Linguistic Structure Prediction with the Sparseptron

Recent advances in natural language processing bring together rich representations and scalable machine learning algorithms.

By Noah A. Smith and André F. T. Martins

DOI: 10.1145/2425676.2425690

"*Charlotte's Web is a children's novel by American author E. B. White, about a pig named Wilbur who is saved from being slaughtered by an intelligent spider named Charlotte.*" When reading this sentence from Wikipedia, xkcd webcomic artist Randall Munroe interpreted Charlotte as a would-be killer.[1] While reasonable, his reading contradicts White's story. Such failures give rise to humor, as in the comic this sentence inspired Munroe to draw, but also to confusion, especially for automated systems. Natural languages (NLs), like English, are full of ambiguity (unlike programming languages, which are designed to be unambiguous). Disambiguating NL strings is a challenge not just of algorithm design, but of artificial intelligence.

AN EXAMPLE OF NL DISAMBIGUATION

One of the key problems of natural language processing (NLP) is the disambiguation of sentences through the design of functions that map NL strings to "deeper," less ambiguous structures. Many techniques exist for this kind of linguistic structure prediction [1]. In this article, our running example is one that draws on algorithms familiar to many computer scientists: dependency parsing. In explaining how dependency parsing works, we will highlight some recent research advances in NLP and machine learning.

Dependency parsing is based on linguistic theories that model the relationships among words in a sentence [2]. Figure 1 shows two different dependency analyses for the phrase "by an intelligent spider named Charlotte." In the first, *by* attaches to *slaughtered*, corresponding to Munroe's comical read-

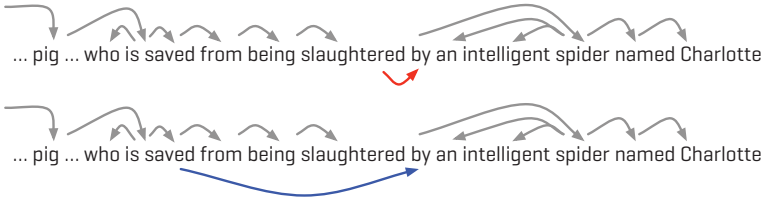
Image courtesy of Randall Munroe. http://xkcd.com/1087/

1 Munroe titled the comic "Cirith Ungol," which is derived from Tolkien's *Lord of the Rings*. It is Elvish for "Pass of the Spider."

ing. The second attaches *by* to *saved*, as presumably intended by the sentence's author since this reading aligns with the story. A complete dependency analysis attaches every word to a parent, which is usually a word to which it serves as an argument or modifier. The analysis often includes labels for each of these word-to-word relationships.

One view of dependency parsing, first suggested by McDonald et al. [3], is that the words form the vertices of a graph, and each possible dependency tree is an arborescence (i.e., a directed spanning tree). If we assign a score to each tree by the sum of the scores of the arcs it includes, then parsing equates to finding the maximum arborescence of a weighted, directed graph. The maximum arborescence problem is solvable in quadratic time in the number of vertices (here, the length of the sentence) [4, 5, 6] (see Figure 2 for an illustration). Many other parsing approaches are available, often based on dynamic programming [7] or integer linear programming [8]. (Our own open-source dependency parser, TurboParser, uses a discrete optimization technique called "alternating directions dual decomposition" [9].)

Figure 1. Two different dependency analyses for the phrase *"by an intelligent spider named Charlotte."* The first corresponds to Randall Munroe's reading and to that found by our parser, TurboParser; the second corresponds to the intended meaning.

