Simple Unsupervised Grammar Induction from Raw Text with Cascaded Finite State Models

Elias Ponvert, Jason Baldridge and Katrin Erk
Department of Linguistics
The University of Texas at Austin
Austin, TX 78712
{ponvert,jbaldrid,katrin.erk}@mail.utexas.edu

Abstract

We consider a new subproblem of unsupervised parsing from raw text, unsupervised partial parsing—the unsupervised version of text chunking. We show that addressing this task directly, using probabilistic finite-state methods, produces better results than relying on the local predictions of a current best unsupervised parser, Seginer's (2007) CCL. These finite-state models are combined in a cascade to produce more general (full-sentence) constituent structures; doing so outperforms CCL by a wide margin in unlabeled PARSEVAL scores for English, German and Chinese. Finally, we address the use of phrasal punctuation as a heuristic indicator of phrasal boundaries, both in our system and in CCL.

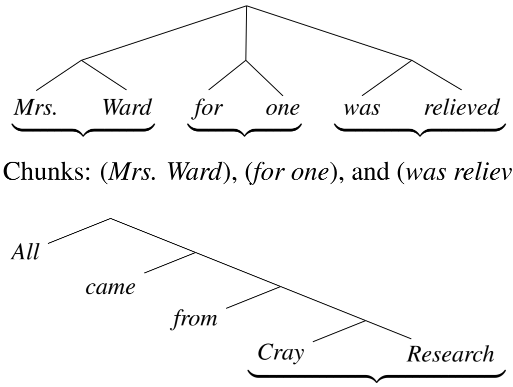
1 Introduction

Unsupervised grammar induction has been an active area of research in computational linguistics for over twenty years (Lari and Young, 1990; Pereira and Schabes, 1992; Charniak, 1993). Recent work (Headden III et al., 2009; Cohen and Smith, 2009; Hänig, 2010; Spitkovsky et al., 2010) has largely built on the dependency model with valence of Klein and Manning (2004), and is characterized by its reliance on gold-standard part-of-speech (POS) annotations: the models are trained on and evaluated using sequences of POS tags rather than raw tokens. This is also true for models which are not successors of Klein and Manning (Bod, 2006; Hänig, 2010).

An exception which learns from raw text and makes no use of POS tags is the *common cover links* parser (CCL, Seginer 2007). CCL established state-of-the-art results for unsupervised *constituency* parsing from raw text, and it is also incremental and extremely fast for both learning and parsing. Unfortunately, CCL is a non-probabilistic algorithm based on a complex set of inter-relating heuristics and a non-standard (though interesting) representation of constituent trees. This makes it hard to extend. Note that although Reichart and Rappoport (2010) improve on Seginer's results, they do so by selecting training sets to best match the particular test sentences—CCL itself is used without modification.

Ponvert et al. (2010) explore an alternative strategy of *unsupervised partial parsing*: directly predicting low-level constituents based solely on word co-occurrence frequencies. Essentially, this means segmenting raw text into multiword constituents. In that paper, we show—somewhat surprisingly—that CCL's performance is mostly dependent on its effectiveness at identifying low-level constituents. In fact, simply extracting non-hierarchical multiword constituents from CCL's output and putting a right-branching structure over them actually works *better* than CCL's own higher level predictions. This result suggests that improvements to low-level constituent prediction will ultimately lead to further gains in overall constituent parsing.

Here, we present such an improvement by using probabilistic finite-state models for phrasal segmentation from raw text. The task for these models is chunking, so we evaluate performance on identification of multiword chunks of all constituent types as well as only noun phrases. Our unsupervised chunkers extend straightforwardly to a cascade that predicts higher levels of constituent structure, similar to the supervised approach of Brants (1999). This forms an overall unsupervised parsing system that outperforms CCL by a wide margin.



(a) Chunks: (*Mrs. Ward*), (*for one*), and (*was relieved*)

(b) Only one chunk extracted: (*Cray Research*)

Fig. 1: Examples of constituent chunks extracted from syntactic trees

2 Data

We use the standard data sets for unsupervised constituency parsing research: for **English**, the Wall Street Journal subset of the Penn Treebank-3 (WSJ, Marcus et al. 1999); for **German**, the Negra corpus v2 (Krenn et al., 1998); for **Chinese**, the Penn Chinese Treebank v5.0 (CTB, Palmer et al., 2006). We lower-case text but otherwise do not alter the raw text of the corpus. Sentence segmentation and tokenization from the treebank is used. As in previous work, punctuation is not used for evaluation.

In much unsupervised parsing work the test sentences are included in the training material. Like Cohen and Smith, Headden III et al., Spitkovsky et al., we depart from this experimental setup and keep the evaluation sets blind to the models during training. For English (WSJ) we use sections 00-22 for training, section 23 for test and we develop using section 24; for German (Negra) we use the first 18602 sentences for training, the last 1000 sentences for development and the penultimate 1000 sentences for testing; for Chinese (CTB) we adopt the data-split of Duan et al. (2007).

