Labeled Pseudo-Projective Dependency Parsing with Support Vector Machines

Joakim Nivre
Johan Hall
Jens Nilsson
School of Mathematics
and Systems Engineering
Växjö University
35195 Växjö, Sweden
{nivre,jha,jni}@msi.vxu.se

Gülşen Eryiğit
Department of
Computer Engineering
Istanbul Technical University
34469 Istanbul, Turkey
gulsen@cs.itu.edu.tr

Svetoslav Marinov
School of Humanities
and Informatics
University of Skövde
Box 408
54128 Skövde, Sweden
svetoslav.marinov@his.se

Abstract

We use SVM classifiers to predict the next action of a deterministic parser that builds labeled projective dependency graphs in an incremental fashion. Non-projective dependencies are captured indirectly by projectivizing the training data for the classifiers and applying an inverse transformation to the output of the parser. We present evaluation results and an error analysis focusing on Swedish and Turkish.

1 Introduction

The CoNLL-X shared task consists in parsing texts in multiple languages using a single dependency parser that has the capacity to learn from treebank data. Our methodology for performing this task is based on four essential components:

- A deterministic algorithm for building labeled projective dependency graphs (Nivre, 2006).
- History-based feature models for predicting the next parser action (Black et al., 1992).
- Support vector machines for mapping histories to parser actions (Kudo and Matsumoto, 2002).
- Graph transformations for recovering non-projective structures (Nivre and Nilsson, 2005).

All experiments have been performed using MaltParser (Nivre et al., 2006), version 0.4, which is made available together with the suite of programs used for pre- and post-processing.[1]

[1] www.msi.vxu.se/users/nivre/research/MaltParser.html

2 Parsing Methodology

2.1 Parsing Algorithm

The parsing algorithm used for all languages is the deterministic algorithm first proposed for unlabeled dependency parsing by Nivre (2003) and extended to labeled dependency parsing by Nivre et al. (2004). The algorithm builds a labeled dependency graph in one left-to-right pass over the input, using a stack to store partially processed tokens and adding arcs using four elementary actions (where *top* is the token on top of the stack and *next* is the next token):

- SHIFT: Push *next* onto the stack.
- REDUCE: Pop the stack.
- RIGHT-ARC(r): Add an arc labeled r from *top* to *next*; push *next* onto the stack.
- LEFT-ARC(r): Add an arc labeled r from *next* to *top*; pop the stack.

Although the parser only derives projective graphs, the fact that graphs are labeled allows non-projective dependencies to be captured using the pseudo-projective approach of Nivre and Nilsson (2005) .

Another limitation of the parsing algorithm is that it does not assign dependency labels to roots, i.e., to tokens having HEAD=0. To overcome this problem, we have implemented a variant of the algorithm that starts by pushing an artificial root token with ID=0 onto the stack. Tokens having HEAD=0 can now be attached to the artificial root in a RIGHT-ARC(r) action, which means that they can be assigned any label. Since this variant of the algorithm increases the overall nondeterminism, it has only been used for the data sets that include informative root labels (Arabic, Czech, Portuguese, Slovene).

	FORM	LEMMA	CPOS	POS	FEATS	DEPREL
S: *top*	+	+	+	+	+	+
S: *top*−1				+		
I: *next*	+	+	+	+	+	
I: *next*+1	+			+		
I: *next*+2				+		
I: *next*+3				+		
G: head of *top*	+					
G: leftmost dep of *top*						+
G: rightmost dep of *top*						+
G: leftmost dep of *next*						+

Table 1: Base model; S: stack, I: input, G: graph

2.2 History-Based Feature Models

History-based parsing models rely on features of the derivation history to predict the next parser action. The features used in our system are all symbolic and extracted from the following fields of the data representation: FORM, LEMMA, CPOSTAG, POSTAG, FEATS, and DEPREL. Features of the type DEPREL have a special status in that they are extracted during parsing from the partially built dependency graph and may therefore contain errors, whereas all the other features have gold standard values during both training and parsing.[2]

Based on previous research, we defined a base model to be used as a starting point for language-specific feature selection. The features of this model are shown in Table 1, where rows denote tokens in a parser configuration (defined relative to the stack, the remaining input, and the partially built dependency graph), and where columns correspond to data fields. The base model contains twenty features, but note that the fields LEMMA, CPOS and FEATS are not available for all languages.

