Dependency Relations Labeller for Czech

Rudolf Rosa and David Mareček

Charles University in Prague, Faculty of Mathematics and Physics
Institute of Formal and Applied Linguistics, Prague, Czech Republic,
{rosa, marecek}@ufal.mff.cuni.cz,
WWW home page: http://ufal.mff.cuni.cz

Abstract. We present a MIRA-based labeller designed to assign dependency relation labels to edges in a dependency parse tree, tuned for Czech language. The labeller was created to be used as a second stage to unlabelled dependency parsers but can also improve output from labelled dependency parsers. We evaluate two existing techniques which can be used for labelling and experiment with combining them together. We describe the feature set used. Our final setup significantly outperforms the best results from the CoNLL 2009 shared task.

Keywords: natural language processing, dependency parsing, sequence labelling

1 Introduction

Dependency trees have for a long time been the primary structure to represent Czech syntax, as well as in other, especially highly non-projective, languages. In recent years, dependency trees have proven to be a valuable alternative to phrase-structure trees even for relatively projective languages, and much effort has been invested into creating various dependency parsers, such as the MST Parser [1] or the MALT Parser [2].

As opposed to phrase-structure trees, assigning a correct label to each dependency relation is often an important task as well. Some parsers perform joint parsing and labelling, producing a labelled dependency tree as their output. Others produce only an unlabelled tree, requiring a standalone labeller to assign the labels in a second step. We present such a labeller in our paper.

In Section 2, we review two existing techniques which can be used to assign labels to dependency relations and try to combine them together. Section 3 contains a description of our feature set. We evaluate our results in Section 4.

2 Labelling Methods

We make use of several algorithms that can be used and combined in various ways in the task of dependency relations labelling, building upon [3] and [4].

A *labelling* $\mathcal{L}$ of a dependency tree is a mapping $\mathcal{L} : E \rightarrow L$, which assigns a label $l \in L$ to each edge $e \in E$ of the dependency tree. The general approach

that we follow to find the best labelling is using edge-based factorization, i.e. to assign a score to each possible pair $[e, l]$, and to try to find a labelling with a maximum overall score. Based on the scoring method used, the score can be a probability or an additive score.

Following [3], we treat the dependency relations labelling as a *sequence labelling problem*. Starting at the root node of the tree, we label all edges going from the current node to its children from left to right (these adjacent edges form a sequence), and then continue in the same way on lower levels of the tree. This implies that at each step we have already processed all ancestor edges and left sibling edges. We utilize this fact in designing the feature set.

To label the sequence, we use the well-known Viterbi algorithm, as described in [3].

Let $\boldsymbol{e} = (e_1, e_2, ..., e_n)$ be a sequence of edges and $\boldsymbol{l} = (l_1, l_2, ..., l_n)$ one of its possible labellings. The score $S_{\boldsymbol{e},\boldsymbol{l}}$ of $\boldsymbol{e}$ being labelled by $\boldsymbol{l}$, which Viterbi tries to maximize, is defined differently for additive scores and for probabilities. For additive scores, it is computed as a sum:

$$S_{\boldsymbol{e},\boldsymbol{l}} = \sum_{i=1}^{n} score(e_i, l_i) \tag{1}$$

whereas for probabilities, it is computed as a product:

$$S_{\boldsymbol{e},\boldsymbol{l}} = \prod_{i=1}^{n} p(e_i, l_i) \tag{2}$$

We use n-best Viterbi, which always keeps n best partial labellings at each step, selecting the best scoring labelling at the end.

2.1 Maximum Likelihood Estimate

We use Maximum Likelihood Estimate (MLE) as a simple baseline method, trying to estimate the probability distribution of labels that can be assigned to an edge, based on its features and the previously assigned labels.

Suppose that we want to find the probability of an edge e with binary features F to be labelled with the label l from the set of all labels L.

We use MLE to estimate the probability distribution of various labels that can be assigned to an edge, based on its features.

We estimate the *emission probability* as an average probability of an edge being labelled by label l given the set of the features F of the edge:

$$p_{em}(l|F) = \frac{1}{|F|} \sum_{f \in F} p_{em}(l|f) \text{ where } p_{em}(l|f) = \frac{count(f \wedge l)}{count(f)} \tag{3}$$

where $count()$ refers to number of occurrences of the feature f or pair of the feature f and the label l in the training data. The averaging can be regarded as lambda smoothing with all lambdas equal to $1/|F|$.