WSJ	Chunks	203K	
	NPs	172K	
	Chnk ∩ NPs	161K	
Negra	Chunks	59K	
	NPs	33K	
	Chnk ∩ NPs	23K	
CTB	Chunks	92K	
	NPs	56K	
	Chnk ∩ NPs	43K	

Table 1: Constituent chunks and base NPs in the datasets.

		% constituents	% words
WSJ	Chunks	32.9	57.7
	NPs	27.9	53.1
Negra	Chunks	45.4	53.6
	NPs	25.5	42.4
CTB	Chunks	32.5	55.4
	NPs	19.8	42.9

Table 2: Percentage of gold standard constituents and words under constituent chunks and base NPs.

3 Tasks and Benchmark

Evaluation. By **unsupervised partial parsing**, or simply **unsupervised chunking**, we mean the segmentation of raw text into (non-overlapping) multiword constituents. The models are intended to capture local constituent structure – the lower branches of a constituent tree. For this reason we evaluate using what we call *constituent chunks*, the subset of gold standard constituents which are i) branching (multiword) but ii) non-hierarchical (do not contain subconstituents). We also evaluate our models based on their performance at identifying base noun phrases, NPs that do not contain nested NPs.

Examples of constituent chunks extracted from treebank constituent trees are in Fig. 1. In English newspaper text, constituent chunks largely correspond with base NPs, but this is less the case with Chinese and German. Moreover, the relationship between NPs and constituent chunks is not a subset relation: some base NPs do have internal constituent structure. The numbers of constituent chunks and NPs for the training datasets are in Table 1. The percentage of constituents in these datasets which fall under these definitions, and the percentage of words under these constituents, are in Table 2.

For parsing, the standard unsupervised parsing metric is unlabeled PARSEVAL. It measures precision and recall on constituents produced by a parser as compared to gold standard constituents.

CCL benchmark. We use Seginer's CCL as a benchmark for several reasons. First, there is a free/open-source implementation facilitating experimental replication and comparison.[1] More importantly, until recently it was the only unsupervised raw text constituent parser to produce results competitive with systems which use gold POS tags (Klein and Manning, 2002; Klein and Manning, 2004; Bod, 2006) – and the recent improved raw-text parsing results of Reichart and Rappoport (2010) make direct use of CCL without modification. There are other raw-text parsing systems of note, EMILE (Adriaans et al., 2000), ABL (van Zaanen, 2000) and ADIOS (Solan et al., 2005); however, there is little consistent treebank-based evaluation of these models. One study by Cramer (2007) found that none of the three performs particularly well under treebank evaluation. Finally, CCL outperforms most published POS-based models when those models are trained on unsupervised word classes rather than gold POS tags. The only exception we are aware of is Hänig's (2010) unsuParse+, which outperforms CCL on Negra, though this is shown only for sentences with ten or fewer words.

Phrasal punctuation. Though punctuation is usually entirely ignored in unsupervised parsing research, Seginer (2007) departs from this in one key aspect: the use of *phrasal punctuation* – punctuation symbols that often mark phrasal boundaries within a sentence. These are used in two ways: i) they impose a hard constraint on constituent spans, in that no constituent (other than sentence root) may extend over a punctuation symbol, and ii) they contribute to the model, specifically in terms of the statistics of words seen adjacent to a phrasal boundary. We follow this convention and use the following set:

`. ? ! ; , -- 。 、`

The last two are ideographic full-stop and comma.[2]

4 Unsupervised partial parsing

We learn partial parsers as constrained sequence models over tags encoding local constituent structure (Ramshaw and Marcus, 1995). A simple tagset is unlabeled BIO, which is familiar from supervised chunking and named-entity recognition: the tag *B* denotes the beginning of a chunk, *I* denotes membership in a chunk and *O* denotes exclusion from any chunk. In addition we use the tag *STOP* for sentence boundaries and phrasal punctuation.

HMMs and PRLGs. The models we use for unsupervised partial parsing are hidden Markov models, and a generalization we refer to as probabilistic right linear grammars (PRLGs). An HMM models a sequence of observed states (words) $\mathbf{x} = \{x_1, x_2, \ldots, x_N\}$ and a corresponding set of hidden states $\mathbf{y} = \{y_1, y_2, \ldots, y_N\}$. HMMs may be thought of as a special case of probabilistic context-free grammars, where the non-terminal symbols are the hidden state space, terminals are the observed states and rules are of the form NONTERM $\rightarrow$ TERM NONTERM (assuming y_1 and y_N are fixed and given). So, the emission and transition emanating from y_n would be characterized as a PCFG rule $y_n \rightarrow x_n\ y_{n+1}$. HMMs factor rule probabilities into emission and transition probabilities:

$$\begin{aligned} P(y_n \rightarrow x_n\ y_{n+1}) &= P(x_n, y_{n+1}|y_n) \\ &\approx P(x_n|y_n)\ P(y_{n+1}|y_n). \end{aligned}$$

