Dependency Parsing by Inference over High-recall Dependency Predictions

Sander Canisius, Toine Bogers,
Antal van den Bosch, Jeroen Geertzen
ILK / Computational Linguistics and AI
Tilburg University, P.O. Box 90153,
NL-5000 LE Tilburg, The Netherlands
{S.V.M.Canisius,A.M.Bogers,
Antal.vdnBosch,J.Geertzen}@uvt.nl

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

1 Introduction

As more and more syntactically-annotated corpora become available for a wide variety of languages, machine learning approaches to parsing gain interest as a means of developing parsers without having to repeat some of the labor-intensive and language-specific activities required for traditional parser development, such as manual grammar engineering, for each new language. The CoNLL-X shared task on multi-lingual dependency parsing (Buchholz et al., 2006) aims to evaluate and advance the state-of-the-art in machine learning-based dependency parsing by providing a standard benchmark set comprising thirteen languages[1]. In this paper, we describe two different machine learning approaches to the CoNLL-X shared task.

Before introducing the two learning-based approaches, we first describe a number of baselines, which provide simple reference scores giving some sense of the difficulty of each language. Next, we present two machine learning systems: 1) an approach that directly predicts all dependency relations in a single run over the input sentence, and 2) a cascade of phrase recognizers. The first approach has been found to perform best and was selected for submission to the competition. We conclude this paper with a detailed error analysis of its output for two of the thirteen languages, Dutch and Spanish.

[1]The data sets were extracted from various existing treebanks (Hajič et al., 2004; Simov et al., 2005; Simov and Osenova, 2003; 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)

2 Baseline approaches

Given the diverse range of languages involved in the shared task, each having different characteristics probably requiring different parsing strategies, we developed four different baseline approaches for assigning labeled dependency structures to sentences. All of the baselines produce strictly projective structures. While the simple rules implementing these baselines are insufficient for achieving state-of-the-art performance, they do serve a useful role in giving a sense of the difficulty of each of the thirteen languages. The heuristics for constructing the trees and labeling the relations used by each of the four baselines are described below.

Binary right-branching trees The first baseline produces right-branching binary trees. The first token in the sentence is marked as the top node with HEAD `0` and DEPREL `ROOT`. For the rest of the tree, token $n - 1$ serves as the HEAD of token n. Figure 1 shows an example of the kind of tree this baseline produces.

Binary left-branching trees The binary left-branching baseline mirrors the previous baseline. The penultimate token in the sentence is marked as the top node with HEAD `0` and DEPREL `ROOT` since punctuation tokens can never serve as `ROOT`[2]. For the rest of the tree, the HEAD of token n is token $n+1$. Figure 2 shows an example of a tree produced by this baseline.

[2]We simply assume the final token in the sentence to be punctuation.

Inward-branching trees In this approach, the first identified verb[3] is marked as the ROOT node. The part of the sentence to the left of the ROOT is left-branching, the part to the right of the ROOT is right-branching. Figure 3 shows an example of a tree produced by this third baseline.

Nearest neighbor-branching trees In our most complex baseline, the first verb is marked as the ROOT node and the other verbs (with DEPREL vc) point to the closest preceding verb. The other tokens point in the direction of their nearest neighboring verb, i.e. the two tokens at a distance of 1 from a verb have that verb as their HEAD, the two tokens at a distance of 2 have the tokens at a distance of 1 as their head, and so on until another verb is a closer neighbor. In the case of ties, i.e. tokens that are equally distant from two different verbs, the token is linked to the preceding token. Figure 4 clarifies this kind of dependency structure in an example tree.

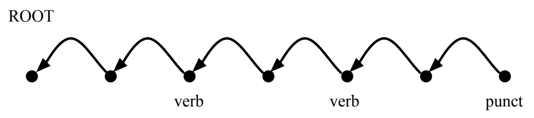

Figure 1: Binary right-branching tree for an example sentence with two verbs.

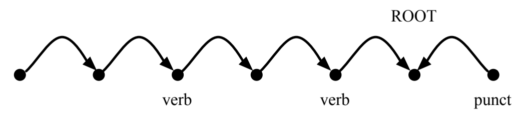

Figure 2: Binary left-branching tree for the example sentence.

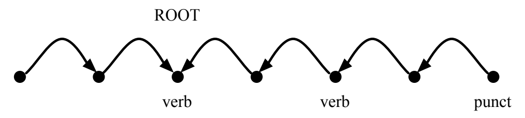

Figure 3: Binary inward-branching tree for the example sentence.

