Sentence diagramming by non-experts: as good as Treebank?

Brendan O'Connor
Carnegie Mellon University
brenocon@cmu.edu

Introduction

The last 10-15 years have seen enormous advances in the accuracy and usefulness of syntactic parsing, specifically in the form of stochastic parsers whose models are fit via supervised learning from an annotated treebank. For English, the most commonly used resource (by far) is the Penn Treebank, which consists of constituent tree annotations for 40,000 sentences from the Wall Street Journal, collected in the early 90's (Marcus, Santorini, and Marcinkiewicz 1994).

While there has been an incredible amount of work developing parsers for this dataset, there has been very little work on creating new syntactically annotated corpora, in part because the task is perceived as very difficult. If the goal of research in syntactic analysis is to create effective parsers, this imbalance of data collection vs. algorithmic research is troubling. First, the WSJ's genre of English may be overly narrow to support the goal of creating broad-coverage English parsers – for example, the corpus contains very few questions. Second, it may be the case that creating more training data will be the simplest route toward creating better syntactic parsers.

We are interested in exploring whether it is possible to have online annotators provide syntactic annotations rich enough to support training a parser. If so, and costs can be reduced to the range of $1/sentence, then it would be very feasible to create corpora of similar size to the Penn Treebank, and also may additionally become possible to conduct small annotation projects for new domains when necessary. This would have enormous utility for natural language processing, given that errors in syntactic analysis are often a bottleneck in applied systems.

Unlabeled dependencies and sentence diagrams

The Penn Treebank's formalism, while simplified compared to earlier annotation work at the time, is forbiddingly complicated to the uninitiated. Its official annotation guidelines weigh in at 318 pages. This approach clearly requires the hiring full-time annotators who must undergo extensive training.

However, if the goal of syntactic annotations is to support better parsing, recent work suggests a much simpler formalism may suffice: unlabeled dependencies. Recent work has focused on dependency parses, which have more practical utility compared to constituency parses for a variety of applications (de Marneffe and Manning 2008). Furthermore, statistical dependency parsers are usually trained by converting the Penn Treebank to dependencies through a set of deterministic rules (e.g. (Johansson and Nugues 2007)); and furthermore, some systems then discard edge labels, training directly on the unlabeled dependency graph (plus part-of-speech tags) — this technique is used in parsers including MSTParser and TurboParser (McDonald et al. 2005; Martins, Smith, and Xing 2009). The key insight is that the hard part of parsing is getting attachment decisions correct; after that, basic grammatical relations flow more or less deterministically from the part-of-speech tags and word-word dependency relations.

This suggests that, if part-of-speech tags are already known, the only information needed from human annotators are unlabeled dependencies. We believe these are fairly intuitive for many literate readers of English to understand and derive.

Furthermore, there already exists a dependency formalism known to thousands of people around the world: *sentence diagrams*. This system was originally developed in the 19th century to assist in teaching English grammar in primary and secondary school education (Kellogg and Reed 1877).

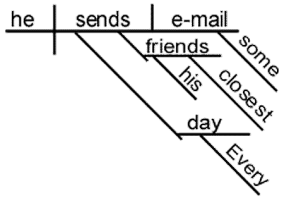

Figure 1: Sentence diagram for *Every day, he sends his closest friends some email.*

An example is shown in Figure 1. It consists of roughly typed dependency information, depicted in a visually pleasing manner that is easy to draw by hand. The top platform is a subject/verb/direct object triple. Every word can have dependent words and phrases, which are drawn down and to the right of the governor word. Note that for this particu-

lar example, the expressed dependency graph is identical to the standard one that would be used to train a dependency parser; and is in fact slightly richer, since it differentiates between subject and object of the verb.

Within primary and secondary education, sentence diagramming has declined in popularity in recent decades, but is still somewhat widespread in parochial (e.g. Catholic) schools, and is occasionally seen in foreign language learning. There is also interesting anecdotal evidence of people who enjoy sentence diagramming itself; for example, college students who begged an English professor to teach a diagraming course,[1] or the existence of numerous websites about sentence diagramming. The famous writer Gertrude Stein said:

> *I really do not know that anything has ever been more exciting than diagramming sentences.*

The quote appears in a recent popular history of sentence diagramming (Florey 2006); it's easy to find hundreds of online comments about the book by people who profess a love of sentence diagramming.