However, without making this independence assumption, we can model right linear rules directly:

$$P(x_n, y_{n+1}|y_n) = P(x_n|y_n, y_{n+1})\ P(y_{n+1}|y_n).$$

So, when we condition emission probabilities on both the current state y_n and the next state y_{n+1}, we have an exact model. This direct modeling of the right linear grammar rule $y_n \rightarrow x_n\ y_{n+1}$ is what we call a probabilistic right-linear grammar. To be clear, a PRLG is just an HMM without the independence of emissions and transitions. See Smith and Johnson (2007) for a discussion, where they refer to PRLGs as Mealy HMMs.

We use expectation maximization to estimate model parameters. For the E step, the forward-backward algorithm (Rabiner, 1989) works identically for the HMM and PRLG. For the M step, we use maximum likelihood estimation with additive smoothing on the emissions probabilities. So, for the HMM and PRLG models respectively, for words

[1] `http://www.seggu.net/ccl`

[2] This set is essentially that of Seginer (2007). While it is clear from our analysis of CCL that it does make use of phrasal punctuation in Chinese, we are not certain whether ideographic comma is included.

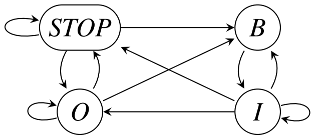


Fig. 2: Possible tag transitions as a state diagram.

	STOP	*B*	*I*	*O*
STOP	.33	.33		.33
B			1	
I	.25	.25	.25	.25
O	.33	.33		.33

Fig. 3: Uniform initialization of transition probabilities subject to the constraints in Fig. 2: rows correspond to antecedent state, columns to following state.

w and tags s, t:

$$\hat{P}(w|t) = \frac{C(t,w) + \lambda}{C(t) + \lambda V}$$

$$\hat{P}(w|s,t) = \frac{C(t,w,s) + \lambda}{C(t,s) + \lambda V}$$

where C are the soft counts of emissions $C(t,w)$, rules $C(t,w,s) = C(t \to w\ s)$, tags $C(t)$ and transitions $C(t,s)$ calculated during the E step; V is the number of terms w, and λ is a smoothing parameter. We fix $\lambda = .1$ for all experiments; more sophisticated smoothing could avoid dependence on λ.

We do not smooth transition probabilities (so $\hat{P}(s|t) = C(t,s)/C(t)$) for two reasons. First, with four tags, there is no data-sparsity concern with respect to transitions. Second, the nature of the task imposes certain constraints on transition probabilities: because we are only interested in multiword chunks, we expressly do not want to generate a *B* following a *B* – in other words $P(B|B) = 0$.

These constraints boil down to the observation that the *B* and *I* states will only be seen in *BII** sequences. This may be expressed via the state transition diagram in Fig. 2. The constraints of also dictate the initial model input to the EM process. We use uniform probability distributions subject to the constraints of Fig. 2. So, initial model transition probabilities are given in Fig. 3. In EM, if a parameter is equal to zero, subsequent iterations of the EM process will not "unset" this parameter; thus, this form of initialization is a simple way of encoding constraints on model parameters. We also experimented with random initial models (subject to the constraints in Fig. 2). Uniform initialization usually works slightly better; also, uniform initialization does not require multiple runs of each experiment, as random initialization does.

Motivating the HMM and PRLG. This approach – encoding a chunking problem as a tagging problem and learning to tag with HMMs – goes back to Ramshaw and Marcus (1995). For unsupervised learning, the expectation is that the model will learn to generalize on phrasal boundaries. That is, the models will learn to associate terms like *the* and *a*, which often occur at the beginnings of sentences and rarely at the end, with the tag *B*, which cannot occur at the end of a sentence. Likewise common nouns like *company* or *asset*, which frequently occur at the ends of sentences, but rarely at the beginning, will come to be associated with the *I* tag, which cannot occur at the beginning.

The basic motivation for the PRLG is the assumption that information is lost due to the independence assumption characteristic of the HMM. With so few states, it is feasible to experiment with the more fine-grained PRLG model.

Evaluation. Using the low-level predictions of CCL as as benchmark, we evaluate the HMM and PRLG chunkers on the tasks of constituent chunk and base NP identification. Models were initialized uniformly as illustrated in Fig. 3. Sequence models learn via EM. We report accuracy only after convergence, that is after the change in full dataset perplexity (log inverse probability) is less than $\%.01$ between iterations. Precision, recall and *F*-score are reported for full constituent identification – brackets which do not match the gold standard exactly are false positives.

Model performance results on held-out test datasets are reported in Table 3. 'CCL' refers to the lowest-level constituents extracted from full CCL output, as a benchmark chunker. The sequence models outperform the CCL benchmark at both tasks and on all three datasets. In most cases, the PRLG sequence model performs better than the HMM; the exception is CTB, where the PRLG model is behind the HMM in evaluation, as well as behind CCL.