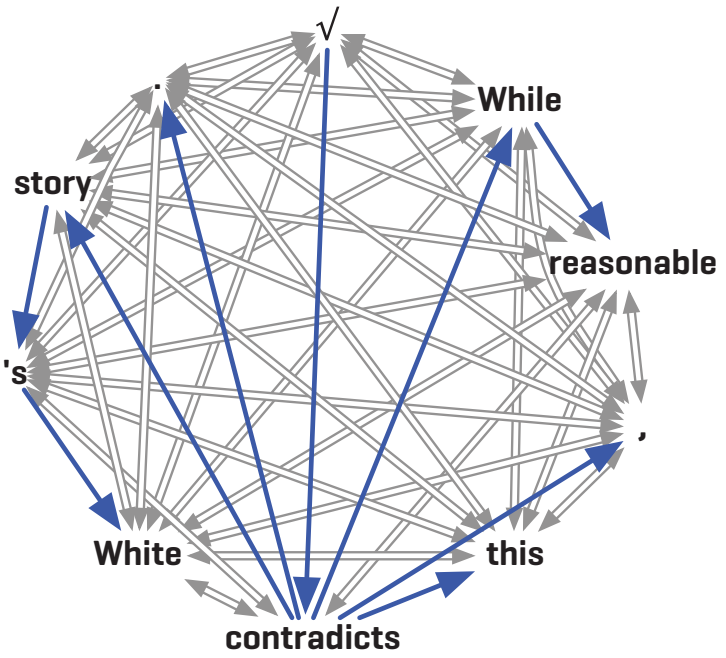


A common theme is assigning scores to trees by adding up the scores of the tree's parts—here, arcs in a dependency tree—and finding a tree with a maximal total score. From here on, we will assume the availability of two functions: ScoreTreeParts, which maps tree parts to real-valued scores, and FindBestTree, which maps an NL string and a ScoreTreeParts function to the maximum-scoring tree using a discrete optimization algorithm (e.g., the maximum arborescence algorithm).

LEARNING TO SCORE TREES

Where does ScoreTreeParts come from? The dominant approach today is to use machine learning to learn how to score tree parts from datasets of examples annotated by linguists. A fa-

Figure 2. A graph showing all possible attachments (edges) between all words (vertices) in the sentence *"While reasonable, this contradicts White's story."* The correct dependency parse, and the one produced by our parser, TurboParser, is shown in blue. The extra √ symbol is used to fix the root of the arborescence.



mous example of such a dataset is the Penn Treebank, a set of 40,000 sentences from a collection of 1980s *Wall Street Journal* articles annotated with phrase structures [10]. There are now a range of treebanks built for additional languages, and they use a range of different linguistic representations [11, 12].

Learning a good ScoreTreeParts function hinges on combining many different kinds of evidence that reveal which candidate tree parts are likely to be part of a correct analysis of a sentence. Grammatical patterns, like the tendency of adjectives to be modifiers of nouns, provide one kind of evidence that can be used if words in the sentence to be parsed have been assigned parts of speech (typically accomplished by a different linguistic structure predictor called a "part-of-speech tagger"). Lexical patterns that depend on specific words are another kind of evidence. For example, a likely subject of the word *slaughter* is *butcher*, while *spider* is less likely. Some patterns are language-specific (e.g., in English, adjectives tend to attach to nouns on their right, while prepositions tend to attach to words on their left) or general (e.g., words tend to attach to parents that are not too far away in the string).

These kinds of evidence are called "features." If we think of each feature as a function that takes a value of 0 (absence) or 1 (presence) for each possible dependency arc, then we can define ScoreTreeParts as a function of an arc a, using the vector of a's feature values, $\mathbf{f}(a)$:

$$\text{ScoreTreeParts}(a, \mathbf{w}) = \mathbf{f}(a) \cdot \mathbf{w}$$

In this expression, $\mathbf{w}$ is a vector of weights that relate each feature to the strength of the arc. The weight for the "adjective attaches to noun" feature is probably going to be positive, giving a bonus to trees that include such an arc, while the weight for "the parent and child are more than 15 string positions apart" is probably going to be negative, penalizing trees with longer attachments.

It should be clear that the quality of the trees our algorithm produces depends heavily on the choice of features. Where do good features come from? There are three dimensions to consider when designing features. One is linguistic theory. The study of language, from ancient grammarians like Pāṇini and Aristotle to modern structuralist and generative linguists, has much to say about the syntax of NL strings, some of which informs the design of features. Linguistic theory tells us, for example, that certain kinds of tree parts are required to model certain kinds of phenomena (e.g., coordinating conjunctions, such as *and*, usually take left and right arguments that are of the same category, like verbs or nouns) and that some kinds of attachments are obligatory or ungrammatical (e.g., intransitive verbs, like *rain*, do not have an argument on the right). The second concern is computational cost. There is a trade-off between the computational complexity of FindBestTree algorithms and the size of the tree parts scored by ScoreTreeParts. Note that above, we score

Figure 3. The sparseptron algorithm.