In the course of research, we manually diagrammed dozens of sentences, and found it to be much more enjoyable than drawing constituency trees. We suspect this is because dependencies are a more naturally understandable abstraction. Indeed, since the formalism was developed for and successfully used in language education — learned by many hundreds of thousands of people — it should be expected that it has design qualities that are especially friendly to the annotator. By contrast, the Treebank formalism is known to no more then a few dozen people worldwide, who perform the task only when paid to do so. It stands to reason that diagramming is a more annotator-friendly system; and, if the two formalisms are equally as good for training parsers, then diagramming is a far superior strategy for syntactic annotation.

It is not clear to what extent the usual sentence diagramming formalism is the best one to use. For example, a recent sentence diagramming textbook (Kolln and Funk 2005) defines very rich syntactic relations including traces, handling of prepositions in fixed expressions, and coordination.[2] We believe our goal should be to derive something similar to an unlabeled version of the dependencies derived by one of the standard Treebank-to-dependency converters as currently applied to gold Penn Treebank parses. At the simplest extreme, we could create a tool to allow annotators to create words and arrows between them. However, we feel it is easier to learn and work with something more like diagramming, which has easy-to-explain rules for where verbs, modifiers, and other parts of speech should go. Their different visual representations make it easier to read and clarify to the annotator the implications of their in-progress syntactic analysis.

This project consists of (1) defining an easy-to-learn formalism that is sufficiently rich to either train a useful dependency parser, or provide information from which dependency annotations can be inferred; (2) developing an annotation tool for it, and (3) testing it out with labmates and then Turkers.

For the third point, we have already conducted a survey on Mechanical Turk, simply asking if anyone had previous experience with sentence diagramming in English. We received about 80 positive responses from workers in the U.S. and India, who described learning them in educational experiences consistent with the history of sentence diagramming as explained above. Many respondents expressed interest in trying out a prototype sentence diagramming HIT, and even, when asked, provided e-mail addresses for us to contact them. (In the longer term, sentence diagramming annotation could be made useful to the annotators as a language education technology.)

The first two points, however, are prerequisite and the largest portion of this project. We feel that the quality of the GUI annotation tool, plus its accompanying guidelines and examples, will be a *huge* factor in whether this type of annotation will work.

There is very little previous work on sentence diagrams and natural language processing; the only work we know of is (Mayfield 2009), which implements and describes a tool that converts dependency parses into sentence diagrams for visualization and inspection purposes. This proposal is to go in the opposite direction.

[1] `http://somanybooksblog.com/2009/05/12/sentence-diagrams/`

[2] One online resource for a similar system: `http://www.geocities.com/gene_moutoux/diagrams.htm`

References

de Marneffe, M. C., and Manning, C. D. 2008. The stanford typed dependencies representation. In *Coling 2008: Proceedings of the workshop on Cross-Framework and Cross-Domain Parser Evaluation*, 18.

Florey, K. B. 2006. *Sister Bernadette's Barking Dog: The Quirky History and Lost Art of Diagramming Sentences*. Melville House, first edition.

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

Kellogg, B., and Reed, A. 1877. *Higher Lessons in English*.

Kolln, M. J., and Funk, R. W. 2005. *Understanding English Grammar*. Longman, 7 edition.

Marcus, M. P.; Santorini, B.; and Marcinkiewicz, M. A. 1994. Building a large annotated corpus of english: The penn treebank. *Computational linguistics* 19(2):313330.

Martins, A. F. T.; Smith, N. A.; and Xing, E. P. 2009. Concise integer linear programming formulations for dependency parsing.

Mayfield, E. 2009. Sentence diagram generation using dependency parsing. *ACL-IJCNLP* 45.

McDonald, R.; Pereira, F.; Ribarov, K.; and Hajic, J. 2005. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, 523530.