Conjunction Representation and Ease of Domain Adaptation*

Emily Pitler
Computer and Information Science
University of Pennsylvania
Philadelphia, PA 19104
epitler@seas.upenn.edu

Abstract

Stylistic differences can interact with constituency-to-dependency conversion programs, causing almost identical constituency trees to have very different dependency trees. A dependency parser's accuracy in attaching both conjunctions and surrounding words in the tree is affected by the choice of the converter, especially on out-of-domain web texts. Some attempts to improve conjunction attachment accuracy are described.

1 Introduction

Dependency parsing is a fundamental NLP task with many practical applications, such as question answering and machine translation. Parsers are often trained on news text, but are most useful if they are robust to domain shifts. The 2012 Shared Task on Parsing the Web (Petrov and McDonald, 2012) compared how well constituency and dependency parsers trained on Wall Street Journal trees could adapt to parsing text from the web.

In the CoNLL 2007 Shared Task on Domain Adaptation (Nivre et al., 2007), in which dependency parsers trained on WSJ were tested on biomedical text and child speech, error analysis showed that the drop in accuracy between in-domain and out-of-domain test data was mostly due to annotation differences between the training and test corpora (Dredze et al., 2007). In the 2012 shared task, identical annotation guidelines were used to produce constituency trees for both the Wall Street Journal sentences and for Web sentences.

While the constituency trees are now annotated consistently, the constituency-to-dependency conversion creates significant differences in the *dependency* representations when there were no significant differences in the original *constituency* trees. We focus on the *representation of conjunctions*, especially *within noun phrases*, and provide both a quantitative and qualitative analysis of how the conversion procedure can make learning more difficult for a dependency parser, especially on new domains. We also describe attempts to compensate for these conversion artifacts.

2 Conjunctions and Conversions to Dependency Trees

Consider first conjunctions with two arguments: A and B. The Stanford converter[1] (De Marneffe et al., 2006) (used to produce dependencies in the 2012 shared task) chooses either A or B as the head, and then represents the conjunction with either A as the parent of both and and B or with B as the parent of and and B.[2] Whether A or B is the head may vary (more details in the next section). As directionality of edges is a commonly used feature in dependency parsing, these inconsistencies can affect learning. Our baseline parser has an accuracy of

*I'd like to thank Mitch Marcus for helpful conversations. This material is based upon work supported under a National Science Foundation Graduate Research Fellowship.

[1] http://nlp.stanford.edu/software/stanford-dependencies.shtml

[2] The Stanford converter has a *collapsed* mode in which the conjunction word is excised and incorporated into the label of the edge between A and B, but the *basic* mode was used in the Shared Task data.

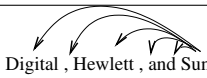
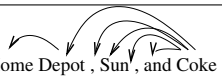
Constituency	Stanford	pennconverter
a. (NP (NNP Digital) (, ,) (NNP Hewlett) (CC and) (NNP Sun))		
b. (NP (NNP Digital) (, ,) (NNP Hewlett) **(, ,)** (CC and) (NNP Sun))		
c. (NP (NP (NNP Home) (NNP Depot)) (, ,) **(NP** (NNP Sun) **)** (, ,) (CC and) **(NP** (NNP Coke) **)**)		
d. (NP (NP (NNP Home) (NNP Depot)) (, ,) (NNP Sun) (, ,) (CC and) (NNP Coke))		

Table 1: Small changes within noun phrases can affect Stanford dependencies, potentially changing a large number of edges. Examples a and c are from the Google Web Treebank development data, and b and d are small variations.

79% on conjunctions which attach to the left and of 66% on conjunctions which attach to the right on the weblogs development set.

In contrast, the pennconverter constituency-to-dependency converter[3] (Johansson and Nugues, 2007), used to prepare dependencies for the CoNLL 2007-2009 shared tasks on dependency parsing, consistently chooses *A* (the leftmost conjunct) as the head, *and* as a child of *A*, and *B* as a child of *and*.

