazer et al., 2003; Atalay et al., 2003). Finally, our results are discussed and error analyses for Chinese and Turkish are presented.

2 Model

Our model is based on the linear model presented in McDonald et al. (2005a),

$$(1)\quad s(\mathbf{x},\mathbf{y}) = \sum_{(i,j)\in\mathbf{y}} s(i,j) = \sum \mathbf{w}\cdot\mathbf{f}(i,j)$$

where $\mathbf{x}$ is a sentence, $\mathbf{y}$ a parse and s a score function over sentence-parse pairs. $\mathbf{f}(i,j)$ is a multidimensional feature vector representation of the edge from token i to token j and $\mathbf{w}$ the corresponding weight vector. Decoding in this model amounts to finding the $\mathbf{y}$ for a given $\mathbf{x}$ that maximises $s(\mathbf{x}, \mathbf{y})$

$$y' = argmax_y s(\mathbf{x}, \mathbf{y})$$

and $\mathbf{y}$ contains no cycles, attaches exactly one head to each non-root token and no head to the root node.

2.1 Decoding

Instead of using the MST algorithm (McDonald et al., 2005b) to maximise equation 1, we present an equivalent ILP formulation of the problem. An advantage of a general purpose inference technique is the addition of further linguistically motivated constraints. For instance, we can add constraints that enforce that a verb can not have more than one subject argument or that coordination arguments should have compatible types. Roth and Yih (2005) is similarly motivated and uses ILP to deal with additional hard constraints in a Conditional Random Field model for Semantic Role Labelling.

There are several explicit formulations of the MST problem as integer programs in the literature (Williams, 2002). They are based on the concept of eliminating subtours (cycles), cuts (disconnections) or requiring intervertex flows (paths). However, in practice these cause long solving times. While the first two types yield an exponential number of constraints, the latter one scales cubically but produces non-fractional solutions in its relaxed version, causing long runtime of the branch and bound algorithm. In practice solving models of this form did not converge after hours even for small sentences.

To get around this problem we followed an incremental approach akin to Warme (1998). Instead of adding constraints that forbid all possible cycles in advance (this would result in an exponential number of constraints) we first solve the problem without any cycle constraints. Only if the result contains cycles we add constraints that forbid these cycles and run the solver again. This process is repeated until no more violated constraints are found. Figure 1 shows this algorithm.

Groetschel et al. (1981) showed that such an approach will converge after a polynomial number of iterations with respect to the number of variables.

```
1.  Solve IP $P_i$
2.  Find violated constraints $C$ in the solution of $P_i$
3.  if $C = \emptyset$ we are done
4.  $P_{i+1} = P_i \cup C$
5.  $i = i + 1$
6.  goto (1)
```

Figure 1: Incremental Integer Linear Programming

In practice, this technique showed fast convergence (less than 10 iterations) in most cases, yielding solving times of less than 0.5 seconds. However, for some sentences in certain languages, such as Chinese or Swedish, an optimal solution could not be found after 500 iterations.

In the following section we present the bjective function, variables and linear constraints that make up the Integer Linear Program.

2.1.1 Variables

In the implementation[1] of McDonald et al. (2005b) dependency labels are handled by finding the best scoring label for a given token pair so that

$$s(i, j) = max\ s(i, j, label)$$

goes into Equation 1. This is only exact as long as no further constraints are added. Since our aim is to add constraints our variables need to explicitly model label decisions. Therefore, we introduce binary variables

$$l_{i,j,label} \forall i \in 0..n, j \in 1..n, label \in best_b(i, j)$$

where n is the number of tokens and the index 0 represents the root token. $best_b(i, j)$ is the set of b labels with maximal $s(i, j, label)$. $l_{i,j,label}$ equals 1 if there is a dependency with the label $label$ between token i (head) and j (child), 0 otherwise.

Furthermore, we introduce binary auxiliary variables

$$d_{i,j} \forall i \in 0..n, j \in 1..n$$

representing the existence of a dependency between tokens i and j. We connect these to the $l_{i,j,label}$ variables by a constraint

$$d_{i,j} = \sum_{label} l_{i,j,label}$$

.

[1]Note, however, that labelled parsing is not described in the publication.

2.1.2 Objective Function

Given the above variables our objective function can be represented as

$$\sum_{i,j} \sum_{label \in best_k(i,j)} s\left(i, j, label\right) \cdot l_{i,j,label}$$

with a suitable k.

2.1.3 Constraints Added in Advance

Only One Head In all our languages every token has exactly one head. This yields

$$\sum_{i>0} d_{i,j} = 1$$

for non-root tokens $j > 0$ and

$$\sum_{i} d_{i,0} = 0$$

for the artificial root node.