2.3 Support Vector Machines

We use support vector machines[3] to predict the next parser action from a feature vector representing the history. More specifically, we use LIBSVM (Chang and Lin, 2001) with a quadratic kernel $K(x_i, x_j) = (\gamma x_i^T x_j + r)^2$ and the built-in one-versus-all strategy for multi-class classification. Symbolic features are converted to numerical features using the standard technique of binarization, and we split values of the FEATS field into its atomic components.[4]

For some languages, we divide the training data into smaller sets, based on some feature s (normally the CPOS or POS of the next input token), which may reduce training times without a significant loss in accuracy (Yamada and Matsumoto, 2003). To avoid too small training sets, we pool together categories that have a frequency below a certain threshold t.

2.4 Pseudo-Projective Parsing

Pseudo-projective parsing was proposed by Nivre and Nilsson (2005) as a way of dealing with non-projective structures in a projective data-driven parser. We projectivize training data by a minimal transformation, lifting non-projective arcs one step at a time, and extending the arc label of lifted arcs using the encoding scheme called HEAD by Nivre and Nilsson (2005), which means that a lifted arc is assigned the label $r{\uparrow}h$, where r is the original label and h is the label of the original head in the non-projective dependency graph.

Non-projective dependencies can be recovered by applying an inverse transformation to the output of the parser, using a left-to-right, top-down, breadth-first search, guided by the extended arc labels $r{\uparrow}h$ assigned by the parser. This technique has been used without exception for all languages.

3 Experiments

Since the projective parsing algorithm and graph transformation techniques are the same for all data sets, our optimization efforts have been focused on *feature selection*, using a combination of backward and forward selection starting from the base model described in section 2.2, and *parameter optimization* for the SVM learner, using grid search for an optimal combination of the kernel parameters γ and r, the penalty parameter C and the termination criterion ϵ, as well as the splitting feature s and the frequency threshold t. Feature selection and parameter optimization have to some extent been interleaved, but the amount of work done varies between languages.

[2] The fields PHEAD and PDEPREL have not been used at all, since we rely on pseudo-projective parsing for the treatment of non-projective structures.

[3] We also ran preliminary experiments with memory-based learning but found that this gave consistently lower accuracy.

[4] Preliminary experiments showed a slight improvement for most languages when splitting the FEATS values, as opposed to taking every combination of atomic values as a distinct value.

	Ara	Bul	Chi	Cze	Dan	Dut	Ger	Jap	Por	Slo	Spa	Swe	Tur	Total
LAS	66.71	87.41	86.92	78.42	84.77	78.59	85.82	91.65	87.60	70.30	81.29	84.58	65.68	80.19
UAS	77.52	91.72	90.54	84.80	89.80	81.35	88.76	93.10	91.22	78.72	84.67	89.50	75.82	85.48
LAcc	80.34	90.44	89.01	85.40	89.16	83.69	91.03	94.34	91.54	80.54	90.06	87.39	78.49	86.75

Table 2: Evaluation on final test set; LAS = labeled attachment score, UAS = unlabeled attachment score, LAcc = label accuracy score; total score excluding Bulgarian

The main optimization criterion has been labeled attachment score on held-out data, using ten-fold cross-validation for all data sets with 100k tokens or less, and an 80-20 split into training and devtest sets for larger datasets. The number of features in the optimized models varies from 16 (Turkish) to 30 (Spanish), but the models use all fields available for a given language, except that FORM is not used for Turkish (only LEMMA). The SVM parameters fall into the following ranges: γ: 0.12–0.20; r: 0.0–0.6; C: 0.1–0.7; ϵ: 0.01–1.0. Data has been split on the POS of the next input token for Czech (t = 200), German (t = 1000), and Spanish (t = 1000), and on the CPOS of the next input token for Bulgarian (t = 1000), Slovene (t = 600), and Turkish (t = 100). (For the remaining languages, the training data has not been split at all.)[5] A dry run at the end of the development phase gave a labeled attachment score of 80.46 over the twelve required languages.

