Punctuation: Making a Point in Unsupervised Dependency Parsing

Valentin I. Spitkovsky
Computer Science Department
Stanford University and Google Inc.
valentin@cs.stanford.edu

Hiyan Alshawi
Google Inc.
Mountain View, CA, 94043, USA
hiyan@google.com

Daniel Jurafsky
Departments of Linguistics and Computer Science
Stanford University, Stanford, CA, 94305, USA
jurafsky@stanford.edu

Abstract

We show how punctuation can be used to improve unsupervised dependency parsing. Our linguistic analysis confirms the strong connection between English punctuation and phrase boundaries in the Penn Treebank. However, approaches that naively include punctuation marks in the grammar (as if they were words) do not perform well with Klein and Manning's Dependency Model with Valence (DMV). Instead, we split a sentence at punctuation and impose parsing restrictions over its fragments. Our grammar inducer is trained on the Wall Street Journal (WSJ) and achieves 59.5% accuracy out-of-domain (Brown sentences with 100 or fewer words), more than 6% higher than the previous best results. Further evaluation, using the 2006/7 CoNLL sets, reveals that punctuation aids grammar induction in 17 of 18 languages, for an overall average net gain of 1.3%. Some of this improvement is from training, but more than half is from parsing with induced constraints, in inference. Punctuation-aware decoding works with existing (even already-trained) parsing models and always increased accuracy in our experiments.

1 Introduction

Unsupervised dependency parsing is a type of grammar induction — a central problem in computational linguistics. It aims to uncover hidden relations between head words and their dependents in free-form text. Despite decades of significant research efforts, the task still poses a challenge, as sentence structure is underdetermined by only raw, unannotated words.

Structure can be clearer in *formatted* text, which typically includes proper capitalization and punctuation (Gravano et al., 2009). Raw word streams, such as utterances transcribed by speech recognizers, are often difficult even for humans (Kim and Woodland, 2002). Therefore, one would expect grammar inducers to exploit any available linguistic meta-data. And yet in unsupervised dependency parsing, sentence-internal punctuation has long been ignored (Carroll and Charniak, 1992; Paskin, 2001; Klein and Manning, 2004; Blunsom and Cohn, 2010, *inter alia*).

HTML is another kind of meta-data that is ordinarily stripped out in pre-processing. However, recently Spitkovsky et al. (2010b) demonstrated that web markup can successfully guide hierarchical syntactic structure discovery, observing, for example, that anchors often match linguistic constituents:

..., whereas McCain is secure on the topic, Obama <a>[$_{\textbf{VP}}$ worries about winning the pro-Israel vote]</a>.

We propose exploring punctuation's potential to aid grammar induction. Consider a motivating example (all of our examples are from WSJ), in which all (six) marks align with constituent boundaries:

[$_{\textbf{SBAR}}$ Although it probably has reduced the level of expenditures for some purchasers], [$_{\textbf{NP}}$ utilization management] — [$_{\textbf{PP}}$ like most other cost containment strategies] — [$_{\textbf{VP}}$ doesn't appear to have altered the long-term rate of increase in health-care costs], [$_{\textbf{NP}}$ the Institute of Medicine], [$_{\textbf{NP}}$ an affiliate of the National Academy of Sciences], [$_{\textbf{VP}}$ concluded after a two-year study].

This link between punctuation and constituent boundaries suggests that we could approximate parsing by treating inter-punctuation fragments independently. In training, our algorithm first parses each fragment separately, then parses the sequence of the resulting head words. In inference, we use a better approximation that allows heads of fragments to be attached by arbitrary external words, e.g.:

The Soviets complicated the issue by offering to [$_{\textbf{VP}}$ include light tanks], [$_{\textbf{SBAR}}$ which are as light as ...].

	Count	*POS Sequence*	*Frac*	*Cum*
1	3,492	NNP	2.8%	
2	2,716	CD CD	2.2	5.0
3	2,519	NNP NNP	2.0	7.1
4	2,512	RB	2.0	9.1
5	1,495	CD	1.2	10.3
6	1,025	NN	0.8	11.1
7	1,023	NNP NNP NNP	0.8	11.9
8	916	IN NN	0.7	12.7
9	795	VBZ NNP NNP	0.6	13.3
10	748	CC	0.6	13.9
11	730	CD DT NN	0.6	14.5
12	705	PRP VBD	0.6	15.1
13	652	JJ NN	0.5	15.6
14	648	DT NN	0.5	16.1
15	627	IN DT NN	0.5	16.6
WSJ	+103,148	*more with Count* $\leq$ 621	83.4%	

Table 1: Top 15 fragments of POS tag sequences in WSJ.

	Count	*Non-Terminal*	*Frac*	*Cum*
1	40,223	S	32.5%	
2	33,607	NP	27.2	59.7
3	16,413	VP	13.3	72.9
4	12,441	PP	10.1	83.0
5	8,350	SBAR	6.7	89.7
6	4,085	ADVP	3.3	93.0
7	3,080	QP	2.5	95.5
8	2,480	SINV	2.0	97.5
9	1,257	ADJP	1.0	98.5
10	369	PRN	0.3	98.8
WSJ	+1,446	*more with Count* $\leq$ 356	1.2%	

Table 2: Top 99% of the lowest dominating non-terminals deriving complete inter-punctuation fragments in WSJ.

2 Definitions, Analyses and Constraints

Punctuation and syntax are related (Nunberg, 1990; Briscoe, 1994; Jones, 1994; Doran, 1998, *inter alia*). But are there simple enough connections between the two to aid in grammar induction? This section explores the regularities. Our study of punctuation in WSJ (Marcus et al., 1993) parallels Spitkovsky et al.'s (2010b, §5) analysis of markup from a weblog, since their proposed constraints turn out to be useful. Throughout, we define an inter-punctuation ***fragment*** as a maximal (non-empty) consecutive sequence of words that does not cross punctuation boundaries and is shorter than its source sentence.