As the lowest-level constituents of CCL were not specifically designed to describe chunks, we also

		English / WSJ			German / Negra			Chinese / CTB		
Task	Model	Prec	Rec	*F*	Prec	Rec	*F*	Prec	Rec	*F*
Chunking	CCL	57.5	53.5	55.4	28.4	29.6	29.0	23.5	23.9	23.7
	HMM	53.8	62.2	57.7	35.0	37.7	36.3	**37.4**	**41.3**	**39.3**
	PRLG	**76.2**	**63.9**	**69.5**	**39.6**	**47.8**	**43.3**	23.0	18.3	20.3
NP	CCL	46.2	51.1	48.5	15.6	29.2	20.3	10.4	17.3	13.0
	HMM	47.7	65.6	55.2	23.8	46.2	31.4	17.0	**30.8**	21.9
	PRLG	**76.8**	**76.7**	**76.7**	**24.6**	**53.4**	**33.6**	**21.9**	28.5	**24.8**

Table 3: Unsupervised chunking results for local constituent structure identification and NP chunking on held-out test sets. CCL refers to the lowest constituents extracted from CCL output.

	WSJ	Negra	CTB
Chunking	57.8	36.0	25.5
NPs	57.8	38.8	23.2

Table 4: Recall of CCL on the chunking tasks.

checked the recall of *all* brackets generated by CCL against gold-standard constituent chunks. The results are given in Table 4. Even compared to this, the sequence models' recall is almost always higher.

The sequence models, as well as the CCL benchmark, show relatively low precision on the Negra corpus. One possible reason for this lies in the design decision of Negra to use relatively flat tree structures. As a result, many structures that in other treebanks would be prepositional phrases with embedded noun phrases – and thus non-local constituents – are flat prepositional phrases here. Examples include "auf die Wiesbadener Staatsanwaelte" (on Wiesbaden's district attorneys) and "in Hannovers Nachbarstadt" (in Hannover's neighbor city).

In fact, in Negra, the sequence model chunkers often find NPs embedded in PPs, which are not annotated as such. For instance, in the PP "hinter den Kulissen" (behind the scenes), both the PRLG and HMM chunkers identify the internal NP, though this is not identified in Negra and thus considered a false positive. The fact that the HMM and PRLG have higher recall on NP identification on Negra than precision is further evidence towards this.

Comparing the HMM and PRLG. To outline some of the factors differentiating the HMM and PRLG, we focus on NP identification in WSJ.

The PRLG has higher precision than the HMM, while the two models are closer in recall. Comparing the predictions directly, the two models often have the same correct predictions and often miss the same gold standard constituents. The improved results of the PRLG are based mostly on the fewer overall brackets predicted, and thus fewer false positives: for WSJ the PRLG incorrectly predicts 2241 NP constituents compared to 6949 for the HMM. Table 5 illustrates the top 5 POS sequences of the false positives predicted by the HMM.[3] (Recall that we use gold standard POS *only* for post-experiment results analysis—the model itself does not have access to them.) By contrast, the sequence representing the largest class of errors of the PRLG is DT NN, with 165 errors – this sequence represents the largest class of predictions for both models.

POS Sequence	# of errors
TO VB	673
NNP NNP	450
MD VB	407
DT JJ	368
DT NN	280

Table 5: Top 5 POS sequences of the false positives predicted by the HMM.

Two of the top classes of errors, MD VB and TO VB, represent verb phrase constituents, which are often predicted by the HMM chunker, but not by the PRLG. The class represented by NNP NNP corresponds with the tendency of the HMM chunker to split long proper names: for example, it systematically splits *new york stock exchange* into two chunks, (*new york*) (*stock exchange*), whereas the PRLG chunker predicts a single four-word chunk.

The most interesting class is DT JJ, which represents the difficulty the HMM chunker has at dis-

[3] For the Penn Treebank tagset, see Marcus et al. (1993).

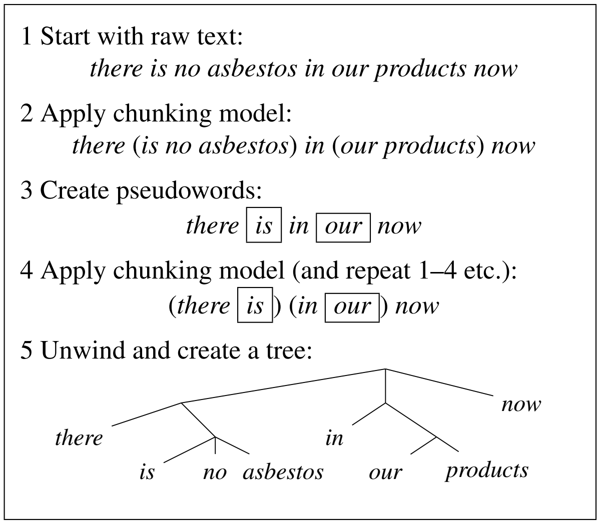

1 Start with raw text:
there is no asbestos in our products now
2 Apply chunking model:
there (is no asbestos) in (our products) now
3 Create pseudowords:
there is in our now
4 Apply chunking model (and repeat 1–4 etc.):
(there is) (in our) now
5 Unwind and create a tree:
there
is
no
asbestos
in
our
products
now