Typed Arity Constraints We might encounter solutions of the basic model that contain, for instance, verbs with two subjects. To forbid these we simply augment our model with constraints such as

$$\sum_{j} l_{i,j,subject} \leq 1$$

for all verbs i in a sentence.

2.1.4 Incremental Constraints

No Cycles If a solution contains one or more cycles C we add the following constraints to our IP: For every $c \in C$ we add

$$\sum_{(i,j)\in c} d_{i,j} \leq |c| - 1$$

to forbid c.

Coordination Argument Constraints In coordination conjuncts have to be of compatible types. For example, nouns can not coordinate with verbs. We implemented this constraint by checking the parses for occurrences of incompatible arguments. If we find two arguments j, k for a conjunction i: $d_{i,j}$ and $d_{i,k}$ and j is a noun and k is a verb then we add

$$d_{i,j} + d_{i,k} \leq 1$$

to forbid configurations in which both dependencies are active.

Projective Parsing In the incremental ILP framework projective parsing can be easily implemented by checking for crossing dependencies after each iteration and forbidding them in the next. If we see two dependencies that cross, $d_{i,j}$ and $d_{k,l}$, we add the constraint

$$d_{i,j} + d_{k,l} \leq 1$$

to prevent this in the next iteration. This can also be used to prevent specific types of crossings. For instance, in Dutch we could only allow crossing dependencies as long as none of the dependencies is a "Determiner" relation.

2.2 Training

We used single-best MIRA(Crammer and Singer, 2003).For all experiments we used 10 training iterations and non-projective decoding. Note that we used the original spanning tree algorithm for decoding during training as it was faster.

3 System Summary

We use four different feature sets. The first feature set, BASELINE, is taken from McDonald and Pereira (2005b). It uses the *FORM* and the *POSTAG* fields. This set also includes features that combine the label and POS tag of head and child such as $(Label, POS_{Head})$ and $(Label, POS_{Child-1})$. For our Arabic and Japanese development sets we obtained the best results with this configuration. We also use this configuration for Chinese, German and Portuguese because training with other configurations took too much time (more than 7 days).

The BASELINE also uses pseudo-coarse-POS tag ($1st$ character of the *POSTAG*) and pseudo-lemma tag (4 characters of the *FORM* when the length is more than 3). For the next configuration we substitute these pseudo-tags by the *CPOSTAG* and *LEMMA* fields that were given in the data. This configuration was used for Czech because for other configurations training could not be finished in time.

The third feature set tries to exploit the generic *FEATS* field, which can contain a list features such as case and gender. A set of features per dependency is extracted using this information. It consists of cross product of the features in *FEATS*. We used this configuration for Danish, Dutch, Spanish and Turkish where it showed the best results during development.

The fourth feature set uses the triplet of label, *POS* child and head as a feature such as $(Label, POS_{Head}, POS_{Child})$. It also uses the *CPOSTAG* and *LEMMA* fields for the head. This configuration is used for Slovene and Swedish data where it performed best during development.

Finally, we add constraints for Chinese, Dutch, Japanese and Slovene. In particular, arity constraints to Chinese and Slovene, coordination and arity constraints to Dutch, arity and selective projectivity constraints for Japanese[2]. For all experiments b was set to 2. We did not apply additional constraints to any other languages due to lack of time.

4 Results

Our results on the test set are shown in Table 1. Our results are well above the average for all languages but Czech. For Chinese we perform significantly better than all other participants ($p = 0.00$) and we are in the top three entries for Dutch, German, Danish. Although Dutch and Chinese are languages were we included additional constraints, our scores are not a result of these. Table 2 compares the result for the languages with additional constraints. Adding constraints only marginally helps to improve the system (in the case of Slovene a bug in our implementation even degraded accuracy). A more detailed explanation to this observation is given in the following section. A possible explanation for our high accuracy in Chinese could be the fact that we were not able to optimise the feature set on the development set (see the previous section). Maybe this prevented us from overfitting. It should be noted that we did use non-projective parsing for Chinese, although the corpus was fully projective. Our worst results in comparison with other participants can be seen for Czech. We attribute this to the reduced training set we had to use in order to produce a model in time, even when using the original MST algorithm.

[2]This is done in order to capture the fact that crossing dependencies in Japanese could only be introduced through disfluencies.

4.1 Chinese

For Chinese the parser was augmented with a set of constraints that disallowed more than one argument of the types *head, goal, nominal, range, theme, reason, DUMMY, DUMMY1* and *DUMMY2*.