We estimate the *transition probability* as a conditional probability of a label l given the previous label l_{prev}, smoothed together with marginal probability of the label l and a constant parameter:

$$p_{tr}(l|l_{prev}) = \lambda_2 \cdot p_{bi}(l|l_{prev}) + \lambda_1 \cdot p_{uni}(l) + \lambda_0 \tag{4}$$

where

$$p_{bi}(l|l_{prev}) = \frac{count(l \wedge l_{prev})}{count(l_{prev})} \text{ and } p_{uni}(l) = \frac{count(l)}{|L|} \tag{5}$$

The smoothing parameters λ_2, λ_1 and λ_0 are estimated using the Expectation Maximization (EM) algorithm on held-out data.

MLE is very fast and also very easy to implement. However, it did not perform well enough.

2.2 Margin Infused Relaxed Algorithm

Following [4] and [3], we decided to use the Margin Infused Relaxed Algorithm (MIRA), described in [5]. MIRA is an online learning algorithm for large-margin multiclass classification, successfully used in [1] for dependency parsing and suggested to be used for dependency relations labelling in a second stage labeller.

MIRA assigns a score for each label that can be assigned to a dependency relation. Feature-based factorization is used, thus the score of a label l to be assigned to an edge e which has the features F is computed as follows:

To find the probability of an edge e with features F to be labelled with the label l from the set of all labels L, we compute a *score*:

$$score(l, e) = \sum_{f \in F} score(l, f) \tag{6}$$

where $score(l, f)$, also called the *weight* of the feature (l, f), is computed by MIRA, trying to minimize the classification error on the training data, iterating over the whole dataset several times. The final scores are then averaged to avoid overtraining.

We used the *single-best MIRA* variant in our experiments.

2.3 MLE & MIRA

Having already implemented both MLE and MIRA, we also tried to combine them together. We have observed that the emission probabilities are crucial for assigning the correct label, whereas the transition probabilities have a rather small impact. We therefore decided to use MLE to estimate transition probabilities the same way as in MLE approach, but to use MIRA to estimate emission scores instead of MLE emission probabilities (this time not using features based on left sibling's label). However, a major issue with this approach is to combine the transition probabilities and emission scores correctly.

Product. Our first attempt was to simply use

$$score(l, e) = p_{tr}(l|l_{prev}) \cdot score_{MIRA}(l, e) \tag{7}$$

as in MLE approach, and to sum these together in Viterbi as in MIRA approach. Although being severely mathematically incorrect (there are grave issues with ordering of the scores and even greater issues with negative scores), this approach did lead to competitive results.

Sigmoid. We then tried to use a mathematically sound way by recomputing the emission score into a probability-like number using the sigmoid function:

$$score(l, e) = p_{tr}(l|l_{prev}) \cdot \frac{1}{1 + e^{-score_{MIRA}(l,e)}} \tag{8}$$

Interestingly, the results of this approach were significantly worse than these of the Product approach. However, as the MIRA-only method proved to outperform both of these combination methods, we did not investigate this further.

3 Feature Set

We construct the feature set by combining features suggested in [1], [3], [6] and [7] and tuning it to maximize performance on Czech data.

Our input for labelling is the parsed sentence, together with its morphological analysis – i.e., for each word, we know its form, lemma and morphological tag. We use coarse Czech tags as described in [8].

We join these fields (and feature functions defined in following sections) in various ways to form feature templates for the labeller. For example, feature template that consists of the parent and child word forms and coarse tag of the child is defined as "`FORM|form|coarse_tag`". We denote a field of the parent node of the edge by using uppercase; the child node is indicated by using lowercase.

Each of the feature values is then joined with its feature template and the label of the edge.

For example, for an edge "Rudolf/N1 ← relaxuje/VB" (Rudolph ← relaxes) with a label "Sb" (subject), the feature template "`FORM|form|coarse_tag`" would return the value "`FORM|form|coarse_tag:`*`relaxuje|Rudolf|N1`*`:Sb`".