2.1 A Linguistic Analysis

Out of 51,558 sentences, most — 37,076 (71.9%) — contain sentence-internal punctuation. These punctuated sentences contain 123,751 fragments, nearly all — 111,774 (90.3%) — of them multi-token.

Common part-of-speech (POS) sequences comprising fragments are diverse (note also their flat distribution — see Table 1). The plurality of fragments are dominated by a clause, but most are dominated by one of several kinds of phrases (see Table 2). As expected, punctuation does not occur at all constituent boundaries: Of the top 15 productions that yield fragments, five do *not* match the exact bracketing of their lowest dominating non-terminal (see ranks 6, 11, 12, 14 and 15 in Table 3, left). Four of them miss a left-adjacent clause, e.g., S $\rightarrow$ S <u>NP VP</u>:

[$_{\text{S}}$ [$_{\text{S}}$ It's an overwhelming job], [$_{\text{NP}}$ she] [$_{\text{VP}}$ says.]]

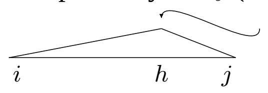
This production is flagged because the fragment <u>NP VP</u> is not *a* constituent — it is two; still, **49.4**% of all fragments do align with whole constituents.

Inter-punctuation fragments correspond more strongly to dependencies (see Table 3, right). Only one production (rank 14) shows a daughter outside her mother's fragment. Some number of such productions is inevitable and expected, since fragments must coalesce (i.e., the root of at least one fragment — in every sentence with sentence-internal punctuation — must be attached by some word from a different, external fragment). We find it noteworthy that in 14 of the 15 most common cases, a word in an inter-punctuation fragment derives precisely the rest of that fragment, attaching none of the other, external words. This is true for **39.2**% of all fragments, and if we include fragments whose heads attach other fragments' heads, agreement increases to **74.0**% (see *strict* and *loose* constraints in §2.2, next).

2.2 Five Parsing Constraints

Spitkovsky et al. (2010b, §5.3) showed how to express similar correspondences with markup as parsing constraints. They proposed four constraints but employed only the strictest three, omitting implementation details. We revisit their constraints, specifying precise logical formulations that we use in our code, and introduce a fifth (most relaxed) constraint.

Let $[x, y]$ be a fragment (or markup) spanning positions x through y (inclusive, with $1 \leq x < y \leq l$), in a sentence of length l. And let $[i, j]_h$ be a sealed span headed by h ($1 \leq i \leq h \leq j \leq l$), i.e., the word at position h dominates precisely $i \dots j$ (but none other):



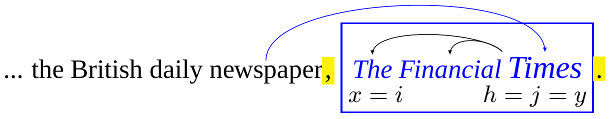
	Count	*Constituent Production*	*Frac*	*Cum*
1	7,115	PP → IN NP	5.7%	
2	5,950	S → NP VP	4.8	10.6
3	3,450	NP → NP PP	2.8	13.3
4	2,799	SBAR → WHNP S	2.3	15.6
5	2,695	NP → NNP	2.2	17.8
6	2,615	S → S NP VP	2.1	19.9
7	2,480	SBAR → IN S	2.0	21.9
8	2,392	NP → NNP NNP	1.9	23.8
9	2,354	ADVP → RB	1.9	25.7
10	2,334	QP → CD CD	1.9	27.6
11	2,213	S → PP NP VP	1.8	29.4
12	1,441	S → S CC S	1.2	30.6
13	1,317	NP → NP NP	1.1	31.6
14	1,314	S → SBAR NP VP	1.1	32.7
15	1,172	SINV → S VP NP NP	0.9	33.6
WSJ	+82,110	*more with Count* ≤ 976	66.4%	

	Count	*Head-Outward Spawn*	*Frac*	*Cum*
1	11,928	IN	9.6%	
2	8,852	NN	7.2	16.8
3	7,802	NNP	6.3	23.1
4	4,750	CD	3.8	26.9
5	3,914	VBD	3.2	30.1
6	3,672	VBZ	3.0	33.1
7	3,436	RB	2.8	35.8
8	2,691	VBG	2.2	38.0
9	2,304	VBP	1.9	39.9
10	2,251	NNS	1.8	41.7
11	1,955	WDT	1.6	43.3
12	1,409	MD	1.1	44.4
13	1,377	VBN	1.1	45.5
14	1,204	IN VBD	1.0	46.5
15	927	JJ	0.7	47.3
WSJ	+65,279	*more with Count* ≤ 846	52.8%	

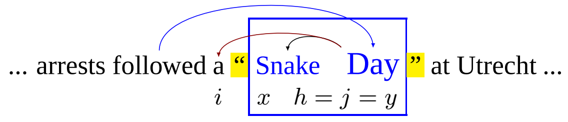
Table 3: Top 15 productions yielding punctuation-induced fragments in WSJ, viewed as constituents (left) and as dependencies (right). For constituents, we recursively expanded any internal nodes that did not align with the associated fragmentation (underlined). For dependencies we dropped all daughters that fell entirely in the same region as their mother (i.e., both inside a fragment, both to its left or both to its right), keeping only crossing attachments (just one).

