Using a maximum entropy-based tagger to improve a very fast vine parser

Anders Søgaard
Center for Language Technology
University of Copenhagen
soegaard@hum.ku.dk

Jonas Kuhn
Dpt. of Linguistics
University of Potsdam
kuhn@ling.uni-potsdam.de

Abstract

In this short paper, an off-the-shelf maximum entropy-based POS-tagger is used as a partial parser to improve the accuracy of an extremely fast linear time dependency parser that provides state-of-the-art results in multilingual unlabeled POS sequence parsing.

1 Introduction

The dependency parsing literature has grown in all directions the past 10 years or so. Dependency parsing is used in a wide variety of applications, and many different parsing techniques have been proposed.

Two dependency parsers have become more popular than the rest, namely MSTParser (McDonald et al., 2005) and MaltParser (Nivre et al., 2007). MSTParser is slightly more accurate than MaltParser on most languages, especially when dependencies are long and non-projective, but MaltParser is theoretically more efficient as it runs in linear time. Both are relatively slow in terms of training (hours, sometimes days), and relatively big models are queried in parsing.

MSTParser and MaltParser can be optimized for speed in various ways,[1] but the many applications of dependency parsers today may turn model size into a serious problem. MSTParser typically takes about a minute to parse a small standard test suite, say 2–300 sentences; the stand-alone version of MaltParser may take 5–8 minutes. Such parsing times are problematic in, say, a machine translation system where for each sentence pair multiple target sentences are parsed (Charniak et al., 2003; Galley and Manning, 2009). Since training takes hours or days, researchers are also more reluctant to experiment with new features, and it is very likely that the features typically used in parsing are suboptimal in, say, machine translation.

Conceptually simpler dependency parsers are also easier to understand, which makes debugging, cross-domain adaption or cross-language adaptation a lot easier. Finally, state-of-the-art dependency parsers may in fact be outperformed by simpler systems on non-standard test languages with, say, richer morphology or more flexible word order.

Vine parsing is a parsing strategy that guarantees fast parsing and smaller models, but the accuracy of dependency-based vine parsers has been non-competitive (Eisner and Smith, 2005; Dreyer et al., 2006).

This paper shows how the accuracy of dependency-based vine parsers can be improved by 1–5% across six very different languages with a very small cost in training time and practically no cost in parsing time.

The main idea in our experiments is to use a maximum entropy-based part-of-speech (POS) tagger to identify roots and tokens whose heads are immediately left or right of them. These are tasks that a tagger can solve. You simply read off a tagged text from the training, resp. test, section of a treebank and replace all tags of roots, i.e. tokens whose syntactic head is an artificial root node, with a new tag ROOT. You then train on the training section and apply your tagger on the test section. The decisions made by the tagger are then, subsequently, used as hard constraints by your parser. When the parser then tries to find root nodes, for instance, it is forced to use the roots assigned by the tagger. This strategy is meaningful if the tagger has better precision for roots than the parser. If it has better recall than the parser, the

[1]Recent work has optimized MaltParser considerably for speed. Goldberg and Elhadad (2008) speed up the MaltParser by a factor of 30 by simplifying the decision function for the classifiers. Parsing is still considerably slower than with our vine parser, i.e. a test suite is parsed in about 15–20 seconds, whereas our vine parser parses a test suite in less than two seconds.

parser may be forced to select roots only from the set of potential roots assigned by the tagger. In our experiments, only the first strategy was used (since the tagger's precision was typically better than its recall).

The dependency parser used in our experiments is very simple. It is based on the Chu-Liu-Edmonds algorithm (Edmonds, 1967), which is also used in the MSTParser (McDonald et al., 2005), but it is informed only by a simple MLE training procedure and omits cycle contraction in parsing. This means that it produces cyclic graphs. In the context of poor training, insisting on acyclic output graphs often compromises accuracy by $> 10\%$. On top of this parser, which is super fast but often does not even outperform a simple structural baseline, hard and soft constraints on dependency length are learned discriminatively. The speed of the parser allows us to repeatedly parse a tuning section to optimize these constraints. In particular, the tuning section (about 7500 tokens) is parsed a fixed number of times for each POS/CPOS tag to find the optimal dependency length constraint when that tag is the tag of the head or dependent word. In general, this discriminative training procedure takes about 10 minutes for an average-sized treebank. The parser only produces unlabeled dependency graphs and is still under development. While accuracy is below state-of-the-art results, our *improved* parser significantly outperforms a default version of the MaltParser that is restricted to POS tags only, on 5/6 languages ($p \leq 0.05$), and it significantly outperforms the baseline vine parser on all languages.