Most of the feature templates are created by joining one or more fields or feature functions with the `COARSE_TAG` and `coarse_tag` fields, as this has been observed to lead to the best results. For instance, feature templates incorporating *attachment direction* feature function `attrdir()` have the following shapes:

- `attdir()|coarse_tag`,
- `attdir()|COARSE_TAG`,
- `attdir()|coarse_tag|COARSE_TAG`

Also the feature functions that can take a field as a parameter are usually declared with `coarse_tag` being the parameter. The complete feature set is available in the TectoMT Share repository.[1]

Thanks to the fact that we get the whole dependency tree on input, we do not have to limit ourselves to local features computed over the edge; we can easily include information about any nodes in the tree.

3.1 First-order Features

Our basic feature set is based on the first order features described in [1] (first-order features are defined as features on single edges), which were designed for unlabelled parsing but proved to be useful for labelling as well. However, we do not conjoin our feature templates with distance of words or attachment direction, as these are more relevant for parsing than for labelling. Also we use morphological lemmas instead of 5-gram prefixes. The combinations of features were modified a little to maximize performance on development data.

The first-order feature templates consist of the basic fields available on input, and of several context features, providing information about nearby words.

- `COARSE_TAG` / `coarse_tag`
- `FORM` / `form`
- `LEMMA` / `lemma`
- `PRECEDING(`*`field`*`)` / `preceding(`*`field`*`)` – a field of the word immediately preceding the parent/child node, e.g. `PRECEDING(coarse_tag)` – the coarse tag of the word which precedes the parent node of the edge
- `FOLLOWING(`*`field`*`)` / `following(`*`field`*`)` – a field of the word immediately following the parent/child node
- `between(`*`field`*`)` – a field of each of the words between the parent node and the child node; this function can return multiple values, creating several features from one feature template
- `attdir()` – the direction of attachment of the child to the parent

3.2 Non-local Features

Based on non-local features described in [3], we extend our feature set with labelling-specific features, which make use of knowledge of potentially the whole parse tree:

- `CHILDNO()` / `childno()` – number of child nodes of the parent/child node
- `isfirstchild()` / `islastchild()` – whether the child node is the first/last child of the parent node

[1] http://ufallab.ms.mff.cuni.cz/tectomt/share/data/models/labeller_mira/cs/pdt_labeller_best.config

3.3 Higher-order Features

Higher order features are based on multiple edges. These can be sibling features as described in [6], parent-child-grandchild features as described in [7], or other variations and conjunctions of these concepts.

The possibility to use some of the features depends on the order of assigning labels to edges in the dependency tree. Following [3], we express the task as a sequence labelling problem, with a sequence defined as a sequence of adjacent edges, i.e. edges sharing the same parent node. This enables us to use the information about the label assigned to sibling edges which are on the left from the current edge.

Furthermore, we decided to go through the tree in a top-down direction, i.e. first labelling the sequence of edges whose parent is the root node, continuing to their children. Thus we can also make use of the knowledge of the label assigned to the edge between the parent and the grandparent of the child node, and we even make use of the label assigned to the edge between the grandparent and the great-grandparent.

The grandparent features belong to features with the biggest influence on accuracy. Their inclusion in the feature set immediately led to an improvement of accuracy by 2%, whereas most of the other features contribute with less than 0.5%. They are especially useful for e.g. prepositional and coordination structures.

- `LABEL()` – label assigned to the edge between the parent and the grandparent of the child node
- `l.label()` – label assigned to the left sibling edge, i.e. the edge between the parent and the left sibling of the child node
- `r.coarse_tag` – coarse tag of the right sibling of the child node
- `G.coarse_tag` – coarse tag of the grandparent of the child node
- `G.label()` – label assigned to the edge between the grandparent and the great-grandparent of the child node
- `G.attdir()` – whether the grandparent node precedes or follows the child node in the sentence

4 Experiments and Results

We use the Prague Dependency Treebank 2.0 [9] for training and testing. We use 68500 sentences as training data, 4500 sentences as development data and 4500 sentences as test data.

In Table 1, we present a comparison of the methods that we tried to use for labelling. We measure labelling accuracy on golden trees. MIRA clearly outperforms all the other methods.

To reliably compare our results with results of other existing labelled parsers, we used the datasets from the CoNLL 2009 Shared Task [10]. On the website[2] of

[2] http://ufal.mff.cuni.cz/conll2009-st/

Table 1. Comparison of several labelling methods. The results obtained on the test-data golden trees using the feature set tuned on the developement data.

labelling method	labelling accuracy [%]
MLE	65.7
MLE + MIRA (sigmoid)	89.0
MLE + MIRA (product)	91.9
MIRA	94.1

the task, the training and test data are available, together with outputs of the participating systems.