Table 2 shows final test results for each language and for the twelve required languages together. The total score is only 0.27 percentage points below the score from the dry run, which seems to indicate that models have not been overfitted to the training data. The labeled attachment score varies from 91.65 to 65.68 but is above average for all languages. We have the best reported score for Japanese, Swedish and Turkish, and the score for Arabic, Danish, Dutch, Portuguese, Spanish, and overall does not differ significantly from the best one. The unlabeled score is less competitive, with only Turkish having the highest reported score, which indirectly indicates that the integration of labels into the parsing process primarily benefits labeled accuracy.

4 Error Analysis

An overall error analysis is beyond the scope of this paper, but we will offer a few general observations before we turn to Swedish and Turkish, focusing on recall and precision of root nodes, as a reflection of global syntactic structure, and on attachment score as a function of arc length. If we start by considering languages with a labeled attachment score of 85% or higher, they are characterized by high precision and recall for root nodes, typically 95/90, and by a graceful degradation of attachment score as arcs grow longer, typically 95–90–85, for arcs of length 1, 2 and 3–6. Typical examples are Bulgarian (Simov et al., 2005; Simov and Osenova, 2003), Chinese (Chen et al., 2003), Danish (Kromann, 2003), and Swedish (Nilsson et al., 2005). Japanese (Kawata and Bartels, 2000), despite a very high accuracy, is different in that attachment score drops from 98% to 85%, as we go from length 1 to 2, which may have something to do with the data consisting of transcribed speech with very short utterances.

A second observation is that a high proportion of non-projective structures leads to fragmentation in the parser output, reflected in lower precision for roots. This is noticeable for German (Brants et al., 2002) and Portuguese (Afonso et al., 2002), which still have high overall accuracy thanks to very high attachment scores, but much more conspicuous for Czech (Böhmová et al., 2003), Dutch (van der Beek et al., 2002) and Slovene (Džeroski et al., 2006), where root precision drops more drastically to about 69%, 71% and 41%, respectively, and root recall is also affected negatively. On the other hand, all three languages behave like high-accuracy languages with respect to attachment score. A very similar pattern is found for Spanish (Civit Torruella and Martí Antonín, 2002), although this cannot be explained by a high proportion of non-projective structures. One possible explanation in this case may be the fact that dependency graphs in the Spanish data are sparsely labeled, which may cause problem for a parser that relies on dependency labels as features.

The results for Arabic (Hajič et al., 2004; Smrž et al., 2002) are characterized by low root accuracy

[5] Detailed specifications of the feature models and learning algorithm parameters can be found on the MaltParser web page.

as well as a rapid degradation of attachment score with arc length (from about 93% for length 1 to 67% for length 2). By contrast, Turkish (Oflazer et al., 2003; Atalay et al., 2003) exhibits high root accuracy but consistently low attachment scores (about 88% for length 1 and 68% for length 2). It is noteworthy that Arabic and Turkish, being "typological outliers", show patterns that are different both from each other and from most of the other languages.

4.1 Swedish

A more fine-grained analysis of the Swedish results reveals a high accuracy for function words, which is compatible with previous studies (Nivre, 2006). Thus, the labeled F-score is 100% for infinitive markers (IM) and subordinating conjunctions (UK), and above 95% for determiners (DT). In addition, subjects (SS) have a score above 90%. In all these cases, the dependent has a configurationally defined (but not fixed) position with respect to its head.