Define $inside(h, x, y)$ as true iff $x \leq h \leq y$; and let $cross(i, j, x, y)$ be true iff $(i < x \wedge j \geq x \wedge j < y) \vee (i > x \wedge i \leq y \wedge j > y)$. The three tightest constraints impose conditions which, when satisfied, disallow sealing $[i, j]_h$ in the presence of an annotation $[x, y]$:

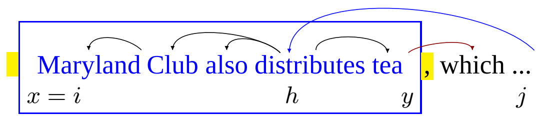
strict — requires $[x, y]$ itself to be sealed in the parse tree, voiding all seals that straddle exactly one of $\{x, y\}$ or protrude beyond $[x, y]$ if their head is inside. This constraint holds for **39.2**% of fragments. By contrast, only 35.6% of HTML annotations, such as anchor texts and italics, agree with it (Spitkovsky et al., 2010b). This necessarily fails in every sentence with internal punctuation (since there, *some* fragment must take charge and attach another), when $cross(i, j, x, y) \vee (inside(h, x, y) \wedge (i < x \vee j > y))$.



loose — if $h \in [x, y]$, requires that everything in $x \dots y$ fall under h, with only h allowed external attachments. This holds for **74.0**% of fragments — 87.5% of markup, failing when $cross(i, j, x, y)$.



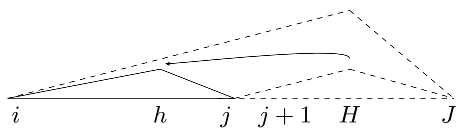
sprawl — still requires that h derive $x \dots y$ but lifts restrictions on external attachments. Holding for **92.9**% of fragments (95.1% of markup), it fails when $cross(i, j, x, y) \wedge \neg inside(h, x, y)$.



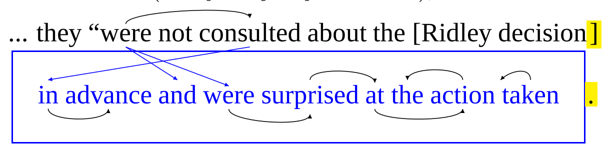
These three strictest constraints lend themselves to a straight-forward implementation as an $O(l^5)$ chart-based decoder. Ordinarily, the probability of $[i, j]_h$ is computed by multiplying the probability of the associated *un*sealed span by two stopping probabilities — that of the word at h on the left (adjacent if $i = h$; non-adjacent if $i < h$) and on the right (adjacent if $h = j$; non-adjacent if $h < j$). To impose a constraint, we ran through all of the annotations $[x, y]$ associated with a sentence and zeroed out this probability if any of them satisfied disallowed conditions.

There are faster — e.g., $O(l^4)$, and even $O(l^3)$ — recognizers for split head automaton grammars (Eisner and Satta, 1999). Perhaps a more practical, but still clear, approach would be to generate n-best lists using a more efficient unconstrained algorithm, then apply the constraints as a post-filtering step.

Relaxed constraints disallow joining adjacent subtrees, e.g., preventing the seal $[i, j]_h$ from merging below the *un*sealed span $[j + 1, J]_H$, on the left:

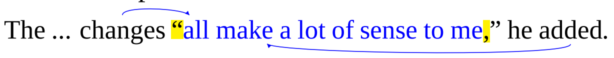


tear — prevents $x \ldots y$ from being torn apart by external heads from *opposite* sides. It holds for **94.7**% of fragments (97.9% of markup), and is violated when $(x \le j \,\wedge\, y > j \,\wedge\, h < x)$, in this case.



... they “were not consulted about the [Ridley decision] in advance and were surprised at the action taken .

thread — requires only that no path from the root to a leaf enter $[x, y]$ twice. This holds for **95.0**% of all fragments (98.5% of markup); it is violated when $(x \le j \,\wedge\, y > j \,\wedge\, h < x) \,\wedge\, (H \le y)$, again, in this case. Example that satisfies *thread* but violates *tear*:

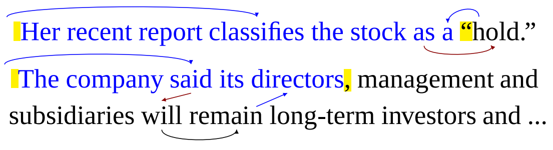


The ... changes “all make a lot of sense to me,” he added.

The case when $[i, j]_h$ is to the right is entirely symmetric, and these constraints could be incorporated in a more sophisticated decoder (since i and J do not appear in the formulae, above). We implemented them by zeroing out the probability of the word at H attaching that at h (to its left), in case of a violation.

Note that all five constraints are nested. In particular, this means that it does not make sense to combine them, for a given annotation $[x, y]$, since the result would just match the strictest one. Our markup number for *tear* is lower (97.9 versus 98.9%) than Spitkovsky et al.’s (2010b), because theirs allowed cases where markup was *neither* torn nor threaded.

Common structures that violate *thread* (and, consequently, all five of the constraints) include, e.g., “seamless” quotations and even ordinary lists:



Her recent report classifies the stock as a “hold.”

The company said its directors, management and subsidiaries will remain long-term investors and ...

2.3 Comparison with Markup

Most punctuation-induced constraints are less accurate than the corresponding markup-induced constraints (e.g., *sprawl*: 92.9 vs. 95.1%; *loose*: 74.0 vs. 87.5%; but not *strict*: 39.2 vs. 35.6%). However, markup is rare: Spitkovsky et al. (2010b, §5.1) observed that only 10% of the sentences in their blog were annotated; in contrast, over 70% of the sentences in WSJ are fragmented by punctuation.

