JMLR: Workshop and Conference Proceedings 21:189–194, 2012 The 11th ICGI

Bootstrapping Dependency Grammar Inducers from Incomplete Sentence Fragments via Austere Models

Valentin I. Spitkovsky VALENTIN@CS.STANFORD.EDU
Computer Science Department, Stanford University and Google Research, Google Inc.

Hiyan Alshawi HIYAN@GOOGLE.COM
Google Research, Google Inc., 1600 Amphitheatre Parkway, Mountain View, CA, 94043

Daniel Jurafsky JURAFSKY@STANFORD.EDU
Departments of Linguistics and Computer Science, Stanford University, Stanford, CA, 94305

Editors: Jeffrey Heinz, Colin de la Higuera and Tim Oates

Abstract

Modern grammar induction systems often employ curriculum learning strategies that begin by training on a subset of all available input that is considered simpler than the full data. Traditionally, filtering has been at granularities of whole input units, e.g., discarding entire sentences with too many words or punctuation marks. We propose instead viewing inter-punctuation fragments as atoms, initially, thus making some simple phrases and clauses of complex sentences available to training sooner. Splitting input text at punctuation in this way improved our state-of-the-art grammar induction pipeline. We observe that resulting partial data, i.e., mostly incomplete sentence fragments, can be analyzed using reduced parsing models which, we show, can be easier to bootstrap than more nuanced grammars. Starting with a new, bare dependency-and-boundary model (DBM-0), our grammar inducer attained 61.2% directed dependency accuracy on Section 23 (all sentences) of the Wall Street Journal corpus: more than 2% higher than previous published results for this task.

Keywords: Dependency Grammar Induction; Unsupervised Dependency Parsing; Curriculum Learning; Partial EM; Punctuation; Unsupervised Structure Learning.

1. Introduction

"Starting small" strategies (Elman, 1993) that gradually increase complexities of training models (Lari and Young, 1990; Brown et al., 1993; Frank, 2000; Gimpel and Smith, 2011) and/or input data (Brent and Siskind, 2001; Bengio et al., 2009; Krueger and Dayan, 2009; Tu and Honavar, 2011) have long been known to aid various aspects of language learning. In dependency grammar induction, pre-training on sentences up to length 15 before moving on to full data can be particularly effective (Spitkovsky et al., 2010a,b, 2011a,b). Focusing on short inputs first yields many benefits: faster training, better chances of guessing larger fractions of correct parse trees, and a preference for more local structures, to name a few. But there are also drawbacks: notably, unwanted biases, since many short sentences are not representative, and data sparsity, since most typical complete sentences can be quite long.

We propose starting with short inter-punctuation fragments of sentences, rather than with small whole inputs exclusively. Splitting text on punctuation allows more and simpler word sequences to be incorporated earlier in training, alleviating data sparsity and complex-

ity concerns. Many of the resulting fragments will be phrases and clauses, since punctuation correlates with constituent boundaries (Ponvert et al., 2010, 2011; Spitkovsky et al., 2011a), and may not fully exhibit sentence structure. Nevertheless, we can accommodate these and other unrepresentative short inputs using our dependency-and-boundary models (DBMs), which distinguish complete sentences from incomplete fragments (Spitkovsky et al., 2012).

DBMs consist of overlapping grammars that share all information about head-dependent interactions, while modeling sentence root propensities and head word fertilities separately, for different types of input. Consequently, they can glean generalizable insights about local substructures from incomplete fragments without allowing their unrepresentative lengths and root word distributions to corrupt grammars of complete sentences. In addition, chopping up data plays into other strengths of DBMs — which learn from phrase boundaries, such as the first and last words of sentences — by increasing the number of visible edges.

Figure 1: Three types of input: (a) fragments lacking sentence-final punctuation are always considered incomplete; (b) sentences with trailing but no internal punctuation are considered complete though unsplittable; and (c) text that can be split on punctuation yields several smaller incomplete fragments, e.g., *Bach's*, *Air* and *followed*. In modeling stopping decisions, *Bach's* is still considered left-complete — and *followed* right-complete — since the original input sentence was complete.

Odds and Ends	*"It happens."*	*Bach's "Air" followed.*
(*a*) An incomplete fragment.	(*b*) A complete sentence that cannot be split on punctuation.	(*c*) A complete sentence that can be split into three fragments.