2 Data

Our languages are chosen from different language families. Arabic is a Semitic language, Czech is Slavic, Dutch is Germanic, Italian is Romance, Japanese is Japonic-Ryukyuan, and Turkish is Uralic. All treebanks, except Italian, were also used in the CONLL-X Shared Task (Buchholz and Marsi, 2006). The Italian treebank is the law section of the TUT Treebank used in the Evalita 2007 Dependency Parsing Challenge (Bosco et al., 2000).

3 Experiments

The Python/C++ implementation of the maximum entropy-based part-of-speech (POS) tagger first described in Ratnaparkhi (1998) that comes with the maximum entropy library in Zhang (2004) was used to identify arcs to the root node and to tokens immediately left or right of the dependent. This was done by first extracting a tagged text from each treebank with dependents of the root node assigned a special tag ROOT. Similarly, tagged texts were extracted in which dependents of their immediate left, resp. right neighbors, were assigned a special tag. Our tagger was trained on the texts extracted from the training sections of the treebanks and evaluated on the texts extracted from the test sections. The number of gold standard, resp. predicted, ROOT/LEFT/RIGHT tags are presented in Figure 1. Precision and f-score are also computed. Note that since our parser uses information from our tagger as hard constraints, i.e. it disregards arcs to the root node or immediate neighbors *not* predicted by our tagger, precision is really what is important, not f-score. Or more precisely, precision indicates *if* our tagger is of any help to us, and f-score tells us to what extent it may be of help.

4 Results

The results in Figure 2 show that using a maximum entropy-based POS tagger to identify roots (ROOT), tokens with immediate left heads (LEFT) and tokens with immediate (RIGHT) heads improves the accuracy of a baseline vine parser across the board for all languages measured in terms of unlabeled attachment score (ULA), or decreases are insignificant (Czech and Turkish). For all six languages, there is a combination of ROOT, LEFT and RIGHT that significantly outperforms the vine parser baseline. In 4/6 cases, absolute improvements are $\geq 2\%$. The score for Dutch is improved by $> 4\%$. The extended vine parser is also significantly better than the MaltParser restricted to POS tags on 5/6 languages. MaltParser is probably better than the vine parser wrt. Japanese because average sentence length in this treebank is *very* short (8.9); constraints on dependency length do not really limit the search space.

In spite of the fact that our parser only uses POS tags (except for the maximum entropy-based tagger which considers both words and tags), scores are now comparable to more mature dependency parsers: ULA excl. punctuation for Arabic is 70.74 for Vine+ROOT+LEFT+RIGHT which is better than six of the systems who participated in the CONLL-X Shared Task and who had access to *all* data in the treebank, i.e. tokens, lemmas, POS

Arabic	Gold	Predicted	Precision	F-score
ROOT	443	394	89.09	83.87
LEFT	3035	3180	84.28	86.24
RIGHT	313	196	82.14	63.26
Czech	Gold	Predicted	Precision	F-score
ROOT	737	649	85.36	79.94
LEFT	1485	1384	85.12	82.12
RIGHT	1288	1177	87.51	83.57
Dutch	Gold	Predicted	Precision	F-score
ROOT	522	360	74.44	60.77
LEFT	1734	1595	87.02	83.39
RIGHT	1300	1200	87.00	83.52
Italian	Gold	Predicted	Precision	F-score
ROOT	100	58	74.36	65.17
LEFT	1601	1640	90.30	91.39
RIGHT	192	129	84.87	74.14
Japanese	Gold	Predicted	Precision	F-score
ROOT	939	984	85.06	87.05
LEFT	1398	1382	97.76	97.19
RIGHT	2838	3016	92.27	95.08
Turkish	Gold	Predicted	Precision	F-score
ROOT	694	685	85.55	84.99
LEFT	750	699	91.70	88.47
RIGHT	3433	3416	84.19	83.98

Figure 1: Tag-specific evaluation of our tagger on the extracted texts.