Fragments are more than 40% likely to be dominated by a clause; for markup, this number is below 10% — nearly 75% of it covered by noun phrases. Further, inter-punctuation fragments are spread more evenly under noun, verb, prepositional, adverbial and adjectival phrases (approximately 27:13:10:3:1 versus 75:13:2:1:1) than markup.[1]

3 The Model, Methods and Metrics

We model grammar via Klein and Manning’s (2004) Dependency Model with Valence (DMV), which ordinarily strips out punctuation. Since this step already requires identification of marks, our techniques are just as “unsupervised.” We would have preferred to test punctuation in their original set-up, but this approach wasn’t optimal, for several reasons. First, Klein and Manning (2004) trained with short sentences (up to only ten words, on WSJ10), whereas most punctuation appears in longer sentences. And second, although we could augment the training data (say, to WSJ45), Spitkovsky et al. (2010a) showed that classic EM struggles with longer sentences. For this reason, we use Viterbi EM and the scaffolding suggested by Spitkovsky et al. (2010a) — also the setting in which Spitkovsky et al. (2010b) tested their markup-induced constraints.

3.1 A Basic System

Our system is based on Laplace-smoothed Viterbi EM, following Spitkovsky et al.’s (2010a) two-stage scaffolding: the first stage trains with just the sentences up to length 15; the second stage then retrains on nearly all sentences — those with up to 45 words.

Initialization

Klein and Manning’s (2004) “ad-hoc harmonic” initializer does not work very well for longer sentences, particularly with Viterbi training (Spitkovsky et al., 2010a, Figure 3). Instead, we use an improved initializer that approximates the attachment probability between two words as an average, over all sentences, of their normalized aggregate *weighted* distances. Our weighting function is $w(d) = 1 + 1/\lg(1 + d)$.[2]

Termination

Spitkovsky et al. (2010a) iterated until successive changes in overall (best parse) per-token cross-entropy dropped below 2^{-20} bits. Since smoothing can (and does, at times) increase the objective, we found it more efficient to terminate early, after ten

[1]Markup and fragments are as likely to be in verb phrases.

[2]Integer $d \ge 1$ is a distance between two tokens; lg is $\log_2$.

steps of suboptimal models. We used the lowest-perplexity (not necessarily the last) model found, as measured by the cross-entropy of the training data.

Constrained Training

Training with punctuation replaces ordinary Viterbi parse trees, at every iteration of EM, with the output of a constrained decoder. In all experiments other than #2 (§5) we train with the *loose* constraint. Spitkovsky et al. (2010b) found this setting to be best for markup-induced constraints. We apply it to constraints induced by inter-punctuation fragments.

Constrained Inference

Spitkovsky et al. (2010b) recommended using the *sprawl* constraint in inference. Once again, we follow their advice in all experiments except #2 (§5).

3.2 Data Sets and Scoring

We trained on the Penn English Treebank's Wall Street Journal portion (Marcus et al., 1993). To evaluate, we automatically converted its labeled constituents into unlabeled dependencies, using deterministic "head-percolation" rules (Collins, 1999), discarding punctuation, any empty nodes, etc., as is standard practice (Paskin, 2001; Klein and Manning, 2004). We also evaluated against the parsed portion of the Brown corpus (Francis and Kučera, 1979), used as a blind, out-of-domain evaluation set,[3] similarly derived from labeled constituent parse trees.

We report directed accuracies — fractions of correctly guessed arcs, including the root, in unlabeled reference dependency parse trees, as is also standard practice (Paskin, 2001; Klein and Manning, 2004). One of our baseline systems (§3.3) produces dependency trees containing punctuation. In this case we do not score the heads assigned to punctuation and use *forgiving scoring* for regular words: crediting correct heads separated from their children by punctuation alone (from the point of view of the child, looking up to the nearest non-punctuation ancestor).

3.3 Baseline Systems

Our primary baseline is the basic system without constraints (*standard training*). It ignores punctuation, as is standard, scoring 52.0% against WSJ45.

A secondary (*punctuation as words*) baseline incorporates punctuation into the grammar as if it were words, as in *supervised* dependency parsing (Nivre et al., 2007b; Lin, 1998; Sleator and Temperley, 1993, *inter alia*). It is worse, scoring only 41.0%.[4,5]

[3]Note that WSJ$\{15, 45\}$ overlap with Section 23 — training on the test set is standard practice in unsupervised learning.

4 Experiment #1: Default Constraints

Our first experiment compares "punctuation as constraints" to the baseline systems. We use default settings, as recommended by Spitkovsky et al. (2010b): *loose* in training; and *sprawl* in inference. Evaluation is on Section 23 of WSJ (all sentence lengths). To facilitate comparison with prior work, we also report accuracies against shorter sentences, with up to ten non-punctuation tokens (WSJ10 — see Table 4).

We find that both constrained regimes improve performance. Constrained decoding alone increases the accuracy of a standardly-trained system from 52.0% to 54.0%. And constrained training yields 55.6% — 57.4% in combination with inference.

[4]We were careful to use exactly the same data sets in both cases, not counting punctuation towards sentence lengths. And we used forgiving scoring (§3.2) when evaluating these trees.

[5]To get this particular number we forced punctuation to be tacked on, as a layer below the tree of words, to fairly compare systems (using the same initializer). Since improved initialization strategies — both ours and Klein and Manning's (2004) "ad-hoc harmonic" initializer — rely on distances between tokens, they could be unfairly biased towards one approach or the other, if punctuation counted towards length. We also trained similar baselines without restrictions, allowing punctuation to appear anywhere in the tree (still with forgiving scoring — see §3.2), using the uninformed uniform initializer (Spitkovsky et al., 2010a). Disallowing punctuation as a parent of a real word made things worse, suggesting that not all marks belong near the leaves (sentence stops, semicolons, colons, etc. make more sense as roots and heads). We tried the weighted initializer also without restrictions and repeated all experiments without scaffolding, on WSJ15 and WSJ45 alone, but treating punctuation as words never came within even 5% of (comparable) standard training. Punctuation, as words, reliably disrupted learning.