2.1 Sensitivity to NP-internal annotations

The original Penn Treebank left noun phrases flat. De Marneffe and Manning (2008) write: "To retrieve adequate heads from a semantic point of view, heuristics are used to inject more structure when the Penn Treebank gives only flat constituents, as is often the case for conjuncts". These compensations are less appropriate when applied to constituency trees that *do* have NP-internal annotations (which Ontonotes (Weischedel et al., 2011) and the Google Web Treebank have).

We can therefore find small, stylistic differences that lead to very different *dependency* trees, yet have almost identical *constituency* trees. For example, leaving out the serial comma is one of the most common stylistic errors (Onwuegbuzie et al., 2010), and therefore web texts may use this comma less consistently than copy-edited news text. Table 1a and 1b show constituency trees and their conversion to dependencies for the phrases *Digital, Hewlett and Sun* and *Digital, Hewlett, and Sun*. When using the Stanford converter, inserting or deleting the comma (which is not even a token which gets scored) causes the parent of *all five surrounding tokens* to flip. For the pennconverter conversions, which have chosen to always use the leftmost conjunct, the comma has no effect on the parents of the surrounding words.

The serial comma is not the only factor that can change the head of a list. Contrast the right-headed 1b with the left-headed 1c. In 1c, "Home Depot" has two words, and so in the gold standard constituency tree all items are noun phrases. If we remove the two unary production rules $NP \rightarrow NNP$ from the constituency tree (1d), the Stanford dependency tree shifts completely to a right-headed list again. The length in words of all other list items affects whether another item should attach to the first or last item. Under the pennconverter representation, neither the serial comma nor the unary productions affect the resulting dependency tree.

[3] http://nlp.cs.lth.se/software/treebank_converter/

	Q&A		Newsgroups		Reviews		Weblogs		Emails		WSJ	
Conj	66.6	(1071)	79.5	(624)	72.9	(1103)	81.3	(594)	71.4	(674)	85.9	(892)
Sib to Conj	75.7	(3362)	80.7	(1755)	77.9	(3473)	80.0	(1721)	74.8	(2058)	85.8	(2638)
Not Conj, Not Sib	83.5	(21562)	86.6	(15755)	84.0	(20597)	87.6	(15639)	81.0	(22808)	92.6	(31030)

Table 2: Unlabeled attachment accuracy of the baseline parser on conjunctions and words which are siblings to them in the gold standard trees compared with words which are not. Conjunctions are not just less accurate – their neighborhoods are too. The number of examples is given in parenthesis.

	Q&A		Newsgroups		Reviews		Weblogs		Emails		WSJ	
Conj	70.9	(1071)	79.8	(624)	78.1	(1103)	83.3	(594)	76.3	(674)	88.6	(892)
Sib to Conj	84.4	(1933)	85.8	(888)	85.9	(2004)	83.1	(943)	80.8	(1092)	89.6	(1432)
Not Conj, Not Sib	85.3	(22991)	87.3	(16622)	85.0	(22066)	88.9	(16417)	82.2	(23774)	92.5	(32236)

Table 3: Same as above, with pennconverter dependencies

3 Experiments: Conjunctions and their Local Neighborhoods

In the previous section, we saw qualitative examples of how very small changes in the constituency trees can lead to a large number of edges changing in dependency trees. In the examples in Table 1, not only did the parent of the conjunction change, but the edges to all of the conjunction's siblings changed as well. As the dependency trees are fairly bushy, there may be a large number of siblings per conjunction. Getting the conjunction's parent wrong may make several other edges wrong as well. We quantify this relationship in this section, by examining how well a baseline non-adapted parser does on conjunctions, words which are siblings of conjunctions in the gold tree, and words which are neither conjunctions nor siblings to a conjunction.