input: dataset D, number of iterations T, feature groups $\{F_g\}$, and group budget B
output: weight vector $\mathbf{w}$ with at most B active groups
initialize $\mathbf{w} = \mathbf{0}$
for $t = 1$ to T:
  from D, select a sentence x and its parse y
  $z \leftarrow$ FindBestTree[x, ScoreTreeParts[$\cdot$, $\mathbf{w}$]]
  set step size $\eta \leftarrow t / |D|^{-1/2}$
  update $\mathbf{w} \leftarrow \mathbf{w} + \eta \cdot [\sum_{a \in y} \mathbf{f}(a) - \sum_{a \in z} \mathbf{f}(a)]$
  divide each group-wise subvector of $\mathbf{w}$ by $\log_2 |F_g|$ and sort the result by decreasing L_2 norms:
    $||\mathbf{w}_1|| / (\log_2 |F_1|) \geq ||\mathbf{w}_2|| / (\log_2 |F_2|) \geq \ldots$
  set $\sigma \leftarrow [||\mathbf{w}_B|| / (\log_2 |F_B|) + ||\mathbf{w}_{B+1}|| / (\log_2 |F_{B+1}|)] / 2$
  for each group g:
    set $\mathbf{w}_g \leftarrow \max\{0, (||\mathbf{w}_g|| - \sigma \log_2 |F_g|) / ||\mathbf{w}_g||\} \cdot \mathbf{w}_g$
return $\mathbf{w}$

single arcs; if we scored all pairs of contiguous arcs, then FindBestTree would correspond to an NP-hard problem [13]. Finally, we have the statistical challenge of using data to select the coefficients **w**. Again, we have a tradeoff: More complex features will occur at lower frequencies in data, and so more data will be required to estimate the coefficients well. Similarly, many features suggested by theory or the data may in fact not be generally helpful, and we would like to eliminate them by setting their coefficients to zero. The remainder of this article is about a learning algorithm designed specifically for selecting the coefficients **w** in linguistic structure prediction problems.

THE SPARSEPTRON

Machine learning offers a wide range of algorithms for problems like linguistic structure prediction. For language-related learning, we require a learning algorithm with a few key properties. First, it must be able to take advantage of high-dimensional input, since dependency parsers tend to make use of millions of features. Second, we would like learning to be fast. While there is no simple way to select values of **w** that will perform well, there are iterative algorithms that have provable convergence properties. The algorithm we describe here will converge to a solution whose objective value differs by at most ε from the optimal model (for a given set of features), known as an ε-accurate solution, in $O(1/\varepsilon^2)$ iterations over the training data. Third, we want the learned coefficients to achieve high accuracy not just on the training data, but also to generalize well to new examples to which it has not been exposed.

Generalization is a special challenge in NLP. This is because language varies so much with the situation in which it is used, the individual speaker, and the matter being talked about. When working with very high-dimensional inputs, one compelling approach to good generalization is to aim for sparsity: Since many features in ScoreTreeParts correlate with each other or are not needed for good performance, we hope for their coefficients to be set at zero (i.e., **w** becomes a sparse vector). Well-known in statistics [14], sparse learners have received a great deal of attention in recent machine learning [15, 16, 17, 18, 19]. Here we go farther, seeking sparsity among whole groups of features. For example, one of our smaller datasets consists of trees in Slovene, a language with rich inflective morphology (i.e., words change their form depending on grammatical gender, number, case, etc.). We do not expect the words themselves to serve as reliable cues for Slovene dependency attachment decisions, since most words in the training data will be seen only in one or a few contexts. (This is less of a problem in a language with less inflective morphology, like English, which has no grammatical gender or case for nouns.)

Dependency parsing is based on linguistic theories that model the relationships among words in a sentence.

We assume here that the features are partitioned into non-overlapping groups, indexed by g. (Our algorithm generalizes to overlapping groups [20], but we consider the simpler non-overlapping case here for clarity.) Let F_g denote the set of features included in group g, and let $\mathbf{w}_g$ denote the subvector of coefficients for F_g. Our algorithm, the sparseptron, is shown in Figure 3.

On each iteration, the sparseptron algorithm considers one training sentence and updates the weights based on the difference between the correct parse y and the current prediction z. This update is equivalent to the classic perceptron [21], adapted for structured problems by Collins [22]. After the update, a transformation on the weights explicitly moves groups of weights to zero to respect the budget B.

DISCUSSION

We have given a specific instance of an online sparse proximal-gradient algorithm, which can handle a wide range of alternative loss functions and regularizers. Our instance is based on the perceptron, which minimizes the "hinge" loss, and a group-L_1 regularizer for group sparsity. Other special cases are reported by Martins et al. [20]; one is the "truncated gradient" algorithm of Langford et al. [23], which corresponds to a standard L_1 regularizer. This variant is notable because it is famous: sometimes called the "lasso" [14], it leads to sparse—but not group-sparse—coef-

Table 1: Selected dependency parsing results, $B = 400$. The "state of the art" column represents the best published scores on this task across a range of systems including our own and those of Koo et al. [25], many using complex tree parts in SCORETREEPARTS, and generally not inducing sparsity. Variation across languages is due to the inherent properties of the language, the idiosyncratic style of the text data selected for Treebank annotation, and the amount of data annotated. For comparison, the best reported accuracy for the English Penn Treebank task is 93 percent.