	WSJ$^\infty$	WSJ10
Supervised DMV	69.8	83.6
w/Constrained Inference	73.0	84.3
Punctuation as Words	41.7	54.8
Standard Training	52.0	63.2
w/Constrained Inference	54.0	63.6
Constrained Training	55.6	67.0
w/Constrained Inference	**57.4**	**67.5**

Table 4: Directed accuracies on Section 23 of WSJ$^\infty$ and WSJ10 for the supervised DMV, our baseline systems and the punctuation runs (all using the weighted initializer).

These are multi-point increases, but they could disappear in a more accurate state-of-the-art system.

To test this hypothesis, we applied constrained decoding to a *supervised* system. We found that this (ideal) instantiation of the DMV benefits as much or more than the unsupervised systems: accuracy increases from 69.8% to 73.0%. Punctuation seems to capture the kinds of, perhaps long-distance, regularities that are not accessible to the model, possibly because of its unrealistic independence assumptions.

5 Experiment #2: Optimal Settings

Spitkovsky et al. (2010b) recommended training with *loose* and decoding with *sprawl* based on their experiments with markup. But are these the right settings for punctuation? Inter-punctuation fragments are quite different from markup — they are more prevalent but less accurate. Furthermore, we introduced a new constraint, *thread*, that Spitkovsky et al. (2010b) had not considered (along with *tear*).

We next re-examined the choices of constraints. Our full factorial analysis was similar, but significantly smaller, than Spitkovsky et al.'s (2010b): we excluded their larger-scale news and web data sets that are not publicly available. Nevertheless, we still tried every meaningful combination of settings, testing both *thread* and *tear* (instead of *strict*, since it can't work with sentences containing sentence-internal punctuation), in both training and inference. We did not find better settings than *loose* for training, and *sprawl* for decoding, among our options.

A full analysis is omitted due to space constraints. Our first observation is that constrained inference, using punctuation, is helpful and robust. It boosted accuracy (on WSJ45) by approximately 1.5%, on average, with all settings. Indeed, *sprawl* was consistently (but only slightly, at 1.6%, on average) better than the rest. Second, constrained training hurt more often than it helped. It degraded accuracy in all but one case, *loose*, where it gained approximately 0.4%, on average. Both improvements are statistically significant: $p \approx 0.036$ for training with *loose*; and $p \approx 5.6 \times 10^{-12}$ for decoding with *sprawl*.

6 More Advanced Methods

So far, punctuation has improved grammar induction in a toy setting. But would it help a modern system? Our next two experiments employ a slightly more complicated set-up, compared with the one used up until now (§3.1). The key difference is that this system is lexicalized, as is standard among the more accurate grammar inducers (Blunsom and Cohn, 2010; Gillenwater et al., 2010; Headden et al., 2009).

Lexicalization

We lexicalize only in the second (full data) stage, using the method of Headden et al. (2009). For words seen at least 100 times in the training corpus, we augment their gold POS tag with the lexical item. The first (data poor) stage remains entirely unlexicalized, with gold POS tags for word classes, as in the earlier systems (Klein and Manning, 2004).

Smoothing

We do not use smoothing in the second stage except at the end, for the final lexicalized model. Stage one still applies "add-one" smoothing at every iteration.

7 Experiment #3: State-of-the-Art

The purpose of these experiments is to compare the punctuation-enhanced DMV with other, recent state-of-the-art systems. We find that, lexicalized (§6), our approach performs better, by a wide margin; without lexicalization (§3.1), it was already better for longer, but not for shorter, sentences (see Tables 5 and 4).

We trained a variant of our system *without* gold part-of-speech tags, using the unsupervised word clusters (Clark, 2000) computed by Finkel and Manning (2009).[6] Accuracy decreased slightly, to 58.2% on Section 23 of WSJ (down only 0.2%). This result improves over substantial performance degradations previously observed for unsupervised dependency parsing with induced word categories (Klein and Manning, 2004; Headden et al., 2008, *inter alia*).

[6]Available from `http://nlp.stanford.edu/software/stanford-postagger-2008-09-28.tar.gz`: `models/egw.bnc.200`

	Brown	WSJ$^\infty$	WSJ10
(Headden et al., 2009)	—	—	68.8
(Spitkovsky et al., 2010b)	53.3	50.4	69.3
(Gillenwater et al., 2010)	—	53.3	64.3
(Blunsom and Cohn, 2010)	—	55.7	67.7
Constrained Training	58.4	58.0	69.3
w/Constrained Inference	**59.5**	**58.4**	**69.5**

Table 5: Accuracies on the out-of-domain Brown100 set and Section 23 of WSJ$^\infty$ and WSJ10, for the lexicalized punctuation run and other recent state-of-the-art systems.