[3]We consider a token a verb if its CPOSTAG starts with a 'V'. This is an obviously imperfect, but language-independent heuristic choice.

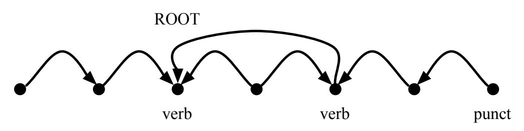

Figure 4: Nearest neighbor-branching tree for the example sentence.

Labeling of identified relations is done using a three-fold back-off strategy. From the training set, we collect the most frequent DEPREL tag for each head-dependent FORM pair, the most frequent DEPREL tag for each FORM, and the most frequent DEPREL tag in the entire training set. The relations are labeled in this order: first, we look up if the FORM pair of a token and its head was present in the training data. If not, then we assign it the most frequent DEPREL tag in the training data for that specific token FORM. If all else fails we label the token with the most frequent DEPREL tag in the entire training set (excluding punct[4] and ROOT).

language	**baseline**	**unlabeled**	**labeled**
Arabic	left	58.82	39.72
Bulgarian	inward	41.29	29.50
Chinese	NN	37.18	25.35
Czech	NN	34.70	22.28
Danish	inward	50.22	36.83
Dutch	NN	34.07	26.87
German	NN	33.71	26.42
Japanese	right	67.18	64.22
Portuguese	right	25.67	22.32
Slovene	right	24.12	19.42
Spanish	inward	32.98	27.47
Swedish	NN	34.30	21.47
Turkish	right	49.03	31.85

Table 1: The labeled and unlabeled scores for the best performing baseline for each language (NN = nearest neighbor-branching).

The best baseline performance (labeled and unlabeled scores) for each language is listed in Table 1. There was no single baseline that outperformed the others on all languages. The nearest neighbor baseline outperformed the other baselines on five of the thirteen languages. The right-branching and

[4]Since the evaluation did not score on punctuation.

inward-branching baselines were optimal on four and three languages respectively. The only language where the left-branching trees provide the best performance is Arabic.

3 Parsing by inference over high-recall dependency predictions

In our approach to dependency parsing, a machine learning classifier is trained to predict (directed) labeled dependency relations between a head and a dependent. For each token in a sentence, instances are generated where this token is a potential dependent of each of the other tokens in the sentence[5]. The label that is predicted for each classification case serves two different purposes at once: 1) it signals whether the token is a dependent of the designated head token, and 2) if the instance does in fact correspond to a dependency relation in the resulting parse of the input sentence, it specifies the type of this relation, as well.

The features we used for encoding instances for this classification task correspond to a rather simple description of the head-dependent pair to be classified. For both the potential head and dependent, there are features encoding a 2-1-2 window of words and part-of-speech tags[6]; in addition, there are two spatial features: a relative position feature, encoding whether the dependent is located to the left or to the right of its potential head, and a distance feature that expresses the number of tokens between the dependent and its head.

One issue that may arise when considering each potential dependency relation as a separate classification case is that inconsistent trees are produced. For example, a token may be predicted to be a dependent of more than one head. To recover a valid dependency tree from the separate dependency predictions, a simple inference procedure is performed. Consider a token for which the dependency relation is to be predicted. For this token, a number of classification cases have been processed, each of them indicating whether and if so how the token is related to one of the other tokens in the sentence. Some of these predictions may be negative, i.e. the token is not a dependent of a certain other token in the sentence, others may be positive, suggesting the token is a dependent of some other token.

If all classifications are negative, the token is assumed to have no head, and consequently no dependency relation is added to the tree for this token; the node in the dependency tree corresponding to this token will then be an isolated one. If one of the classifications is non-negative, suggesting a dependency relation between this token as a dependent and some other token as a head, this dependency relation is added to the tree. Finally, there is the case in which more than one prediction is non-negative. By definition, at most one of these predictions can be correct; therefore, only one dependency relation should be added to the tree. To select the most-likely candidate from the predicted dependency relations, the candidates are ranked according to the classification confidence of the base classifier that predicted them, and the highest-ranked candidate is selected for insertion into the tree.

For our base classifier we used a memory-based learner as implemented by TiMBL (Daelemans et al., 2004). In memory-based learning, a machine learning method based on the nearest-neighbor rule, the class for a given test instance is predicted by performing weighted voting over the class labels of a certain number of most-similar training instances. As a simple measure of confidence for such a prediction, we divide the weight assigned to the majority class by the total weight assigned to all classes. Though this confidence measure is a rather ad-hoc one, which should certainly not be confused with any kind of probability, it tends to work quite well in practice, and arguably did so in the context of this study. The parameters of the memory-based learner have been optimized for accuracy separately for each language on training and development data sampled internally from the training set.