Language	Attachment Accuracy (%)	Accuracy, Compared to Non-Sparse Model (%)	Nonzero Coefficients (%)	State of the Art (%)
Arabic	78.2	+0.1	59	81.1
Danish	89.9	0.0	48	91.9
Japanese	93.1	+0.2	22	93.7
Slovene	83.2	0.0	49	87.0
Spanish	83.0	-0.9	47	87.0
Turkish	75.6	+0.3	46	77.6

ficient vectors. To simulate L_1 regularization with the sparseptron, simply assign each feature to its own group.

We recommend using the sparseptron as a first pass for selecting feature groups, followed by warm-start learning with a strong cost-aware (but non-sparse) learning algorithm such as the margin-infused relaxation algorithm (MIRA) [24] on only the selected feature groups; this second stage is known in statistics as "debiasing."

To measure the accuracy of a linguistic structure predictor, we compare its output with a gold standard dataset similar to the training data. By ensuring this test dataset is distinct from the training data, we test the generalization ability of our predictor. We applied the sparseptron with debiasing to several NLP problems and compared its accuracy to the state-of-the-art MIRA [24]. Selected results for dependency parsing are shown in Table 1, using an arc-scoring ScoreTreeParts function. In addition, the sparseptron achieved:

- On an English text chunking task, indistinguishable accuracy from MIRA, with 3 percent as many features, using $B = 20$.
- On a named entity recognition task in English, Dutch, and Spanish, strictly better accuracy than MIRA and a sparse L_1-regularized model, with 4–11 percent as many features, using $B = 200$.

For more detailed experimental results, see Martins et al. [20].

CONCLUSIONS

Disambiguating NL text is central to developing applications that can summarize, translate, answer questions, and extract structured information. Recent advances in NLP have hinged on rich representations (e.g., millions of features of local disambiguation decisions suggested by humans, as explored here) and sophisticated, scalable learning algorithms that make the most of those representations, including the elimination of features unhelpful for generalization. Learning algorithms that allow the use of rich linguistic features, but can infer from data how to eliminate whole groups of unnecessary ones, can lead to accurate and efficient disambiguation.

ACKNOWLEDGMENTS

We acknowledge our collaborators on the sparseptron and TurboParser, Pedro Aguiar, Mário Figueiredo, and Eric Xing. A. M. was supported by a FCT/ICTI grant through the CMU-Portugal Program, and also by Priberam. N. S. was supported by NSF CAREER IIS-1054319. This work was partially supported by the FET programme (EU FP7), under the SIMBAD project (contract 213250).

References

[1] Smith, N. A. *Linguistic Structure Prediction*. Morgan and Claypool, 2011.

[2] Tesnière, L. *Éléments de syntaxe structurale*. Klincksieck, 1959.

[3] McDonald, R., Pereira, F., Ribarov, K., and Hajič, J. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing* (Vancouver, Oct. 6-8). Association for Computational Linguistics, Stroudsburg, PA, 2005, 523–530.

[4] Chu, Y. J. and Liu, T. H. On the shortest arborescence of a directed graph. *Science Sinica* 14 (1965), 1396–1400.

[5] Edmonds, J. Optimum branchings. *Journal of Research of the National Bureau of Standards* 71B, 4 (1967), 233–240.

[6] Tarjan, R. E. Finding optimum branchings. *Networks* 7, 1 (1977), 25–35.

[7] Eisner, J. Three new probabilistic models for dependency parsing: an exploration. In *Proceedings of the 16th Conference on Computational Linguistics* (Copenhagen, Denmark, Aug. 5-6). Association for Computational Linguistics, Stroudsburg, PA, 1996, 340–345.

[8] Martins, A. F. T., Smith, N. A., and Xing, E. P. Concise integer linear programming formulations for dependency parsing. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing* (Singapore, Aug. 2-7). Association for Computational Linguistics, Stroudsburg, PA, 2009, 342–350.