3.1 Baseline Parser

We use the Koo and Collins (2010) third-order grandparent-sibling parser as our starting point. We use margin-based pruning using marginals produced by a first-order parser (Collins et al., 2008). The parser was trained for 15 iterations of averaged perceptron training; we chose the model from the iteration with the highest unlabeled attachment accuracy on the emails development set as the final model. Part-of-speech tagging was done using MXPOST (Ratnaparkhi et al., 1994), with 10-fold cross validation used to produce automatic tags for the training set and the full training set used to produce tags for the development and test sets. No adaptation was done for the tagger. Labels were added to the predicted edges in a post-processing step using a multiclass SVM classifier[4] with features similar to those described in McDonald et al. (2006). Our focus in this paper is on unlabeled attachment accuracy.

3.2 Results on Conjunctions and Local Neighborhoods

Table 2 shows the unlabeled attachment accuracy of conjunctions, their siblings (in the gold tree), and words which are neither conjunctions nor siblings to one on the test sets of the five domains in the Google Web Treebank (Q&A, newsgroups, reviews, weblogs, emails) and the source domain (WSJ). The accuracy on conjunctions is substantially lower than non-conjunctions — for example in Q&A, the conjunction accuracy is 66.6%, while the non-conjunction/non-sibling words are attached correctly at a rate of 83.5%. It is not just conjunctions that drop in accuracy though; across domains, the siblings of conjunctions are also substantially lower than words which are not siblings of conjunctions. While conjunctions themselves do not make up too large a proportion of the test set, their sphere of influence (which appears to include at least their siblings) affects around 15% of total scored tokens.

As a comparison, Table 3 shows the same breakdown when dependencies for training and testing are produced using pennconverter to convert from constituency trees. While accuracies on non-conjunctions/non-siblings are similar across the two conversions, conjunctions and their siblings are

[4] http://svmlight.joachims.org/svm_multiclass.html

higher under the pennconverter dependencies across all domains. The accuracies of the siblings of conjunctions are now more similar to the accuracies of non-conjunctions than they are to the conjunction accuracies for all domains except weblogs and WSJ.

4 Attempts to Improve Conjunctions

4.1 Graph Transformations for Conjunctions

If a particular representation is difficult to learn on, one possibility is to deterministically transform the representation into one more easily learnable, learn a parser in that space, and then invert the transformation to produce final predictions. This has been done within constituency parsing (Collins, 1999) and for conjunctions within dependency parsing for Czech (Nilsson et al., 2006). We identified all siblings of conjunctions with a *conj* edge label, moved these to become children of the conjunction, trained a parser, and then after parsing, moved all children of conjunctions up to a sibling position. This led to overall lower results. One reason was an unexpected effect on hyphenated words. For example, without transformations, *orange-and-blue* (tokenized as *orange - and - blue* in the training set) has *blue* as the parent of every other token, and the only edge with a gold-standard label of *conj* is the edge into the second hyphen. The result is that the conjunction appears to be between the second hyphen and *blue*.

4.2 Parser Stacking with Different Converter

Given the higher accuracy of attaching conjunctions under *pennconverter* dependencies (Table 3), we attempted to use parser stacking (Martins et al., 2008; McDonald and Nivre, 2011) to improve the accuracies of conjunctions. We first trained our parser on the pennconverter dependencies, then used the first parser's predictions as additional features when training the parser on Stanford dependencies. The hope was that the second stage parser could learn the transformations required between the two conversions (i.e., conjunctions are represented as a sibling relationship in one and a grandparent relationship in the other).

For each potential edge (h, m), we included features for whether h (or m) was predicted by the first stage parser as m's (h's) parent, grand-parent, great-grandparent, or siblings of any of the above, conjoined with the part-of-speech tags of h and m and whether there was any conjunction along the path between h and m in the predicted pennconverter tree. We also included more general features for whether h was the ancestor, descendant, or neither of m, also conjoined with their POS tags. Development experiments showed these were harmful for attaching parenthesis, so these were only used for non-parenthesis tokens; other parenthetical specific features were introduced for parenthesis tokens.

For both the first and second stage parsers, we used features based on the pointwise mutual information between pairs and triples of words occurring with conjunctions or prepositions, using counts derived from the unlabeled data provided in the 2012 Shared Task. These features were identical to the Google n-gram features shown to help prepositions and conjunctions within parsing in Pitler (2012).

