UN PARSER A DIPENDENZE PER L'ITALIANO

A DEPENDENCY PARSER FOR ITALIAN

MICHAEL SCHIEHLEN

SOMMARIO/*ABSTRACT*

Questo articolo descrive gli adattamenti apportati a un parser a dipendenze multilingue per adeguarlo all'italiano e al task di Parsing di Evalita; descrive inoltre l'approccio adottato, le feature usate e l'algoritmo di apprendimento.

The paper describes adaptations to a multilingual dependency parser for Italian and the EVALITA parsing task, such as determination and linking of empty nodes. The paper also presents the parsing approach, features used, and the training algorithm.

Keywords: Dependency Parsing, Structured Prediction, Online Learning.

1 The Parser

Our parser is modelled after McDonald's non–projective parser [2]. The parser computes all possible projective dependency trees and chooses the one receiving the maximum weight. Weights are determined in repeated iterations over training data. A difference is that labelling is not relegated to a postprocessing step, but performed during parsing proper [5]. We used the same parser in the shared task of CoNLL'07, but have now integrated some insights gained in working on the Italian track of CoNLL'07. In particular, we switched off the second–order features [3]. The parser is explained below, but more details about the parser can be found in [5].

1.1 Parsing Algorithm

For parsing, we adopt Eisner's bottom-up chart-parsing algorithm in McDonald's [2] 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 $\mathbf{\Phi}_{ij}$ with associated weights $\boldsymbol{w}_{ij}$. Eisner's algorithm achieves optimal tree packing by storing partial structures in two matrices ⌂ and ◺. 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.

$$\begin{aligned}
{}^{i}\searrow_{j} &= \boldsymbol{w}_{ij} \cdot \mathbf{\Phi}_{ij} \\
{}_{i}\text{⌂}_{j} &= \max_{k:i\leq k<j} {}_{i}\text{◺}_{k} + {}_{k+1}\text{◿}_{j} + {}^{i}\searrow_{j} \qquad (1)\\
{}_{i}\text{◺}_{j} &= \max_{k:i<k\leq j} {}_{i}\text{⌂}_{k} + {}_{k}\text{◺}_{j} \qquad (2)\\
\text{root} &= \max_{k:1\leq k\leq n} {}_{1}\text{◿}_{k} + {}_{k}\text{◺}_{n} + \boldsymbol{w}_{0k} \cdot \mathbf{\Phi}_{0k}
\end{aligned}$$

1.2 Feature Representation

In deriving features, we used all information given in the treebank, 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 [2]: $\phi'_{ij} = \{w_i, fp_i, lcp_i, w_j, fp_j, lcp_j, w_i w_j, w_i lcp_j, lcp_i w_j, lcp_i lcp_j, fp_i lcp_j, fp_i fp_j, fp_i fp_j agr_{ij}, fp_{i-1} fp_i fp_{j-1} fp_j, fp_{i-1} fp_i fp_j fp_{j+1}, fp_i fp_{i+1} fp_{j-1} fp_j, fp_i fp_{i+1} fp_j fp_{j+1}\}$, and token features for root words $\phi_{0r} = \{w_r, fp_r, lcp_r\}$. Furthermore, we recorded the tag of each token m between i and j ($\phi_{ij} = \phi'_{ij} \cup \{fp_i fp_j fp_m\}$). All features but unary token features were optionally extended with direction of dependency ($i < j$ or $i > j$) and binned token distance ($|i - j| = 1, 2, 3, 4, \geq 5, \geq 10$).

1.3 Structural Learning

For determining feature weights $\boldsymbol{w}$, we used online passive–aggressive learning (OPAL) [1]. 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

[1]Agreement was computed from morphological features, viz. gender, number and person. We also added a nominative case feature to finite verbs.

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

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

Having a closed–form solution, OPAL is easier to implement and more efficient than the MIRA algorithm [2], while still achieving a performance comparable to MIRA's on many problems [1].

1.4 Learning Labels for Dependency Relations

To derive dependency relation (deprel) labels, we split each feature along the label dimension, associating each deprel d with its own feature vector (cf. eq. (3), where $\otimes$ is the tensor product and Λ^o the orthogonal encoding).

$$\Phi_{ij,\Lambda^o(d)} = \phi_{ij} \otimes \Lambda^o(d) \qquad (3)$$

Parsing only considers the best deprel label.

$$^i\searrow_j = \max_{d\in\mathcal{D}} w_{ij,\Lambda^o(d)}\Phi_{ij,\Lambda^o(d)} \qquad (4)$$

More details can be found in [5].

1.5 Adaptations to the EVALITA Parsing Task

In EVALITA, the task was somewhat different from that in CoNLL in that not only dependency structure and labels should be learned, but also the distribution of empty nodes. In order to find empty nodes, we first parsed the input without any empty nodes, and then applied a maximum entropy learner (Zhang Le's MaxEnt toolkit) to predict for each word whether or not an empty node should be inserted after it. For this decision, we used information specific to the respective word (e.g. word form, lemma, fine grained POS tag, coarse grained POS tag), and word–specific information sculled from the initial parse (e.g. deprel, deprel+POS tag combination, all the deprels of daughter nodes). In this way, the system could detect e.g. missing subjects. After this step, empty nodes were inserted and a separate parsing model was trained on this new input set.

2 Treebank Transformation

We applied our own conversion tool to derive CoNLL'07 format from the original TUT representation format. First, the input was converted to UTF 8. Then the information on each token line was distributed among the CoNLL slots (word form, lemma, fine-grained POS tag, coarse-grained POS tag, morphological features). Empty nodes were treated in this step as nodes of their own, independent of other nodes. Finally, information on the POS tag of dependent and head was discarded in the deprel slot, as the feature representation used in our parser is rich enough to account for such interactions.

REFERENCES

[1] K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Yoram Singer. Online Passive–Aggressive Algorithms. *Journal of Machine Learning*, 7:551–585, 2006.

[2] R. McDonald, K. Crammer, and F. Pereira. Online Large-Margin Training of Dependency Parsers. In *ACL'05*, 2005.

[3] R. McDonald and F. Pereira. Online Learning of Approximate Dependency Parsing Algorithms. In *EACL'06*, Trento, Italy, 2006.

[4] J. Nivre, J. Hall, S. Kübler, R. McDonald, J. Nilsson, S. Riedel, and D. Yuret. The CoNLL 2007 Shared Task on Dependency Parsing. In *EMNLP'07*, Praha, 2007.

[5] M. Schiehlen and K. Spranger. Global Learning of Labelled Dependency Trees. In *EMNLP'07*, Praha, 2007.

[6] I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. Support Vector Machine Learning for Interdependent and Structured Output Spaces. In *Proceedings of ICML'04*, 2004.

CONTACT

MICHAEL SCHIEHLEN
Institute for Natural Language Processing
University of Stuttgart
Azenbergstraße 12
D-70174 Stuttgart
Germany
Email: Michael.Schiehlen@ims.uni-stuttgart.de

DR. MICHAEL SCHIEHLEN is assistant professor at the IMS. His research interests are focussed on computational semantics and machine learning, but also extend to linguistic processing in general.