By enforcing arity constraints we could either turn wrong labels/heads into right ones and improve accuracy or turn right labels/heads into wrong ones and degrade accuracy. For the test set the number of improvements (36) was higher than the number of errors (22). However, this margin was outweighed by a few sentences we could not properly process because our inference method timed out. Our overall improvement was thus unimpressive 7 tokens.

In the context of duplicate "head" dependencies (that is, dependencies labelled "head") the number of sentences where accuracy dropped far outweighed the number of sentences where improvements could be gained. Removing the arity constraints on "head" labels therefore should improve our results.

This shows the importance of good second best dependencies. If the dependency with the second highest score is the actual gold dependency and its score is close to the highest score, we are likely to pick this dependency in the presence of additional constraints. On the other hand, if the dependency with the second highest score is not the gold one and its score is too high, we will probably include this dependency in order to fulfil the constraints.

There may be some further improvement to be gained if we train our model using k-best MIRA with $k > 1$ since it optimises weights with respect to the k best parses.

4.2 Turkish

There is a considerable gap between the unlabelled and labelled results for Turkish. And in terms of labels the POS type *Noun* gives the worst performance because many times a subject was classified as object or vice a versa.

Case information in Turkish assigns argument roles for nouns by marking different semantic roles. Many errors in the Turkish data might have been caused by the fact that this information was not adequately used. Instead of fine-tuning our feature set to Turkish we used the feature cross product as de-

Model	AR	CH	CZ	DA	DU	GE	JP	PO	SL	SP	SW	TU
OURS	66.65	89.96	67.64	83.63	78.59	86.24	90.51	84.43	71.20	77.38	80.66	58.61
AVG	59.94	78.32	67.17	78.31	70.73	78.58	85.86	80.63	65.16	73.53	76.44	55.95
TOP	66.91	89.96	80.18	84.79	79.19	87.34	91.65	87.60	73.44	82.25	84.58	65.68

Table 1: Labelled accuracy on the test sets.

Constraints	DU	CH	SL	JA
with	3927	4464	3612	4526
without	3928	4471	3563	4528

Table 2: Number of tokens correctly classified with and without constraints.

scribed in Section 3. Some of the rather meaningless combinations might have neutralised the effect of sensible ones. We believe that using morphological case information in a sound way would improve both the unlabelled and the labelled dependencies. However, we have not performed a separate experiment to test if using the case information alone would improve the system any better. This could be the focus of future work.

5 Conclusion

In this work we presented a novel way of solving the linear model of McDonald et al. (2005a) for projective and non-projective parsing based on an incremental ILP approach. This allowed us to include additional linguistics constraints such as “a verb can only have one subject.”

Due to time constraints we applied additional constraints to only four languages. For each one we gained better results than the baseline without constraints, however, this improvement was only marginal. This can be attributed to 4 main reasons: Firstly, the next best solution that fulfils the constraints was even worse (Chinese). Secondly, noisy POS tags caused coordination constraints to fail (Dutch). Thirdly, inference timed out (Chinese) and fourthly, constraints were not violated that often in the first place (Japanese).

However, the effect of the first problem might be reduced by training with a higher k. The second problem could partly be overcome by using a better tagger or by a special treatment within the constraint handling for word types which are likely to be mistagged. The third problem could be avoidable by adding constraints during the branch and bound algorithm, avoiding the need to resolve the full problem “from scratch” for every constraint added. With these remedies significant improvements to the accuracy for some languages might be possible.

6 Acknowledgements

We would like to thank Beata Kouchnir, Abhishek Arun and James Clarke for their help during the course of this project.

References

Koby Crammer and Yoram Singer. 2003. Ultraconservative online algorithms for multiclass problems. *J. Mach. Learn. Res.*, 3:951–991.

M. Groetschel, L. Lovasz, and A. Schrijver. 1981. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, I:169– 197.

R. McDonald and F. Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *Proc. of the 11th Annual Meeting of the EACL*.

R. McDonald, K. Crammer, and F. Pereira. 2005a. Online large-margin training of dependency parsers. In *Proc. of the 43rd Annual Meeting of the ACL*.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajic. 2005b. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of HLT/EMNLP 2005*, Vancouver, B.C., Canada.

D. Roth and W. Yih. 2005. Integer linear programming inference for conditional random fields. In *Proc. of the International Conference on Machine Learning (ICML)*, pages 737–744.

David Michael Warme. 1998. *Spanning Trees in Hypergraphs with Application to Steiner Trees*. Ph.D. thesis, University of Virginia.

Justin C. Williams. 2002. A linear-size zero - one programming model for the minimum spanning tree problem in planar graphs. *Networks*, 39:53–60.