Arguments of the verb, such as objects (DO, IO) and predicative complements (SP), have a slightly lower accuracy (about 85% labeled F-score), which is due to the fact that they "compete" in the same structural positions, whereas adverbials (labels that end in A) have even lower scores (often below 70%). The latter result must be related both to the relatively fine-grained inventory of dependency labels for adverbials and to attachment ambiguities that involve prepositional phrases. The importance of this kind of ambiguity is reflected also in the drastic difference in accuracy between noun pre-modifiers (AT) ($F > 97\%$) and noun post-modifiers (ET) ($F \approx 75\%$).

Finally, it is worth noting that coordination, which is often problematic in parsing, has high accuracy. The Swedish treebank annotation treats the second conjunct as a dependent of the first conjunct and as the head of the coordinator, which seems to facilitate parsing.[6] The attachment of the second conjunct to the first (CC) has a labeled F-score above 80%, while the attachment of the coordinator to the second conjunct (++) has a score well above 90%.

4.2 Turkish

In Turkish, very essential syntactic information is contained in the rich morphological structure, where concatenated suffixes carry information that in other languages may be expressed by separate words. The Turkish treebank therefore divides word forms into smaller units, called inflectional groups (IGs), and the task of the parser is to construct dependencies between IGs, not (primarily) between word forms (Eryiğit and Oflazer, 2006). It is then important to remember that an unlabeled attachment score of 75.8% corresponds to a word-to-word score of 82.7%, which puts Turkish on a par with languages like Czech, Dutch and Spanish. Moreover, when we break down the results according to whether the head of a dependency is part of a multiple-IG word or a complete (single-IG) word, we observe a highly significant difference in accuracy, with only 53.2% unlabeled attachment score for multiple-IG heads versus 83.7% for single-IG heads. It is hard to say at this stage whether this means that our methods are ill-suited for IG-based parsing, or whether it is mainly a case of sparse data for multiple-IG words.

When we break down the results by dependency type, we can distinguish three main groups. The first consists of determiners and particles, which have an unlabeled attachment score over 80% and which are found within a distance of 1–1.4 IGs from their head.[7] The second group mainly contains subjects, objects and different kinds of adjuncts, with a score in the range 60–80% and a distance of 1.8–5.2 IGs to their head. In this group, information about case and possessive features of nominals is important, which is found in the FEATS field in the data representation. We believe that one important explanation for our relatively good results for Turkish is that we break down the FEATS information into its atomic components, independently of POS and CPOS tags, and let the classifier decide which one to use in a given situation. The third group contains distant dependencies, such as sentence modifiers, vocatives and appositions, which have a much lower accuracy.

5 Conclusion

The evaluation shows that labeled pseudo-projective dependency parsing, using a deterministic parsing algorithm and SVM classifiers, gives competitive parsing accuracy for all languages involved in the

[6] The analysis is reminiscent of the treatment of coordination in the Collins parser (Collins, 1999).

[7] Given that the average IG count of a word is 1.26 in the treebank, this means that they are normally adjacent to the head word.

shared task, although the level of accuracy varies considerably between languages. To analyze in depth the factors determining this variation, and to improve our parsing methods accordingly to meet the challenges posed by the linguistic diversity, will be an important research goal for years to come.

Acknowledgments

We are grateful for the support from TÜBİTAK (The Scientific and Technical Research Council of Turkey) and the Swedish Research Council. We also want to thank Atanas Chanev for assistance with Slovene, the organizers of the shared task for all their hard work, and the creators of the treebanks for making the data available.

References

A. Abeillé, editor. 2003. *Treebanks: Building and Using Parsed Corpora*, volume 20 of *Text, Speech and Language Technology*. Kluwer Academic Publishers, Dordrecht.

S. Afonso, E. Bick, R. Haber, and D. Santos. 2002. "Floresta sintá(c)tica": a treebank for Portuguese. In *Proc. of LREC-2002*, pages 1698–1703.