Fig. 4: Cascaded chunking illustrated. Pseudowords are indicated with boxes.

tinguishing determiner-adjective from determiner-noun pairs. The PRLG chunker systematically gets DT JJ NN trigrams as chunks. The greater context provided by right branching rules allows the model to explicitly estimate separate probabilities for $P(I \rightarrow \textit{recent } I)$ versus $P(I \rightarrow \textit{recent } O)$. That is, *recent* within a chunk versus ending a chunk. Bigrams like *the acquisition* allow the model to learn rules $P(B \rightarrow \textit{the } I)$ and $P(I \rightarrow \textit{acquisition } O)$. So, the PRLG is better able to correctly pick out the trigram chunk (*the recent acquisition*).

5 Constituent parsing with a cascade of chunkers

We use cascades of chunkers for full constituent parsing, building hierarchical constituents bottom-up. After chunking is performed, all multiword constituents are collapsed and represented by a single pseudoword. We use an extremely simple, but effective, way to create pseudoword for a chunk: pick the term in the chunk with the highest corpus frequency, and mark it as a pseudoword. The sentence is now a string of symbols (normal words and pseudowords), to which a subsequent unsupervised chunking model is applied. This process is illustrated in Fig. 4.

Each chunker in the cascade chunks the raw text, then regenerates the dataset replacing chunks with pseudowords; this process is iterated until no new chunks are found. The separate chunkers in the cascade are referred to as *levels*. In our experiments the cascade process took a minimum of 5 levels, and a maximum of 7. All chunkers in the cascade have the same settings in terms of smoothing, the tagset and initialization.

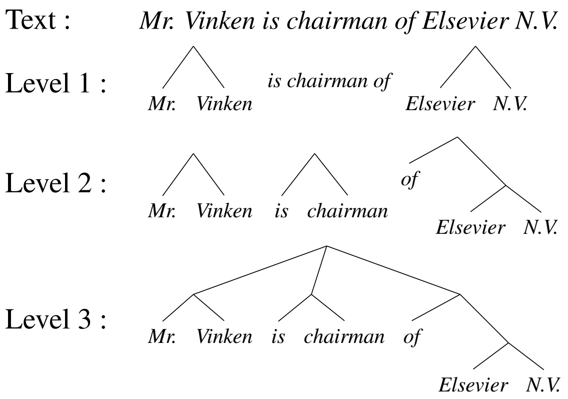


Fig. 5: PRLG cascaded chunker output.

		NPs		PPs	
		Lev 1	Lev 2	Lev 1	Lev 2
WSJ	HMM	66.5	68.1	20.6	70.2
	PRLG	77.5	78.3	9.1	77.6
Negra	HMM	54.7	62.3	24.8	48.1
	PRLG	61.6	65.2	40.3	44.0
CTB	HMM	33.3	35.4	34.6	38.4
	PRLG	30.9	33.6	31.6	47.1

Table 7: NP and PP recall at cascade levels 1 and 2. The level 1 NP numbers differ from the NP chunking numbers from Table 3 since they include root-level constituents which are often NPs.

Evaluation. Table 6 gives the unlabeled PARSEVAL scores for CCL and the two finite-state models. PRLG achieves the highest *F*-score for all datasets, and does so by a wide margin for German and Chinese. CCL does achieve higher recall for English.

While the first level of constituent analysis has high precision and recall on NPs, the second level often does well finding prepositional phrases (PPs), especially in WSJ; see Table 7. This is illustrated in Fig. 5. This example also illustrates a PP attachment error, which are a common problem for these models.

We also evaluate using short – 10-word or less – sentences. That said, we maintain the training/test split from before. Also, making use of the open

Parsing	English / WSJ			German / Negra			Chinese / CTB		
Model	Prec	Rec	*F*	Prec	Rec	*F*	Prec	Rec	*F*
CCL	53.6	**50.0**	51.7	33.4	32.6	33.0	37.0	21.6	27.3
HMM	48.2	43.6	45.8	30.8	**50.3**	38.2	43.0	29.8	35.2
PRLG	**60.0**	49.4	**54.2**	**38.8**	47.4	**42.7**	**50.4**	**32.8**	**39.8**

Table 6: Unlabeled PARSEVAL scores for cascaded models.