2. Methodology

All of our experiments make use of DBMs, which are head-outward (Alshawi, 1996) class-based models, to generate projective dependency parse trees for Penn English Treebank's Wall Street Journal (WSJ) portion (Marcus et al., 1993). Instead of gold parts-of-speech, we use context-sensitive unsupervised tags,[1] obtained by relaxing a hard clustering produced by Clark's (2003) algorithm using an HMM (Goldberg et al., 2008). As in our original set-up without gold tags (Spitkovsky et al., 2011b), training is split into two stages of Viterbi EM (Spitkovsky et al., 2010b): first on shorter inputs (15 or fewer tokens), then on most sentences (up to length 45). Evaluation is against the reference parse trees of Section 23.[2]

Our baseline system learns DBM-2 in Stage I and DBM-3 (with punctuation-induced constraints) in Stage II, starting from uniform punctuation-crossing attachment probabilities (see Appendix A for details of DBMs). Smoothing and termination of both stages are as in Stage I of the original system. This strong baseline achieves 59.7% directed dependency accuracy — somewhat higher than our previous state-of-the-art result (59.1%, see also Table 1). In all experiments we will only make changes to Stage I's training, initialized from the same exact trees as in the baselines and affecting Stage II only via its initial trees.

1. http://nlp.stanford.edu/pubs/goldtags-data.tar.bz2:untagger.model
2. Unlabeled dependencies are converted from labeled constituents using deterministic "head-percolation" rules (Collins, 1999) — after discarding punctuation marks, tokens that are not pronounced where they appear (i.e., having gold part-of-speech tags `$` and `#`) and any empty nodes — as is standard practice.

Table 1: Directed dependency and exact tree accuracies (DDA / TA) for our baseline, experiments with split data, and previous state-of-the-art on Section 23 of WSJ.

	Stage I	*Stage II*	*DDA*	*TA*
Baseline (§2)	DBM-2	constrained DBM-3	59.7	3.4
Experiment #1 (§3)	split DBM-2	constrained DBM-3	60.2	3.5
Experiment #2 (§4)	split DBM-i	constrained DBM-3	60.5	4.9
Experiment #3 (§5)	split DBM-0	constrained DBM-3	61.2	5.0
(Spitkovsky et al., 2011b, §5.2)	constrained DMV	constrained L-DMV	59.1	—

Table 2: Feature-sets parametrizing dependency-and-boundary models three, two, i and zero: if *comp* is false, then so are $comp_{root}$ and both of $comp_{dir}$; otherwise, $comp_{root}$ is true for unsplit inputs, $comp_{dir}$ for prefixes (if $dir = \texttt{L}$) and suffixes (when $dir = \texttt{R}$).

Model		$\mathbb{P}_{\text{ATTACH}}$ *(root-head)*	$\mathbb{P}_{\text{ATTACH}}$ *(head-dependent)*	$\mathbb{P}_{\text{STOP}}$ *(adjacent/not)*
DBM-3	(Appendix A)	$(\diamond, \texttt{L}, c_r, comp_{root})$	$(c_h, dir, c_d, cross)$	$(comp_{dir}, c_e, dir, adj)$
DBM-2	(§3, Appendix A)	$(\diamond, \texttt{L}, c_r, comp_{root})$	(c_h, dir, c_d)	$(comp_{dir}, c_e, dir, adj)$
DBM-i	(§4, Appendix B)	$(\diamond, \texttt{L}, c_r, comp_{root})$	(c_h, dir, c_d)	$(comp_{dir}, c_e, dir)$
DBM-0	(§5, Appendix B)	$(\diamond, \texttt{L}, c_r)$ iff $comp_{root}$	(c_h, dir, c_d)	$(comp_{dir}, c_e, dir)$

3. Experiment #1 (DBM-2): Learning from Fragmented Data

In our experience (Spitkovsky et al., 2011a), punctuation can be viewed as implicit partial bracketing constraints (Pereira and Schabes, 1992): assuming that some (head) word from each inter-punctuation fragment derives the entire fragment is a useful approximation in the unsupervised setting. With this restriction, splitting text at punctuation is equivalent to learning partial parse forests — partial because longer fragments are left unparsed, and forests because even the parsed fragments are left unconnected (Moore et al., 1995). We allow grammar inducers to focus on modeling lower-level substructures first,[3] before forcing them to learn how these pieces may fit together. Deferring decisions associated with potentially long-distance inter-fragment relations and dependency arcs from longer fragments to a later training stage is thus a variation on the "easy-first" strategy (Goldberg and Elhadad, 2010), which is a fast and powerful heuristic from the supervised dependency parsing setting.