	Arabic	Czech	Dutch	Italian	Japanese	Turkish
MaltParser	66.22	67.78	65.03	75.48	**89.13**	68.94
Vine	67.99	66.70	65.98	75.50	83.15	68.53
Vine+ROOT	68.68	66.65	66.21	78.06	83.82	68.45
Vine+ROOT+LEFT	69.68	68.14	68.05	77.14	84.64	68.37
Vine+RIGHT	68.50	67.38	68.18	**78.55**	84.17	**69.87**
Vine+ROOT+RIGHT	69.20	67.32	68.40	78.29	84.78	69.79
Vine+ROOT+LEFT+RIGHT	**70.28**	**68.70**	**70.06**	77.26	85.45	69.74

Figure 2: Labeled attachment scores (LASs) for MaltParser limited to POS tags, our baseline vine parser (Vine) and our extensions of Vine. Best scores bold-faced.

tags, features and dependency relations; not just the POS tags as in our case. In particular, our result is 2.28 better than Dreyer et al. (2006) who also use soft and hard constraints on dependency lengths. They extend the parsing algorithm in Eisner and Smith (2005) to labeled k-best parsing and use a reranker to find the best parse according to predefined global features. ULA excl. punctuation for Turkish is 67.06 which is better than six of the shared task participants, incl. Dreyer et al. (2006) (60.45).

The improvements come at an extremely low cost. The POS tagger simply stores its decisions in a very small table, typically 5–10 cells per sentence, that is queried in no time in parsing. Parsing a standard small test suite takes less than two seconds, and the cost of the additional look-up is too small to be measured. The training time of the maximum entropy-based tagger is typically a matter of seconds or half a minute. Even running it on the 1249k Prague Dependency Treebank (Czech) is only a matter of minutes.

5 Conclusion and future work

Vine parsers are motivated by efficiency and robustness (Dreyer et al., 2006), which has become more and more important over the last few years, but none of the systems introduced in the literature provide competitive results in terms of accuracy. Our experiments show how dependency-based vine parsers can be significantly improved by using a maximum entropy-based POS tagger for initial partial parsing with almost no cost in terms of training and parsing time.

Our choice of parser restricted us in a few respects. Most importantly, our results are below state-of-the-art results, and it is not clear if the strategy scales to more accurate parsers. The strategy of using a POS tagger to do partial parsing and subsequently forward high precision decisions to a parser only works on graph-based or constraint-based dependency parsers where previous decisions can be hardwired into candidate weight matrices by setting weights to 0. It would be difficult if at all possible to implement in history-based dependency parsers such as MaltParser. Experiments will be performed with the MSTParser soon.

Our parser also restricted us to considering unlabeled dependency graphs. A POS tagger, however, can also be used to identify grammatical functions (subjects, objects, . . .), for example, which may be used to hardwire dependency relations into candidate weight matrices. POS taggers may also be used to identify other dependency relations or more fine-grained features that can improve the accuracy of dependency parsers.

References

Cristina Bosco, Vincenzo Lombardo, Daniela Vassallo, and Leonardo Lesmo. 2000. Building a treebank for Italian. In *LREC*, pages 99–105, Athens, Greece.

Sabine Buchholz and Erwin Marsi. 2006. CONLL-X shared task on multilingual dependency parsing. In *CONLL-X*, pages 149–164, New York City, NY.

Eugene Charniak, Kevin Knight, and Kenji Yamada. 2003. Syntax-based language models for statistical machine translation. In *MT Summit IX*, New Orleans, Louisiana.

Markus Dreyer, David A. Smith, and Noah A. Smith. 2006. Vine parsing and minimum risk reranking for speed and precision. In *CONLL-X*, pages 201–205, New York City, NY.

J. Edmonds. 1967. Optimum branchings. *Journal of Research of the National Bureau of Standards*, 71:233–240.

Jason Eisner and Noah A. Smith. 2005. Parsing with soft and hard constraints on dependency length. In *IWPT'05*, pages 30–41, Vancouver, Canada.

Michel Galley and Cristopher Manning. 2009. Quadratic time dependency parsing for machine translation. In *ACL'09*, Singapore, Singapore. To appear.

Yoav Goldberg and Michael Elhadad. 2008. splitSVM: fast, space-efficient, non-heuristic, polynomial kernel computation for NLP applications. In *ACL'08, Short Papers*, pages 237–240, Columbus, Ohio.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič. 2005. Non-projective dependency parsing using spanning tree algorithms. In *HLT-EMNLP 2005*, pages 523–530, Vancouver, British Columbia.

Joakim Nivre, Johan Hall, Sandra Kübler, Ryan McDonald, Jens Nilsson, Sebastian Riedel, and Deniz Yuret. 2007. The CONLL 2007 shared task on dependency parsing. In *EMNLP-CONLL'07*, pages 915–932, Prague, Czech Republic.

Adwait Ratnaparkhi. 1998. *Maximum entropy models for natural language ambiguity resolution*. Ph.D. thesis, University of Pennsylvania.

Le Zhang. 2004. Maximum entropy modeling toolkit for Python and C++. University of Edinburgh. Available at homepages.inf.ed.ac.uk/lzhang10/maxent_toolkit.html.