source implementation by F. Luque,[4] we compare on WSJ and Negra to the *constituent context model* (CCM) of Klein and Manning (2002). CCM learns to predict a set of brackets over a string (in practice, a string of POS tags) by jointly estimating constituent and distituent strings and contexts using an iterative EM-like procedure (though, as noted by Smith and Eisner (2004), CCM is deficient as a generative model). Note that this comparison is methodologically problematic in two respects. On the one hand, CCM is evaluated using gold standard POS sequences as input, so it receives a major source of supervision not available to the other models. On the other hand, the other models use punctuation as an indicator of constituent boundaries, but all punctuation is dropped from the input to CCM. Also, note that CCM performs better when trained on short sentences, so here CCM is trained only on the 10-word-or-less subsets of the training datasets.[5]

The results from the cascaded PRLG chunker are near or better than the best performance by CCL or CCM in these experiments. These and the full-length parsing results suggest that the cascaded chunker strategy generalizes better to longer sentences than does CCL. CCM does very poorly on longer sentences, but does not have the benefit of using punctuation, as do the raw text models; unfortunately, further exploration of this trade-off is beyond the scope of this paper. Finally, note that CCM has higher recall, and lower precision, generally, than the raw text models. This is due, in part, to the chart structure used by CCM in the calculation of constituent and distituent probabilities: as in CKY parsing, the chart structure entails the trees predicted will be binary-branching. CCL and the cascaded models can predict higher-branching constituent structures, so fewer constituents are predicted overall.

		Prec	Rec	*F*
WSJ	CCM	*62.4*	***81.4***	*70.7*
	CCL	71.2	73.1	**72.1**
	HMM	64.4	64.7	64.6
	PRLG	**74.6**	66.7	70.5
Negra	CCM	*52.4*	***83.4***	***64.4***
	CCL	52.9	54.0	53.0
	HMM	47.7	72.0	57.4
	PRLG	**56.3**	72.1	63.2
CTB	CCL	54.4	44.3	48.8
	HMM	55.8	53.1	54.4
	PRLG	**62.7**	**56.9**	**59.6**

Table 8: Evaluation on 10-word-or-less sentences. CCM scores are italicized as a reminder that CCM uses gold-standard POS sequences as input, so its results are not strictly comparable to the others.

6 Phrasal punctuation revisited

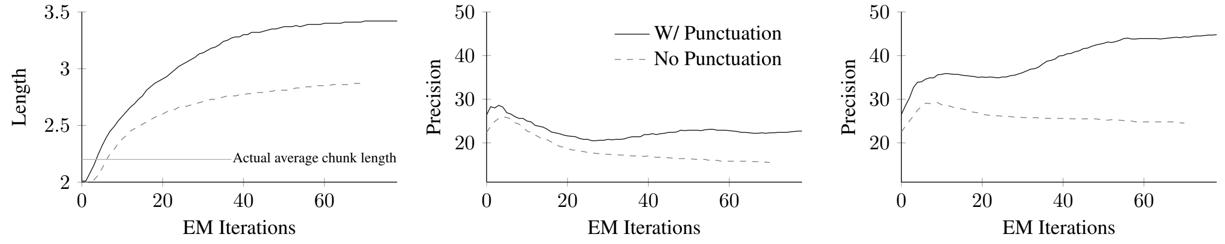
Up to this point, the proposed models for chunking and parsing use phrasal punctuation as a phrasal separator, like CCL. We now consider how well these models perform in absence of this constraint.

Table 9a provides comparison of the sequence models' performance on the constituent chunking task without using phrasal punctuation in training and evaluation. The table shows absolute improvement (+) or decline (−) in precision and recall when phrasal punctuation is removed from the data. The punctuation constraint seems to help the chunkers some, but not very much; ignoring punctuation seems to *improve* chunker results for the HMM on Chinese. Overall, the effect of phrasal punctuation on the chunker models' performance is not clear.

The results for cascaded parsing differ strongly from those for chunking, as Table 9b indicates. Using phrasal punctuation to constrain bracket prediction has a larger impact on cascaded parsing re-

[4] `http://www.cs.famaf.unc.edu.ar/~francolq/en/proyectos/dmvccm/`

[5] This setup is the same as Seginer's (2007), except the train/test split.



a) Average Predicted Constituent Length

b) Chunking Precision

c) Precision at All Brackets

Fig. 6: Behavior of the PRLG model on CTB over the course of EM.