		Unlexicalized, Unpunctuated				*Lexicalized*		*... **and Punctuated***		
CoNLL Year		*Initialization @15*		*Training @15*		*Retraining @45*		*Retraining @45*		*Net*
& Language		*1.*	*w/Inference*	*2.*	*w/Inference*	*3.*	*w/Inference*	*3′.*	*w/Inference*	*Gain*
Arabic	2006	23.3	23.6 (+0.3)	32.8	33.1 (+0.4)	31.5	31.6 (+0.1)	32.1	32.6 (+0.5)	+1.1
	’7	25.6	26.4 (+0.8)	33.7	34.2 (+0.5)	32.7	33.6 (+0.9)	34.9	35.3 (+0.4)	+2.6
Basque	’7	19.3	20.8 (+1.5)	29.9	30.9 (+1.0)	29.3	30.1 (+0.8)	29.3	29.9 (+0.6)	+0.6
Bulgarian	’6	23.7	24.7 (+1.0)	39.3	40.7 (+1.4)	38.8	39.9 (+1.1)	39.9	40.5 (+0.6)	+1.6
Catalan	’7	33.2	34.1 (+0.8)	54.8	55.5 (+0.7)	54.3	55.1 (+0.8)	54.3	55.2 (+0.9)	+0.9
Czech	’6	18.6	19.6 (+1.0)	34.6	35.8 (+1.2)	34.8	35.7 (+0.9)	37.0	37.8 (+0.8)	+3.0
	’7	17.6	18.4 (+0.8)	33.5	35.4 (+1.9)	33.4	34.4 (+1.0)	35.2	36.2 (+1.0)	+2.7
Danish	’6	22.9	24.0 (+1.1)	35.6	36.7 (+1.2)	36.9	37.8 (+0.9)	36.5	37.1 (+0.6)	+0.2
Dutch	’6	15.8	16.5 (+0.7)	*11.2*	12.5 (+1.3)	11.0	11.9 (+1.0)	13.7	14.0 (+0.3)	+3.0
English	’7	25.0	25.4 (+0.5)	47.2	49.5 (+2.3)	47.5	48.8 (+1.3)	49.3	50.3 (+0.9)	+2.8
German	’6	19.2	19.6 (+0.4)	27.4	28.0 (+0.7)	27.0	27.8 (+0.8)	28.2	28.6 (+0.4)	+1.6
Greek	’7	18.5	18.8 (+0.3)	20.7	21.4 (+0.7)	20.5	21.0 (+0.5)	20.9	21.2 (+0.3)	+0.7
Hungarian	’7	17.4	17.7 (+0.3)	*6.7*	7.2 (+0.5)	6.6	7.0 (+0.4)	7.8	8.0 (+0.2)	+1.4
Italian	’7	25.0	26.3 (+1.2)	29.6	29.9 (+0.3)	29.7	29.7 (+0.1)	28.3	28.8 (+0.5)	*-0.8*
Japanese	’6	30.0	30.0 (+0.0)	*27.3*	27.3 (+0.0)	27.4	27.4 (+0.0)	27.5	27.5 (+0.0)	+0.1
Portuguese	’6	27.3	27.5 (+0.2)	32.8	33.7 (+0.9)	32.7	33.4 (+0.7)	33.3	33.5 (+0.3)	+0.8
Slovenian	’6	21.8	21.9 (+0.2)	28.3	30.4 (+2.1)	28.4	30.4 (+2.0)	29.8	31.2 (+1.4)	+2.8
Spanish	’6	25.3	26.2 (+0.9)	31.7	32.4 (+0.7)	31.6	32.3 (+0.8)	31.9	32.3 (+0.5)	+0.8
Swedish	’6	31.0	31.5 (+0.6)	44.1	45.2 (+1.1)	45.6	46.1 (+0.5)	46.1	46.4 (+0.3)	+0.8
Turkish	’6	22.3	22.9 (+0.6)	39.1	39.5 (+0.4)	39.9	39.9 (+0.1)	40.6	40.9 (+0.3)	+1.0
	’7	22.7	23.3 (+0.6)	41.7	42.3 (+0.6)	41.9	42.1 (+0.2)	41.6	42.0 (+0.4)	+0.1
Average:		23.4	24.0 (**+0.7**)	31.9	32.9 (**+1.0**)	31.9	32.6 (**+0.7**)	32.6	33.2 (**+0.5**)	**+1.3**

Table 6: Multi-lingual evaluation for CoNLL sets, measured at all three stages of training, with and without constraints.

8 Experiment #4: Multi-Lingual Testing

This final batch of experiments probes the generalization of our approach (§6) across languages. The data are from 2006/7 CoNLL shared tasks (Buchholz and Marsi, 2006; Nivre et al., 2007a), where punctuation was identified by the organizers, who also furnished disjoint train/test splits. We tested against all sentences in their evaluation sets.[7,8]

The gains are *not* English-specific (see Table 6). Every language improves with constrained decoding (more so without constrained training); and all but Italian benefit in combination. Averaged across all eighteen languages, the net change in accuracy is 1.3%. After standard training, constrained decoding alone delivers a 0.7% gain, on average, never causing harm in any of our experiments. These gains are statistically significant: $p \approx 1.59 \times 10^{-5}$ for constrained training; and $p \approx 4.27 \times 10^{-7}$ for inference.

[7]With the exception of Arabic ’07, from which we discarded one sentence with 145 tokens. We down-weighed languages appearing in both years by 50% in our analyses, and excluded Chinese entirely, since it had already been cut up at punctuation.

[8]Note that punctuation was treated differently in the two years: in ’06, it was always at the leaves of the dependency trees; in ’07, it matched original annotations of the source treebanks. For both, we used punctuation-insensitive scoring (§3.2).

We did not detect synergy between the two improvements. However, note that without constrained training, “full” data sets do not help, on average, despite having more data and lexicalization. Furthermore, *after* constrained training, we detected no evidence of benefits to additional retraining: not with the relaxed *sprawl* constraint, nor unconstrained.

9 Related Work

Punctuation has been used to improve parsing since rule-based systems (Jones, 1994). Statistical parsers reap dramatic gains from punctuation (Engel et al., 2002; Roark, 2001; Charniak, 2000; Johnson, 1998; Collins, 1997, *inter alia*). And it is even known to help in *unsupervised* constituent parsing (Seginer, 2007). But for *dependency* grammar induction, until now, punctuation remained unexploited.

Parsing Techniques Most-Similar to Constraints A “divide-and-rule” strategy that relies on punctuation has been used in supervised constituent parsing of long Chinese sentences (Li et al., 2005). For English, there has been interest in *balanced* punctuation (Briscoe, 1994), more recently using rule-based filters (White and Rajkumar, 2008) in a combinatory categorial grammar (CCG). Our focus is specifically on *unsupervised* learning of *dependency* grammars and is similar, in spirit, to Eisner and Smith's (2005) "vine grammar" formalism. An important difference is that instead of imposing static limits on allowed dependency lengths, our restrictions are dynamic — they disallow some long (and some short) arcs that would have otherwise crossed nearby punctuation.