The base classifier in our parser is faced with a classification task with a highly skewed class distribution, i.e. instances that correspond to a dependency relation are largely outnumbered by those that do not. In practice, such a huge number of negative instances usually results in classifiers that tend

[5]To prevent explosion of the number of classification cases to be considered for a sentence, we restrict the maximum distance between a token and its potential head. For each language, we selected this distance so that, on the training data, 95% of the dependency relations is covered.

[6]More specifically, we used the part-of-speech tags from the POSTAG column of the shared task data files.

to predict fairly conservatively, resulting in high precision, but low recall. In the approach introduced above, however, it is better to have high recall, even at the cost of precision, than to have high precision at the cost of recall. A missed relation by the base classifier can never be recovered by the inference procedure; however, due to the constraint that each token can only be a dependent of one head, excessive prediction of dependency relations can still be corrected by the inference procedure. An effective method for increasing the recall of a classifier is down-sampling of the training data. In down-sampling, instances belonging to the majority class (in this case the negative class) are removed from the training data, so as to obtain a more balanced distribution of negative and non-negative instances.

Figure 5 shows the effect of systematically removing an increasingly larger part of the negative instances from the training data. First of all, the figure confirms that down-sampling helps to improve recall, though it does so at the cost of precision. More importantly however, it also illustrates that this improved recall is beneficial for the performance of the dependency parser. The shape of the performance curve of the dependency parser closely follows that of the recall. Remarkably, parsing performance continues to improve with increasingly stronger down-sampling, even though precision drops considerably as a result of this. This shows that the confidence of the classifier for a certain prediction is a sufficiently reliable indication of the quality of that prediction for fixing the over-prediction of dependency relations. Only when the number of negative training instances is reduced to equal the number of positive instances, the performance of the parser is negatively affected. Based on a quick evaluation of various down-sampling ratios on a 90%-10% train-test split of the Dutch training data, we decided to down-sample the training data for all languages with a ratio of two negative instances for each positive one.

Table 2 lists the unlabeled and labeled attachment scores of the resulting system for all thirteen languages.

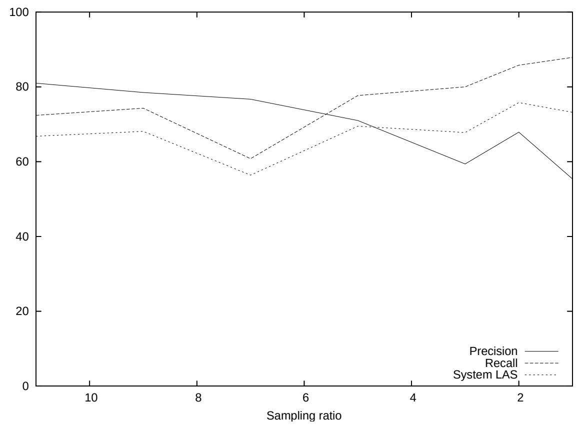

Figure 5: The effect of down-sampling on precision and recall of the base classifier, and on labeled accuracy of the dependency parser. The x-axis refers to the number of negative instances for each positive instance in the training data. Training and testing was performed on a 90%-10% split of the Dutch training data.

4 Cascaded dependency parsing

One of the alternative strategies explored by us was modeling the parsing process as a cascaded pair of basic learners. This approach is similar to Yamada and Matsumoto (2003) but we only use their Left and Right reduction operators, not Shift. In the first phase, each learner predicted dependencies between neighboring words. Dependent words were removed and the remaining words were sent to the learners for further rounds of processing until all words but one had been assigned a head. Whenever crossing links prevented further assignments of heads to words, the learner removed the remaining word requiring the longest dependency link. When the first phase was finished another learner assigned labels to pairs of words present in dependency links.

Unlike in related earlier work (Tjong Kim Sang, 2002), we were unable to compare many different learner configurations. We used two different training files for the first phase: one for predicting the dependency links between adjacent words and one for predicting all other links. As a learner, we used TiMBL with its default parameters. We evaluated different feature sets and ended up with using words, lemmas, POS tags and an extra pair of features with the POS tags of the children of the focus word. With this configuration, this cascaded approach achieved a labeled score of 62.99 on the Dutch test data compared to 74.59 achieved by our main approach.

language	unlabeled	labeled
Arabic	74.59	57.64
Bulgarian	82.51	78.74
Chinese	82.86	78.37
Czech	72.88	60.92
Danish	82.93	77.90
Dutch	77.79	74.59
German	80.01	77.56
Japanese	89.67	87.41
Portuguese	85.61	77.42
Slovene	74.02	59.19
Spanish	71.33	68.32
Swedish	85.08	79.15
Turkish	64.19	51.07

Table 2: The labeled and unlabeled scores for the submitted system for each of the thirteen languages.