	WSJ		Negra		CTB	
	Prec	Rec	Prec	Rec	Prec	Rec
HMM	−5.8	−9.8	−0.1	−0.4	+0.7	+4.9
PRLG	−2.5	−2.1	−2.1	−2.1	−7.0	+1.2

a) Constituent Chunking

	WSJ		Negra		CTB	
	Prec	Rec	Prec	Rec	Prec	Rec
CCL	−14.1	−13.5	−10.7	−4.6	−11.6	−6.0
HMM	−7.8	−8.6	−2.8	+1.7	−13.4	−1.2
PRLG	−10.1	−7.2	−4.0	−4.5	−22.0	−11.8

b) (Cascade) Parsing

Table 9: Effects of dropping phrasal punctuation in unsupervised chunking and parsing evaluations relative to Tables 3 and 6.

sults almost across the board. This is not surprising: while performing unsupervised partial parsing from raw text, the sequence models learn two general patterns: i) they learn to chunk rare sequences, such as named entities, and ii) they learn to chunk high-frequency function words next to lower frequency content words, which often correlate with NPs headed by determiners, PPs headed by prepositions and VPs headed by auxiliaries. When these patterns are themselves replaced with pseudowords (see Fig. 4), the models have fewer natural cues to identify constituents. However, within the degrees of freedom allowed by punctuation constraints as described, the chunking models continue to find relatively good constituents.

While CCL makes use of phrasal punctuation in previously reported results, the open source implementation allows it to be evaluated without this constraint. We did so, and report results in Table 9b. CCL is, in fact, very sensitive to phrasal punctuation. Comparing CCL to the cascaded chunkers when none of them use punctuation constraints, the cascaded chunkers (both HMMs and PRLGs) outperform CCL for each evaluation and dataset.

For the CTB dataset, best chunking performance and cascaded parsing performance flips from the HMM to the PRLG. More to the point, the PRLG is actually with worst performing model at the constituent chunking task, but the best performing cascade parser; also, this model has the most serious degrade in performance when phrasal punctuation is dropped from input. To investigate, we track the performance of the chunkers on the development dataset over iterations of EM. This is illustrated in Fig. 6 with the PRLG model. First of all, Fig. 6a reveals the average length of the constituents predicted by the PRLG model increases over the course of EM. However, the average constituent chunk length is 2.22. So, the PRLG chunker is predicting constituents that are longer than the ones targeted in the constituent chunking task: regardless of whether they are legitimate constituents or not, often they will likely be counted as false positives in this evaluation. This is confirmed by observing the constituent chunking precision in Fig. 6b, which peaks when the average predicted constituent length is about the same the actual average length of those in the evaluation. The question, then, is whether the longer chunks predicted correspond to actual constituents or not. Fig. 6c shows that the PRLG, when constrained by phrasal punctuation, does continue to improve its constituent prediction accuracy over the course of EM. These correctly predicted constituents are not counted as such in the constituent chunking or base NP evaluations, but they factor directly into improved accuracy when this model is part of a cascade.

7 Related work

Our task is the unsupervised analogue of chunking (Abney, 1991), popularized by the 1999 and 2000 Conference on Natural Language Learning shared tasks (Tjong et al., 2000). In fact, our models follow Ramshaw and Marcus (1995), treating structure prediction as sequence prediction using BIO tagging.

In addition to Seginer's CCL model, the unsupervised parsing model of Gao and Suzuki (2003) and Gao et al. (2004) also operates on raw text. Like us, their model gives special treatment to local constituents, using a language model to characterize phrases which are linked via a dependency model. Their output is not evaluated directly using treebanks, but rather applied to several information retrieval problems.

In the supervised realm, Hollingshead et al. (2005) compare context-free parsers with finite-state partial parsing methods. They find that full parsing maintains a number of benefits, in spite of the greater training time required: they can train on less data more effectively than chunkers, and are more robust to shifts in textual domain.

Brants (1999) reports a supervised cascaded chunking strategy for parsing which is strikingly similar to the methods proposed here. In both, Markov models are used in a cascade to predict hierarchical constituent structure; and in both, the parameters for the model at each level are estimated independently. There are major differences, though: the models here are learned from raw text without tree annotations, using EM to train parameters; Brants' cascaded Markov models use supervised maximum likelihood estimation. Secondly, between the separate levels of the cascade, we collapse constituents into symbols which are treated as tokens in subsequent chunking levels; the Markov models in the higher cascade levels in Brants' work actually emit constituent structure. A related approach is that of Schuler et al. (2010), who report a supervised hierarchical hidden Markov model which uses a right-corner transform. This allows the model to predict more complicated trees with fewer levels than in Brants' work or this paper.

8 Conclusion

In this paper we have introduced a new subproblem of unsupervised parsing: unsupervised partial parsing, or unsupervised chunking. We have proposed a model for unsupervised chunking from raw text that is based on standard probabilistic finite-state methods. This model produces better local constituent predictions than the current best unsupervised parser, CCL, across datasets in English, German, and Chinese. By extending these probabilistic finite-state methods in a cascade, we obtain a general unsupervised parsing model. This model outperforms CCL in PARSEVAL evaluation on English, German, and Chinese.

Like CCL, our models operate from raw (albeit segmented) text, and like it our models decode very quickly; however, unlike CCL, our models are based on standard and well-understood computational linguistics technologies (hidden Markov models and related formalisms), and may benefit from new research into these core technologies. For instance, our models may be improved by the application of (unsupervised) discriminative learning techniques with features (Berg-Kirkpatrick et al., 2010); or by incorporating topic models and document information (Griffiths et al., 2005; Moon et al., 2010).