Incorporating partial bracketings into grammar induction is an idea tracing back to Pereira and Schabes (1992). It inspired Spitkovsky et al. (2010b) to mine parsing constraints from the web. In that same vein, we prospected a more abundant and natural language-resource — punctuation, using constraint-based techniques they developed for web markup.

Modern Unsupervised Dependency Parsing

State-of-the-art in unsupervised dependency parsing (Blunsom and Cohn, 2010) uses tree substitution grammars. These are powerful models, capable of learning large dependency fragments. To help prevent overfitting, a non-parametric Bayesian prior, defined by a hierarchical Pitman-Yor process (Pitman and Yor, 1997), is trusted to nudge training towards fewer and smaller grammatical productions.

We pursued a complementary strategy: using Klein and Manning's (2004) much simpler Dependency Model with Valence (DMV), but persistently steering training away from certain constructions, as guided by punctuation, to help prevent *underfitting*.

Various Other Uses of Punctuation in NLP

Punctuation is hard to predict,[9] partly because it can signal long-range dependences (Lu and Ng, 2010). It often provides valuable cues to NLP tasks such as part-of-speech tagging and named-entity recognition (Hillard et al., 2006), information extraction (Favre et al., 2008) and machine translation (Lee et al., 2006; Matusov et al., 2006). Other applications have included Japanese sentence analysis (Ohyama et al., 1986), genre detection (Stamatatos et al., 2000), bilingual sentence alignment (Yeh, 2003), semantic role labeling (Pradhan et al., 2005), Chinese creation-title recognition (Chen and Chen, 2005) and word segmentation (Li and Sun, 2009), plus, recently, automatic vandalism detection in Wikipedia (Wang and McKeown, 2010).

[9]Punctuation has high semantic entropy (Melamed, 1997); for an analysis of the many roles played in the WSJ by the comma — the most frequent and unpredictable punctuation mark in that data set — see Beeferman et al. (1998, Table 2).

10 Conclusions and Future Work

Punctuation improves dependency grammar induction. Many unsupervised (and supervised) parsers could be easily modified to use *sprawl*-constrained decoding in inference. It applies to pre-trained models and, so far, helped every data set and language.

Tightly interwoven into the fabric of writing systems, punctuation frames most unannotated plain-text. We showed that rules for converting markup into accurate parsing constraints are still optimal for inter-punctuation fragments. Punctuation marks are more ubiquitous and natural than web markup: what little punctuation-induced constraints lack in precision, they more than make up in recall — perhaps both types of constraints would work better yet in tandem. For language acquisition, a natural question is whether prosody could similarly aid grammar induction from speech (Kahn et al., 2005).

Our results underscore the power of simple models and algorithms, combined with common-sense constraints. They reinforce insights from *joint* modeling in *supervised* learning, where simplified, independent models, Viterbi decoding and expressive constraints excel at sequence labeling tasks (Roth and Yih, 2005). Such evidence is particularly welcome in *unsupervised* settings (Punyakanok et al., 2005), where it is crucial that systems scale gracefully to volumes of data, on top of the usual desiderata — ease of implementation, extension, understanding and debugging. Future work could explore softening constraints (Hayes and Mouradian, 1980; Chang et al., 2007), perhaps using features (Eisner and Smith, 2005; Berg-Kirkpatrick et al., 2010) or by learning to associate different settings with various marks: Simply adding a hidden tag for "ordinary" versus "divide" types of punctuation (Li et al., 2005) may already usefully extend our model.

Acknowledgments

Partially funded by the Air Force Research Laboratory (AFRL), under prime contract no. FA8750-09-C-0181, and by NSF, via award #IIS-0811974. We thank Omri Abend, Slav Petrov and anonymous reviewers for many helpful suggestions, and we are especially grateful to Jenny R. Finkel for shaming us into using punctuation, to Christopher D. Manning for reminding us to explore "punctuation as words" baselines, and to Noah A. Smith for encouraging us to test against languages other than English.

References

D. Beeferman, A. Berger, and J. Lafferty. 1998. CYBERPUNC: A lightweight punctuation annotation system for speech. In *ICASSP*.

T. Berg-Kirkpatrick, A. Bouchard-Côté, J. DeNero, and D. Klein. 2010. Painless unsupervised learning with features. In *NAACL-HLT*.

P. Blunsom and T. Cohn. 2010. Unsupervised induction of tree substitution grammars for dependency parsing. In *EMNLP*.

E. J. Briscoe. 1994. Parsing (with) punctuation, etc. Technical report, Xerox European Research Laboratory.

S. Buchholz and E. Marsi. 2006. CoNLL-X shared task on multilingual dependency parsing. In *CoNLL*.

G. Carroll and E. Charniak. 1992. Two experiments on learning probabilistic dependency grammars from corpora. Technical report, Brown University.

M.-W. Chang, L. Ratinov, and D. Roth. 2007. Guiding semi-supervision with constraint-driven learning. In *ACL*.

E. Charniak. 2000. A maximum-entropy-inspired parser. In *NAACL*.

C. Chen and H.-H. Chen. 2005. Integrating punctuation rules and naïve Bayesian model for Chinese creation title recognition. In *IJCNLP*.

A. Clark. 2000. Inducing syntactic categories by context distribution clustering. In *CoNLL-LLL*.

M. Collins. 1997. Three generative, lexicalised models for statistical parsing. In *ACL*.

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

C. D. Doran. 1998. *Incorporating Punctuation into the Sentence Grammar: A Lexicalized Tree Adjoining Grammar Perspective*. Ph.D. thesis, University of Pennsylvania.

J. Eisner and G. Satta. 1999. Efficient parsing for bilexical context-free grammars and head-automaton grammars. In *ACL*.