N. B. Atalay, K. Oflazer, and B. Say. 2003. The annotation process in the Turkish treebank. In *Proc. of LINC-2003*.

E. Black, F. Jelinek, J. D. Lafferty, D. M. Magerman, R. L. Mercer, and S. Roukos. 1992. Towards history-based grammars: Using richer models for probabilistic parsing. In *Proc. of the 5th DARPA Speech and Natural Language Workshop*, pages 31–37.

A. Böhmová, J. Hajič, E. Hajičová, and B. Hladká. 2003. The PDT: a 3-level annotation scenario. In Abeillé (Abeillé, 2003), chapter 7.

S. Brants, S. Dipper, S. Hansen, W. Lezius, and G. Smith. 2002. The TIGER treebank. In *Proc. of TLT-2002*.

C.-C. Chang and C.-J. Lin, 2001. *LIBSVM: A Library for Support Vector Machines*. Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm.

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.

M. Civit Torruella and M^{a} A. Martí Antonín. 2002. Design principles for a Spanish treebank. In *Proc. of TLT-2002*.

M. Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania.

S. Džeroski, T. Erjavec, N. Ledinek, P. Pajas, Z. Žabokrtsky, and A. Žele. 2006. Towards a Slovene dependency treebank. In *Proc. of LREC-2006*.

G. Eryiğit and K. Oflazer. 2006. Statistical dependency parsing of Turkish. In *Proc. of EACL-2006*.

J. Hajič, O. Smrž, P. Zemánek, J. Šnaidauf, and E. Beška. 2004. Prague Arabic dependency treebank: Development in data and tools. In *Proc. of NEMLAR-2004*, pages 110–117.

Y. Kawata and J. Bartels. 2000. Stylebook for the Japanese treebank in VERBMOBIL. Verbmobil-Report 240, Seminar für Sprachwissenschaft, Universität Tübingen.

M. T. Kromann. 2003. The Danish dependency treebank and the underlying linguistic theory. In *Proc. of TLT-2003*.

T. Kudo and Y. Matsumoto. 2002. Japanese dependency analysis using cascaded chunking. In *Proc. of CoNLL-2002*, pages 63–69.

J. Nilsson, J. Hall, and J. Nivre. 2005. MAMBA meets TIGER: Reconstructing a Swedish treebank from antiquity. In *Proc. of the NODALIDA Special Session on Treebanks*.

J. Nivre and J. Nilsson. 2005. Pseudo-projective dependency parsing. In *Proc. of ACL-2005*, pages 99–106.

J. Nivre, J. Hall, and J. Nilsson. 2004. Memory-based dependency parsing. In *Proc. CoNLL-2004*, pages 49–56.

J. Nivre, J. Hall, and J. Nilsson. 2006. MaltParser: A data-driven parser-generator for dependency parsing. In *Proc. of LREC-2006*.

J. Nivre. 2003. An efficient algorithm for projective dependency parsing. In *Proc. of IWPT-2003*, pages 149–160.

J. Nivre. 2006. *Inductive Dependency Parsing*. Springer.

K. Oflazer, B. Say, D. Zeynep Hakkani-Tür, and G. Tür. 2003. Building a Turkish treebank. In Abeillé (Abeillé, 2003), chapter 15.

K. Simov and P. Osenova. 2003. Practical annotation scheme for an HPSG treebank of Bulgarian. In *Proc. of LINC-2003*, pages 17–24.

K. Simov, P. Osenova, A. Simov, and M. Kouylekov. 2005. Design and implementation of the Bulgarian HPSG-based treebank. In *Journal of Research on Language and Computation – Special Issue*, pages 495–522. Kluwer Academic Publishers.

O. Smrž, J. Šnaidauf, and P. Zemánek. 2002. Prague dependency treebank for Arabic: Multi-level annotation of Arabic corpus. In *Proc. of the Intern. Symposium on Processing of Arabic*, pages 147–155.

L. van der Beek, G. Bouma, R. Malouf, and G. van Noord. 2002. The Alpino dependency treebank. In *Computational Linguistics in the Netherlands (CLIN)*.

H. Yamada and Y. Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *Proc. of IWPT-2003*, pages 195–206.