UPPARSE, the software used for the experiments in this paper, is available under an open-source license to facilitate replication and extensions.[6]

Acknowledgments. This material is based upon work supported in part by the U. S. Army Research Laboratory and the U. S. Army Research Office under grant number W911NF-10-1-0533. Support for the first author was also provided by Mike Hogg Endowment Fellowship, the Office of Graduate Studies at The University of Texas at Austin.

This paper benefited from discussion in the Natural Language Learning reading group at UT Austin, especially from Collin Bannard, David Beaver, Matthew Lease, Taesun Moon and Ray Mooney. We also thank the three anonymous reviewers for insightful questions and helpful comments.

[6] `http://elias.ponvert.net/upparse`.

References

S. Abney. 1991. Parsing by chunks. In R. Berwick, S. Abney, and C. Tenny, editors, *Principle-based Parsing*. Kluwer.

P. W. Adriaans, M. Trautwein, and M. Vervoort. 2000. Towards high speed grammar induction on large text corpora. In *SOFSEM*.

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

R. Bod. 2006. Unsupervised parsing with U-DOP. In *CoNLL*.

T. Brants. 1999. Cascaded markov models. In *EACL*.

E. Charniak. 1993. *Statistical Language Learning*. MIT.

S. B. Cohen and N. A. Smith. 2009. Shared logistic normal distributions for soft parameter tying in unsupervised grammar induction. In *HLT-NAACL*.

B. Cramer. 2007. Limitations of current grammar induction algorithms. In *ACL-SRW*.

X. Duan, J. Zhao, and B. Xu. 2007. Probabilistic models for action-based Chinese dependency parsing. In *ECML/PKDD*.

J. Gao and H. Suzuki. 2003. Unsupervised learning of dependency structure for language modeling. In *ACL*.

J. Gao, J.Y. Nie, G. Wu, and G. Cao. 2004. Dependence language model for information retrieval. In *SIGIR*.

T. L. Griffiths, M. Steyvers, D. M. Blei, and J. M. Tenenbaum. 2005. Integrating topics and syntax. In *NIPS*.

C. Hänig. 2010. Improvements in unsupervised cooccurence based parsing. In *CoNLL*.

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

K. Hollingshead, S. Fisher, and B. Roark. 2005. Comparing and combining finite-state and context-free parsers. In *HLT-EMNLP*.

D. Klein and C. D. Manning. 2002. A generative constituent-context model for improved grammar induction. In *ACL*.

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

B. Krenn, T. Brants, W. Skut, and Hans Uszkoreit. 1998. A linguistically interpreted corpus of German newspaper text. In *Proceedings of the ESSLLI Workshop on Recent Advances in Corpus Annotation*.

K. Lari and S. J. Young. 1990. The estimation of stochastic context-free grammars using the inside-outside algorithm. *Computer Speech & Language*, 4:35 – 56.

M.P. Marcus, B. Santorini, and M.A. Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Compuational Linguistics*, pages 313–330.

M.P. Marcus, B. Santorini, M.A. Marcinkiewicz, and A. Taylor, 1999. *Treebank-3*. LDC.

T. Moon, J. Baldridge, and K. Erk. 2010. Crouching Dirichlet, hidden Markov model: Unsupervised POS tagging with context local tag generation. In *EMNLP*.

M. Palmer, F. D. Chiou, N. Xue, and T. K. Lee, 2005. *Chinese Treebank 5.0*. LDC.

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

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

L.R. Rabiner. 1989. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*.

L. A. Ramshaw and M. P. Marcus. 1995. Text chunking using transformation-based learning. In *Proc. of Third Workshop on Very Large Corpora*.

R. Reichart and A. Rappoport. 2010. Improved fully unsupervised parsing with Zoomed Learning. In *EMNLP*.

W. Schuler, S. AbdelRahman, T. Miller, and L. Schwartz. 2010. Broad-coverage parsing using human-like memory constraints. *Compuational Linguistics*, 3(1).

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

N. A. Smith and J. Eisner. 2004. Annealing techniques for unsupervised statistical language learning. In *ACL*.

N. A. Smith and M. Johnson. 2007. Weighted and probabilistic CFGs. *Computational Lingusitics*.

Z. Solan, D. Horn, E. Ruppin, and S. Edelman. 2005. Unsupervised learning of natural languages. *PNAS*, 102.

V. I. Spitkovsky, H. Alshawi, and D. Jurafsky. 2010. From baby steps to leapfrog: How "less is more" in unsupervised dependency parsing. In *NAACL-HLT*.

E. F. Tjong, K. Sang, and S. Buchholz. 2000. Introduction to the CoNLL-2000 Shared Task: Chunking. In *CoNLL-LLL*.

M. van Zaanen. 2000. ABL: Alignment-based learning. In *COLING*.