We bootstrapped DBM-2 using snippets of text obtained by slicing up all input sentences at punctuation. Splitting data increased the number of training tokens from 163,715 to 709,215 (and effective short training inputs from 15,922 to 34,856). Ordinarily, tree generation would be conditioned on an exogenous sentence-completeness status (*comp*), using presence of sentence-final punctuation as a binary proxy. We refined this notion, accounting for new kinds of fragments: (i) for the purposes of modeling roots, only unsplit sentences could remain complete; as for stopping decisions, (ii) leftmost fragments (prefixes of complete original sentences) are left-complete; and, analogously, (iii) rightmost fragments (suffixes) retain their status vis-à-vis right stopping decisions (see Figure 1). With this set-up, performance improved from 59.7 to 60.2% (from 3.4 to 3.5% for exact trees — see Table 1).

Next, we will show how to make better use of the additional fragmented training data.

3. About which our *loose* and *sprawl* punctuation-induced constraints agree (Spitkovsky et al., 2011a, §2.2).

4. Experiment #2 (DBM-i): Learning with a Coarse Model

In modeling head word fertilities, DBMs distinguish between the adjacent case (adj = T, deciding whether or not to have any children in a given direction, $dir \in \{\texttt{L},\texttt{R}\}$) and non-adjacent cases (adj = F, whether to cease spawning additional daughters — see $\mathbb{P}_{\text{STOP}}$ in Table 2). This level of detail can be wasteful for short fragments, however, since non-adjacency will be exceedingly rare there: most words will not have many children. Therefore, we can reduce the model by eliding adjacency. On the down side, this leads to some loss of expressive power; but on the up side, pooled information about phrase edges could flow more easily inwards from input boundaries, since it will not be quite so needlessly subcategorized.

We implemented DBM-i by conditioning all stopping decisions only on the direction in which a head word is growing, the input's completeness status in that direction and the identity of the head's farthest descendant on that side (the head word itself, in the adjacent case — see Table 2 and Appendix B). With this smaller initial model, directed dependency accuracy on the test set improved only slightly, from 60.2 to 60.5%; however, performance at the granularities of whole trees increased dramatically, from 3.5 to 4.9% (see Table 1).

5. Experiment #3 (DBM-0): Learning with an Ablated Model

DBM-i maintains separate root distributions for complete and incomplete sentences (see $\mathbb{P}_{\text{ATTACH}}$ for $\diamond$ in Table 2), which can isolate verb and modal types heading typical sentences from the various noun types deriving captions, headlines, titles and other fragments that tend to be common in news-style data. Heads of inter-punctuation fragments are less homogeneous than actual sentence roots, however. Therefore, we can simplify the learning task by approximating what would be a high-entropy distribution with a uniform multinomial, which is equivalent to updating DBM-i via a "partial" EM variant (Neal and Hinton, 1999).

We implemented DBM-0 by modifying DBM-i to hardwire the root probabilities as one over the number of word classes (1/200, in our case), for all incomplete inputs. With this more compact, asymmetric model, directed dependency accuracy improved substantially, from 60.5 to 61.2% (though only slightly for exact trees, from 4.9 to 5.0% — see Table 1).

6. Conclusion

We presented an effective divide-and-conquer strategy for bootstrapping grammar inducers. Our procedure is simple and efficient, achieving state-of-the-art results on a standard English dependency grammar induction task by simultaneously scaffolding on both model and data complexity, using a greatly simplified dependency-and-boundary model with inter-punctuation fragments of sentences. Future work could explore inducing structure from sentence prefixes and suffixes — or even bootstrapping from intermediate n-grams, perhaps via novel parsing models that may be better equipped for handling distituent fragments.

Acknowledgments

We thank the anonymous reviewers and conference organizers for their help and suggestions.

Funded, in part, by Defense Advanced Research Projects Agency (DARPA) Machine Reading Program, under Air Force Research Laboratory (AFRL) prime contract no. FA8750-09-C-0181.

References

H. Alshawi. Head automata for speech translation. In *ICSLP*, 1996.

Y. Bengio, J. Louradour, R. Collobert, and J. Weston. Curriculum learning. In *ICML*, 2009.

M. R. Brent and J. M. Siskind. The role of exposure to isolated words in early vocabulary development. *Cognition*, 81, 2001.