[9] Martins, A. F. T., Smith, N. A., Aguiar, P. M. Q., and Figueiredo, M. A. T. Dual decoposition with many overlapping components. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing* (Edinburgh, July 27-31). Association for Computational Linguistics, Stroudsburg, PA, 2011, 238–249. TurboParser download: http://www.ark.cs.cmu.edu/TurboParser/

[10] Marcus, M. P., Santorini, B., and Marcinkiewicz, M. A. Building a large annotated corpus of English: The Penn treebank. *Computational Linguistics* 19, 2 (1993), 313–330.

[11] Buchholz, S. and Marsi, E. The CoNLL-X shared task on multilingual dependency parsing. In *Proceedings of the Conference on Computational Natural Language Learning* (New York, June 8-9). Association for Computational Linguistics, Stroudsburg, PA, 2006, 149–164.

[12] Nivre, J., Hall, J., Kübler, S., McDonald, R., Nilsson, J., Riedel, S., and Yuret, D. The CoNLL 2007 shared task on dependency parsing. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, (Prague, June 28-30). Association for Computational Linguistics, Stroudsburg, PA, 2007, 915–932.

[13] McDonald, R. and Satta, G. On the complexity of non-projective data-driven dependency parsing. In *Proceedings of the International Conference on Parsing Technologies* (Prague, June 23-24). Association for Computational Linguistics, Stroudsburg, PA, 2007, 121–132.

[14] Tibshirani, R. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, Series B (Methodological)* 58, 1 (1996), 267–288.

[15] Guyon, I. and Elisseeff, A. An introduction to variable and feature selection. *Journal of Machine Learning Research* 3 (March 2003), 1157–1182,.

[16] Kazama, J.; Tsujii, J. Evaluation and extension of maximum entropy models with inequality constraints. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing* (Sapporo, Japan, July 11-12). Association for Computational Linguistics, Stroudsburg, PA, 2003, 137–144.

[17] Goodman, J. Exponential priors for maximum entropy models. In *Proceedings of Annual Meeting of the North American Chapter of the Association for Computational Linguistics, HLT-NAACL 2004* (Boston, May 2-7). Association for Computational Linguistics, Stroudsburg, PA, 2004, 305–312.

[18] Jenatton, R., Audibert, J.-Y., ANDBach, F. Structured variable selection with sparsity inducing norms. Technical report. arXiv:0904.3523. 2009.

[19] Wright, S., Nowak, R., and Figueiredo, M. A. T. Sparse reconstruction by separable approximation. I*EEE Transactions on Signal Processing* 57, 7 (2009), 2479–2493.

[20] Martins, A. F. T., Smith, N. A., Figueiredo, M. A. T., and Aguiar, P. M. Q. Structured sparsity for structured prediction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing* (Edinburgh, July 27-31). Association for Computational Linguistics, Stroudsburg, PA, 2011, 1500–1511.

[21] Rosenblatt, F. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review 65*, 6 (1958), 386-408.

[22] Collins, M. Discriminative training methods for hidden Markov models: theory and experiments with perceptron algorithms. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing* (Philadelphia, July 6-7). Association for Computational Linguistics, Stroudsburg, PA, 2002, 1–8.

[23] Langford, J., Li, L., and Zhang, T. Sparse online learning via truncated gradient. *Journal of Machine Learning Research* 10 (March 2009), 777–801.

[24] Crammer, K., Dekel, O., Keshet, J., Shalev-Shwartz, S., and Singer, Y. Online passive-aggressive algorithms. *Journal of Machine Learning Research* 7 (March 2006), 551–585.

[25] Koo, T., Rush, A. M., Collins, M., Jaakkola, T., and Sontag, D. Dual decomposition for parsing with non-projective head automata. In *Proceedings of the Conference on Empirical Methods for Natural Language Processing* (Cambridge, MA, Oct. 9-11). Association for Computational Linguistics, Stroudsburg, PA, 2010, 1288–1298.

Biographies

Noah A. Smith is the Finmeccanica Associate Professor of Language Technologies and Machine Learning in the School of Computer Science at Carnegie Mellon University. He received his Ph.D. in computer science, as a Hertz Foundation Fellow, from Johns Hopkins University in 2006 and his B.S. in computer science and B.A. in linguistics from the University of Maryland in 2001. His research interests include statistical natural language processing, especially unsupervised methods, machine learning for structured data, and applications of natural language processing.

André Martins is a research scientist in the Machine Learning Group at Priberam Labs, in Portugal. He received his dual Ph.D. in language technologies from Instituto Superior Técnico and Carnegie Mellon University, in 2012. His research interests include statistical natural language processing, machine learning for structured data, and optimization algorithms.

© 2013 ACM 1529-4972/13/03 $15.00