The stacked parser was the submitted system, which performed only slightly better than the baseline parser on overall attachment accuracy and (surprisingly) no better than the baseline parser on conjunctions. Error analysis showed that the distance and relationship between the conjoined items (according to the Stanford converter) in the pennconverter tree varied widely, based on various other differing design choices between the two conversion systems. For example, the pennconverter system chooses auxiliaries as the heads, while the Stanford converter chooses lexical verbs as heads.

5 Conclusion

There is no clear consensus about how conjunctions should be represented in dependency trees. While there are several possible plausible alternatives, we expect that dependency trees with the below two properties will be easier to learn from:

1. Uniform decisions about left or right heads
2. Conjuncts directly connected to the conjunction with an edge

It proved harder than expected to mitigate the lack of these properties within a dependency parser. Lessening the reliance of constituency-to-dependency converters (and parsers training on derived dependency trees) on stylistic cues may lead to more robust parsing on less edited web text.

References

M. Collins, A. Globerson, T. Koo, X. Carreras, and P.L. Bartlett. 2008. Exponentiated gradient algorithms for conditional random fields and max-margin markov networks. *The Journal of Machine Learning Research*, 9:1775–1822.

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

M.C. De Marneffe and C.D. Manning. 2008. The stanford typed dependencies representation. In *Coling 2008: Proceedings of the workshop on Cross-Framework and Cross-Domain Parser Evaluation*, pages 1–8. Association for Computational Linguistics.

M.C. De Marneffe, B. MacCartney, and C.D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of LREC*, volume 6, pages 449–454.

Mark Dredze, John Blitzer, Partha Pratim Talukdar, Kuzman Ganchev, João Graca, and Fernando Pereira. 2007. Frustratingly hard domain adaptation for dependency parsing. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*, pages 1051–1055.

R. Johansson and P. Nugues. 2007. Extended constituent-to-dependency conversion for English. In *Proc. of the 16th Nordic Conference on Computational Linguistics (NODALIDA)*, pages 105–112.

T. Koo and M. Collins. 2010. Efficient third-order dependency parsers. In *Proceedings of ACL*, pages 1–11.

A.F.T. Martins, D. Das, N.A. Smith, and E.P. Xing. 2008. Stacking dependency parsers. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 157–166.

R. McDonald and J. Nivre. 2011. Analyzing and integrating dependency parsers. *Computational Linguistics*, 37(1):197–230.

Ryan McDonald, Kevin Lerman, and Fernando Pereira. 2006. Multilingual dependency analysis with a two-stage discriminative parser. In *Proceedings of the Tenth Conference on Computational Natural Language Learning*, pages 216–220.

J. Nilsson, J. Nivre, and J. Hall. 2006. Graph transformations in data-driven dependency parsing. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 257–264. Association for Computational Linguistics.

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 *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*, pages 915–932.

A.J. Onwuegbuzie, J.P. Combs, J.R. Slate, and R.K. Frels. 2010. Editorial: Evidence-based guidelines for avoiding the most common apa errors in journal article submissions. *Research in the Schools*, 16(2).

Slav Petrov and Ryan McDonald. 2012. Overview of the 2012 shared task on parsing the web. In *Notes of the First Workshop on Syntactic Analysis of Non-Canonical Language (SANCL)*.

E. Pitler. 2012. Attacking parsing bottlenecks with unlabeled data and relevant factorizations. In *Proceedings of ACL*.

A. Ratnaparkhi, J. Reynar, and S. Roukos. 1994. A maximum entropy model for prepositional phrase attachment. In *Proceedings of the workshop on Human Language Technology*, pages 250–255. Association for Computational Linguistics.

R. Weischedel, M. Palmer, M. Marcus, E. Hovy, S. Pradhan, L. Ramshaw, N. Xue, A. Taylor, J. Kaufman, M. Franchini, et al. 2011. Ontonotes release 4.0. *LDC2011T03, Philadelphia, Penn.: Linguistic Data Consortium*.