P. F. Brown, V. J. Della Pietra, S. A. Della Pietra, and R. L. Mercer. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19, 1993.

A. Clark. Combining distributional and morphological information for part of speech induction. In *EACL*, 2003.

M. Collins. *Head-Driven Statistical Models for Natural Language Parsing*. PhD thesis, University of Pennsylvania, 1999.

J. L. Elman. Learning and development in neural networks: The importance of starting small. *Cognition*, 48, 1993.

R. Frank. From regular to context-free to mildly context-sensitive tree rewriting systems: The path of child language acquisition. In A. Abeillé and O. Rambow, editors, *Tree Adjoining Grammars: Formalisms, Linguistic Analysis and Processing*. CSLI Publications, 2000.

K. Gimpel and N. A. Smith. Concavity and initialization for unsupervised dependency grammar induction. Technical report, CMU, 2011.

Y. Goldberg and M. Elhadad. An efficient algorithm for easy-first non-directional dependency parsing. In *NAACL-HLT*, 2010.

Y. Goldberg, M. Adler, and M. Elhadad. EM can find pretty good HMM POS-taggers (when given a good start). In *HLT-ACL*, 2008.

K. A. Krueger and P. Dayan. Flexible shaping: How learning in small steps helps. *Cognition*, 110, 2009.

K. Lari and S. J. Young. The estimation of stochastic context-free grammars using the inside-outside algorithm. *Computer Speech and Language*, 4, 1990.

M. P. Marcus, B. Santorini, and M. A. Marcinkiewicz. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19, 1993.

R. Moore, D. Appelt, J. Dowding, J. M. Gawron, and D. Moran. Combining linguistic and statistical knowledge sources in natural-language processing for ATIS. In *SLST*, 1995.

R. M. Neal and G. E. Hinton. A view of the EM algorithm that justifies incremental, sparse, and other variants. In M. I. Jordan, editor, *Learning in Graphical Models*. MIT Press, 1999.

F. Pereira and Y. Schabes. Inside-outside reestimation from partially bracketed corpora. In *ACL*, 1992.

E. Ponvert, J. Baldridge, and K. Erk. Simple unsupervised identification of low-level constituents. In *ICSC*, 2010.

E. Ponvert, J. Baldridge, and K. Erk. Simple unsupervised grammar induction from raw text with cascaded finite state models. In *ACL-HLT*, 2011.

V. I. Spitkovsky, H. Alshawi, and D. Jurafsky. From Baby Steps to Leapfrog: How "Less is More" in unsupervised dependency parsing. In *NAACL-HLT*, 2010a.

V. I. Spitkovsky, H. Alshawi, D. Jurafsky, and C. D. Manning. Viterbi training improves unsupervised dependency parsing. In *CoNLL*, 2010b.

V. I. Spitkovsky, H. Alshawi, and D. Jurafsky. Punctuation: Making a point in unsupervised dependency parsing. In *CoNLL*, 2011a.

V. I. Spitkovsky, A. X. Chang, H. Alshawi, and D. Jurafsky. Unsupervised dependency parsing without gold part-of-speech tags. In *EMNLP*, 2011b.

V. I. Spitkovsky, H. Alshawi, and D. Jurafsky. Three dependency-and-boundary models for grammar induction. In *EMNLP-CoNLL*, 2012.

K. Tu and V. Honavar. On the utility of curricula in unsupervised learning of probabilistic grammars. In *IJCAI*, 2011.

Appendix A. The Dependency-and-Boundary Models (DBMs 1, 2 and 3)

All DBMs begin by choosing a class for the root word (c_r). Remainders of parse structures, if any, are produced recursively. Each node spawns off ever more distant left dependents by (i) deciding whether to have more children, conditioned on direction (left), the class of the (leftmost) fringe word in the partial parse (initially, itself), and other parameters (such as adjacency of the would-be child); then (ii) choosing its child's category, based on direction, the head's own class, etc. Right dependents are generated analogously, but using separate factors. Unlike traditional head-outward models, DBMs condition their generative process on more observable state: left and right end words of phrases being constructed. Since left and right child sequences are still generated independently, DBM grammars are split-head.

DBM-2 maintains two related grammars: one for complete sentences ($comp = \texttt{T}$), approximated by presence of final punctuation, and another for incomplete fragments. These grammars communicate through shared estimates of word attachment parameters, making it possible to learn from mixtures of input types without polluting root and stopping factors.