We decided to compare our results with the four best-performing systems. We trained our labeller on the official training data and then used it to relabel the outputs of the four systems. Labelled Attachment Score (LAS) was then measured by comparing the results with the golden test data.

The results in Table 2 show that, although we did not improve LAS of each of the systems, we managed to improve LAS of the *che* system [11] by 1.2%, leading to a LAS of 81.2%, which significantly outperforms all of the four original systems, leading to a state-of-the-art labelling – that is, at least on this task.

Manual inspection of the original and the relabelled outputs of the *che* system showed that most of the differences are in differentiating between the Attribute, Adverbial and Object, and between the Object and Subject.

Table 2. Labelled Attachment Scores on CoNLL2009 shared task best systems outputs

system	LAS original [%]	LAS relabelled [%]
merlo	**80.4**	79.9
bohnet	80.1	81.0
che	80.0	**81.2**
chen	79.7	79.5

5 Conclusion

We created a labeller which provides to our knowledge state-of-the-art dependency relations labelling for Czech language. It is a standalone labeller and therefore can be used as a second-stage to any unlabelled dependency parser. Moreover, it might even improve the accuracy of a labelled parser by relabelling its output. It uses the Margin Infused Relaxed Algorithm, which makes it possible to use a large number of features, both first- and higher-order.

In future we are planning to experiment with using labels assigned by another labelled parser as input features for the labeller, which might lead to even better results.

It would be favourable to reduce the feature set and prune the model if possible to achieve lower time and memory consumption with similar accuracy. We leave this for our future work as well.

The labeller is published on CPAN[3] as a part of the MSTperl package. Various trained models for the labeller can be obtained in the TectoMT Share repository.[4]

Acknowledgements. This research has been supported by the EU Seventh Framework Programme under grant agreement n° 247762 (Faust).

References

1. McDonald, R., Crammer, K., Pereira, F.: Online large-margin training of dependency parsers. In: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics, Association for Computational Linguistics (2005) 91–98
2. Nivre, J., Hall, J., Nilsson, J.: Maltparser: A data-driven parser-generator for dependency parsing. In: Proceedings of LREC. Volume 6. (2006) 2216–2219
3. McDonald, R., Lerman, K., Pereira, F.: Multilingual dependency analysis with a two-stage discriminative parser. In: Proceedings of the Tenth Conference on Computational Natural Language Learning. CoNLL-X '06, Stroudsburg, PA, USA, Association for Computational Linguistics (2006) 216–220
4. Rosenfeld, B., Feldman, R., Fresko, M.: A systematic cross-comparison of sequence classifiers. SDM 2006 (2006) 563–567
5. Crammer, K., Singer, Y.: Ultraconservative online algorithms for multiclass problems. The Journal of Machine Learning Research **3** (2003) 951–991
6. McDonald, R., Pereira, F.: Online learning of approximate dependency parsing algorithms. In: Proceedings of EACL. Volume 6. (2006) 81–88
7. Carreras, X.: Experiments with a higher-order projective dependency parser. In: Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL. Volume 7. (2007) 957–961
8. Collins, M., Ramshaw, L., Hajič, J., Tillmann, C.: A statistical parser for Czech. In: Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics. ACL '99, Stroudsburg, PA, USA, Association for Computational Linguistics (1999) 505–512
9. Hajič, J., Hajičová, E., Panevová, J., Sgall, P., Pajas, P., Štěpánek, J., Havelka, J., Mikulová, M.: Prague Dependency Treebank 2.0. CD-ROM, Linguistic Data Consortium, LDC Catalog No.: LDC2006T0 1, Philadelphia (2006)
10. Hajič, J., Ciaramita, M., Johansson, R., Kawahara, D., Martí, M., Màrquez, L., Meyers, A., Nivre, J., Padó, S., Štěpánek, J., et al.: The CoNLL-2009 shared task: Syntactic and semantic dependencies in multiple languages. In: Proceedings of the Thirteenth Conference on Computational Natural Language Learning: Shared Task, Association for Computational Linguistics (2009) 1–18
11. Chen, W., Zhang, Y., Isahara, H.: A two-stage parser for multilingual dependency parsing. In: Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL. (2007) 1129–1133

[3] http://www.cpan.org/

[4] http://ufallab.ms.mff.cuni.cz/tectomt/share/data/models/labeller_mira/cs/