5 Error analysis

We examined the system output for two languages in more detail: Dutch and Spanish.

5.1 Dutch

With a labeled attachment score of 74.59 and an unlabeled attachment score of 77.79, our submitted Dutch system performs somewhat above the average over all submitted systems (labeled 70.73, unlabeled 75.07). We review the most notable errors made by our system.

From a part-of-speech (CPOSTAG) perspective, a remarkable relative amount of head and dependency errors are made on **conjunctions**. A likely explanation is that the tag "Conj" applies to both coordinating and subordinating conjunctions; we did not use the FEATS information that made this distinction, which would have likely solved some of these errors.

Left- and right-directed attachment to heads is roughly equally successful. Many errors are made on relations attaching to ROOT; the system appears to be overgenerating attachments to ROOT, mostly in cases when it should have generated rightward attachments. Unsurprisingly, the more distant the head is, the less accurate the attachment; especially recall suffers at distances of three and more tokens.

The most frequent attachment error is generating a ROOT attachment instead of a "mod" (modifier) relation, often occurring at the start of a sentence. Many errors relate to ambiguous adverbs such as *bovendien* (moreover), *tenslotte* (after all), and *zo* (thus), which tend to occur rather frequently at the beginning of sentences in the test set, but less so in the training set. The test set appears to consist largely of formal journalistic texts which typically tend to use these marked rhetorical words in sentence-initial position, while the training set is a more mixed set of texts from different genres plus a significant set of individual sentences, often manually constructed to provide particular examples of syntactic constructions.

5.2 Spanish

The Spanish test data set was the only data set on which the alternative cascaded approach (72.15) outperformed our main approach (68.32). A detailed comparison of the output files of the two systems has revealed two differences. First, the amount of circular links, a pair of words which have each other as head, was larger in the analysis of the submitted system (7%) than in the cascaded analysis (3%) and the gold data (also 3%). Second, the number of root words per sentence (always 1 in the gold data) was more likely to be correct in the cascaded analysis (70% correct; other sentences had no root) than in the submitted approach (40% with 20% of the sentences being assigned no roots and 40% more than one root). Some of these problems might be solvable with post-processing

Acknowledgements

This research is funded by NWO, the Netherlands Organization for Scientific Research under the IMIX programme, and the Dutch Ministry for Economic Affairs' IOP-MMI programme.

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 the Third Intern. Conf. on Language Resources and Evaluation (LREC)*, pages 1698–1703.

N. B. Atalay, K. Oflazer, and B. Say. 2003. The annotation process in the Turkish treebank. In *Proc. of the 4th*

Intern. Workshop on Linguistically Interpreteted Corpora (LINC).

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 the First Workshop on Treebanks and Linguistic Theories (TLT)*.

S. Buchholz, E. Marsi, A. Dubey, and Y. Krymolowski. 2006. CoNLL-X shared task on multilingual dependency parsing. In *Proc. of the Tenth Conf. on Computational Natural Language Learning (CoNLL-X)*. SIGNLL.

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 the First Workshop on Treebanks and Linguistic Theories (TLT)*.

W. Daelemans, J. Zavrel, K. Van der Sloot, and A. Van den Bosch. 2004. TiMBL: Tilburg memory based learner, version 5.1, reference guide. Technical Report ILK 04-02, ILK Research Group, Tilburg University.

S. Džeroski, T. Erjavec, N. Ledinek, P. Pajas, Z. Žabokrtsky, and A. Žele. 2006. Towards a Slovene dependency treebank. In *Proc. of the Fifth Intern. Conf. on Language Resources and Evaluation (LREC)*.

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.

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 the Second Workshop on Treebanks and Linguistic Theories (TLT)*.

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*.

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 the 4th Intern. Workshop on Linguistically Interpreteted Corpora (LINC)*, 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.

Erik Tjong Kim Sang. 2002. Memory-based shallow parsing. *Journal of Machine Learning Research*, 2(Mar):559–594.

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

Hiroyasu Yamada and Yuji Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *8th International Workshop of Parsing Technologies (IWPT2003)*. Nancy, France.