DBM-3 conditions attachments on additional context, distinguishing arcs that cross punctuation boundaries ($cross = \texttt{T}$) from lower-level dependencies. We allowed only heads of fragments to attach other fragments as part of (*loose*) constrained Viterbi EM; in inference, entire fragments could be attached by arbitrary external words (*sprawl*). All missing families of factors (e.g., those of punctuation-crossing arcs) were initialized as uniform multinomials.

Appendix B. Partial Dependency-and-Boundary Models (DBMs *i* and 0)

Since dependency structures are trees, few heads get to spawn multiple dependents on the same side. High fertilities are especially rare in short fragments, inviting economical models whose stopping parameters can be lumped together (because in adjacent cases heads and fringe words coincide: $adj = \texttt{T} \rightarrow h = e$, hence $c_h = c_e$). Eliminating inessential components, such as the likely-heterogeneous root factors of incomplete inputs, can also yield benefits.

Consider the sentence ⓐ ⓩ. It admits two structures: $\underline{\text{ⓐ}}\frown\text{ⓩ}$ and $\text{ⓐ}\frown\underline{\text{ⓩ}}$. In theory, neither should be preferred. In practice, if the first parse occurs $100p\%$ of the time, a multi-component model could re-estimate total probability as $p^n + (1-p)^n$, where n may exceed its number of independent components. Only root and adjacent stopping factors are non-deterministic here: $\mathbb{P}_{\text{ROOT}}(\text{ⓐ}) = \mathbb{P}_{\text{STOP}}(\text{ⓩ}, \texttt{L}) = p$ and $\mathbb{P}_{\text{ROOT}}(\text{ⓩ}) = \mathbb{P}_{\text{STOP}}(\text{ⓐ}, \texttt{R}) = 1-p$; attachments are fixed (ⓐ can only attach ⓩ and vice-versa). Tree probabilities are thus cubes ($n = 3$): a root and two stopping factors (one for each word, on different sides), $\mathbb{P}(\text{ⓐ ⓩ}) = \mathbb{P}(\underline{\text{ⓐ}}\frown\text{ⓩ}) + \mathbb{P}(\text{ⓐ}\frown\underline{\text{ⓩ}})$

$$= \overbrace{\mathbb{P}_{\text{ROOT}}(\text{ⓐ})}^{p}\underbrace{\mathbb{P}_{\text{STOP}}(\text{ⓐ}, \texttt{L})}_{1}\overbrace{(1 - \mathbb{P}_{\text{STOP}}(\text{ⓐ}, \texttt{R}))}^{p}\underbrace{\mathbb{P}_{\text{ATTACH}}(\text{ⓐ}, \texttt{R}, \text{ⓩ})}_{1}\overbrace{\mathbb{P}_{\text{STOP}}(\text{ⓩ}, \texttt{L})}^{p}\underbrace{\mathbb{P}_{\text{STOP}}(\text{ⓩ}, \texttt{R})}_{1}$$

$$+ \overbrace{\mathbb{P}_{\text{ROOT}}(\text{ⓩ})}^{1-p}\underbrace{\mathbb{P}_{\text{STOP}}(\text{ⓩ}, \texttt{R})}_{1}\overbrace{(1 - \mathbb{P}_{\text{STOP}}(\text{ⓩ}, \texttt{L}))}^{1-p}\underbrace{\mathbb{P}_{\text{ATTACH}}(\text{ⓩ}, \texttt{L}, \text{ⓐ})}_{1}\overbrace{\mathbb{P}_{\text{STOP}}(\text{ⓐ}, \texttt{R})}^{1-p}\underbrace{\mathbb{P}_{\text{STOP}}(\text{ⓐ}, \texttt{L})}_{1} = p^3 + (1-p)^3.$$

For $p \in [0, 1]$ and $n \in \mathbb{Z}^+$, $p^n + (1-p)^n \leq 1$, with strict inequality if $p \notin \{0, 1\}$ and $n > 1$. Clearly, as n grows above one, optimizers will more strongly prefer extreme solutions $p \in \{0, 1\}$, despite lacking evidence in the data. Since the exponent n is related to numbers of input words and independent modeling components, a recipe of short inputs — combined with simpler, partial models — could help alleviate some of this pressure towards arbitrary determinism.