J. Eisner and N. A. Smith. 2005. Parsing with soft and hard constraints on dependency length. In *IWPT*.

D. Engel, E. Charniak, and M. Johnson. 2002. Parsing and disfluency placement. In *EMNLP*.

B. Favre, R. Grishman, D. Hillard, H. Ji, D. Hakkani-Tür, and M. Ostendorf. 2008. Punctuating speech for information extraction. In *ICASSP*.

J. R. Finkel and C. D. Manning. 2009. Joint parsing and named entity recognition. In *NAACL-HLT*.

W. N. Francis and H. Kučera, 1979. *Manual of Information to Accompany a Standard Corpus of Present-Day Edited American English, for use with Digital Computers*. Department of Linguistics, Brown University.

J. Gillenwater, K. Ganchev, J. Graça, F. Pereira, and B. Taskar. 2010. Posterior sparsity in unsupervised dependency parsing. Technical report, University of Pennsylvania.

A. Gravano, M. Jansche, and M. Bacchiani. 2009. Restoring punctuation and capitalization in transcribed speech. In *ICASSP*.

P. J. Hayes and G. V. Mouradian. 1980. Flexible parsing. In *ACL*.

W. P. Headden, III, D. McClosky, and E. Charniak. 2008. Evaluating unsupervised part-of-speech tagging for grammar induction. In *COLING*.

W. P. Headden, III, M. Johnson, and D. McClosky. 2009. Improving unsupervised dependency parsing with richer contexts and smoothing. In *NAACL-HLT*.

D. Hillard, Z. Huang, H. Ji, R. Grishman, D. Hakkani-Tür, M. Harper, M. Ostendorf, and W. Wang. 2006. Impact of automatic comma prediction on POS/name tagging of speech. In *IEEE/ACL: SLT*.

M. Johnson. 1998. PCFG models of linguistic tree representations. *Computational Linguistics*, 24.

B. E. M. Jones. 1994. Exploring the role of punctuation in parsing natural text. *COLING*.

J. G. Kahn, M. Lease, E. Charniak, M. Johnson, and M. Ostendorf. 2005. Effective use of prosody in parsing conversational speech. In *HLT-EMNLP*.

J.-H. Kim and P. C. Woodland. 2002. Implementation of automatic capitalisation generation systems for speech input. In *ICASSP*.

D. Klein and C. D. Manning. 2004. Corpus-based induction of syntactic structure: Models of dependency and constituency. In *ACL*.

Y.-S. Lee, S. Roukos, Y. Al-Onaizan, and K. Papineni. 2006. IBM spoken language translation system. In *TC-STAR: Speech-to-Speech Translation*.

Z. Li and M. Sun. 2009. Punctuation as implicit annotations for Chinese word segmentation. *Computational Linguistics*, 35.

X. Li, C. Zong, and R. Hu. 2005. A hierarchical parsing approach with punctuation processing for long Chinese sentences. In *IJCNLP*.

D. Lin. 1998. Dependency-based evaluation of MINIPAR. In *Evaluation of Parsing Systems*.

W. Lu and H. T. Ng. 2010. Better punctuation prediction with dynamic conditional random fields. In *EMNLP*.

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

E. Matusov, A. Mauser, and H. Ney. 2006. Automatic sentence segmentation and punctuation prediction for spoken language translation. In *IWSLT*.

I. D. Melamed. 1997. Measuring semantic entropy. In *ACL-SIGLEX: Tagging Text with Lexical Semantics*.

J. Nivre, J. Hall, S. Kübler, R. McDonald, J. Nilsson, S. Riedel, and D. Yuret. 2007a. The CoNLL 2007 shared task on dependency parsing. In *EMNLP-CoNLL*.

J. Nivre, J. Hall, J. Nilsson, A. Chanev, G. Eryiğit, S. Kübler, S. Marinov, and E. Marsi. 2007b. MaltParser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13.

G. Nunberg. 1990. *The Linguistics of Punctuation*. CSLI Publications.

Y. Ohyama, T. Fukushima, T. Shutoh, and M. Shutoh. 1986. A sentence analysis method for a Japanese book reading machine for the blind. In *ACL*.

M. A. Paskin. 2001. Grammatical bigrams. In *NIPS*.

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

J. Pitman and M. Yor. 1997. The two-parameter Poisson-Dirichlet distribution derived from a stable subordinator. *Annals of Probability*, 25.

S. Pradhan, K. Hacioglu, W. Ward, J. H. Martin, and D. Jurafsky. 2005. Semantic role chunking combining complementary syntactic views. In *CoNLL*.

V. Punyakanok, D. Roth, W.-t. Yih, and D. Zimak. 2005. Learning and inference over constrained output. In *IJCAI*.

B. E. Roark. 2001. *Robust Probabilistic Predictive Syntactic Processing: Motivations, Models, and Applications*. Ph.D. thesis, Brown University.

D. Roth and W.-t. Yih. 2005. Integer linear programming inference for conditional random fields. In *ICML*.

Y. Seginer. 2007. Fast unsupervised incremental parsing. In *ACL*.

D. D. Sleator and D. Temperley. 1993. Parsing English with a link grammar. In *IWPT*.

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

V. I. Spitkovsky, D. Jurafsky, and H. Alshawi. 2010b. Profiting from mark-up: Hyper-text annotations for guided parsing. In *ACL*.

E. Stamatatos, N. Fakotakis, and G. Kokkinakis. 2000. Text genre detection using common word frequencies. In *COLING*.

W. Y. Wang and K. R. McKeown. 2010. "Got you!": Automatic vandalism detection in Wikipedia with web-based shallow syntactic-semantic modeling. In *COLING*.

M. White and R. Rajkumar. 2008. A more precise analysis of punctuation for broad-coverage surface realization with CCG. In *GEAF*.

K. C. Yeh. 2003. Bilingual sentence alignment based on punctuation marks. In *ROCLING: Student*.