Co-training an Unsupervised Constituency Parser with Weak Supervision

Anonymous ACL submission

Abstract

We introduce a method for unsupervised parsing that relies on bootstrapping classifiers to identify if a node dominates a specific span in a sentence. There are two types of classifiers, an inside classifier that acts on a span, and an outside classifier that acts on everything outside of a given span. Through self-training and co-training with the two classifiers, we show that the interplay between them helps improve the accuracy of both, and as a result, effectively parse. A seed bootstrapping technique prepares the data to train these classifiers. Our analyses further validate that such an approach in conjunction with weak supervision using prior branching knowledge of a known language (left/right-branching) and minimal heuristics injects strong inductive bias into the parser, achieving 63.1 F_1 on the English (PTB) test set. In addition, we show the effectiveness of our architecture by evaluating on treebanks for Chinese (CTB) and Japanese (KTB) and achieve new state-of-the-art results.[1]

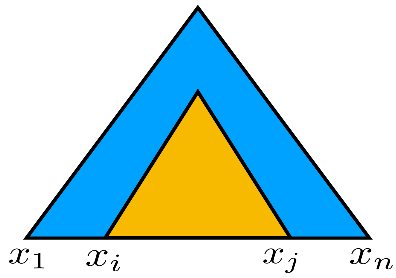

Figure 1: A depiction of a syntax tree, with the inside string as depicted by the sequence $x_i \cdots x_j$ and the outside string as depicted by the sequence $(x_1 \cdots x_{i-1}, x_{j+1} \cdots x_n)$ that provides external context for the inside representations.

1 Introduction

Pre-trained language models (PLMs) have become a standard tool in the Natural Language Processing (NLP) toolkit, offering the benefits of learning from large amounts of unlabeled data while providing modular function in many NLP tasks that require supervision. Recent work has shown that PLMs capture different types of linguistic regularities and information, for instance, the lower layers capture phrase-level information which becomes less prominent in the upper layers (Jawahar et al., 2019), span representations constructed from these models can encode rich syntactic phenomena, like the ability to track subject-verb agreement (Goldberg, 2019), dependency trees can be embedded within the geometry of BERT's hidden states (Hewitt and Manning, 2019), and most relevantly to this paper, syntactic information via self-attention mechanisms (Wang et al., 2019; Kim et al., 2020).

We offer another perspective on the way PLMs represent syntactic information. We demonstrate the usability of PLMs to capture syntactic information by developing an unsupervised parsing model that makes heavy use of PLMs. The learning algorithm is light in the injection of hard bias to parse text, emphasizing the role of PLMs in capturing syntactic information.

Our approach to unsupervised parsing is inspired by recent work in the area of spectral learning for parsing (Cohen et al., 2014, 2013) and unsupervised estimation of probabilistic context-free grammars (PCFGs; Clark and Fijalkow, 2020). At its core, our learning algorithm views the presence or absence of a node dominating a substring in the final parse tree as a latent variable, where patterns of co-occurrence of the string that the node dominates (the "inside" string) and the rest of the sentence (the "outside" string) dictate whether the node is present or not. With spectral learning for latent-variable PCFGs (L-PCFGs; Cohen et al., 2012) the notion of inside *trees* versus outside *trees* is important, but in our case, given that the trees are not present during learning, we have to further specialize it to extract information only from the strings.

[1] For code or data, please contact the authors.

Consider the diagram of a syntax tree in Figure 1, decomposed into two parts. Following the main notion in spectral learning, each of these parts (the orange part and the blue part) is a "view" of the whole tree that provides information on the identity of the node that spans the words $x_i \cdots x_j$. In the case of the tree being unobserved during training, we have to rely only on the substrings that are spanned by the blue part or the orange part, to hypothesize whether indeed a node exists there.

To represent the inside and outside views, we make use of PLMs. We encode these substrings, and then bootstrap a classifier that determines whether a given span is a constituent or not. The bootstrapping process alternates between the two views, and at each point adds predictions on the training set that it is confident about to train a new classifier. This can be thought of as a form of co-training (Yarowsky, 1995; Blum and Mitchell, 1998), a training technique that relies on multiple views of training instances. We formulate the task of identifying constituents and distituents (referring to spans that are not constituents) in a sentence as a binary classification task by devising a strategy to convert the unlabeled data into a classification task. Firstly, we build a sequence classification model by fine-tuning a Transformer-based PLM on the unlabeled training sentences to distinguish between the true and false inside strings of constituents. Secondly, we use the highly-confident inside strings to produce the outside strings. Additionally, through the use of semi-supervised learning techniques, we jointly use both the inside and outside passes to enrich the model's ability to determine the breakpoints in a sentence. Our final model achieves 63.1 sentence F_1 averaged over multiple runs with random seed on the Penn Treebank test set. We also report strong results for the Japanese and Chinese treebanks.

2 Problem Formulation and Inference

We give a treatment to the problem of unsupervised constituency parsing. In that setup, the training algorithm is given an unlabeled corpus (set of sentences) and its goal is to learn a function mapping a sentence x to an unlabeled phrase-structure tree y that indicates the constituents in x. In previous work with models such as the Constituent-Context Model (CCM; Klein and Manning 2002), the Dependency Model with Valence (DMV; Klein and Manning 2005), and Unsupervised Maximum Likelihood estimator for Data-Oriented Parsing (UML-DOP; Bod 2006), the parts of speech (POS) of the words in x are also given as input both during inference and during training, but we do not make use of such POS tags.

Inference While our learning algorithm is grammarless, for inference we make use of a dynamic programming algorithm, akin to CYK, to predict the parse tree. Inference assumes that each possible span in the tree was scored with a score function $s(i, j)$ where i and j are endpoints in the sentence. The score function is learned through our algorithm. We then proceed by finding the tree t^* such that:

$$t^* = \arg\max_{t \in \mathcal{T}} \sum_{(i,j) \in t} s(i, j),$$

where $\mathcal{T}$ is the set of possible binary trees over the sentence and $(i, j) \in t$, with a slight abuse of notation, denotes that the span (i, j) appears in t.

When $s(i, j)$ is the probability of a span (i, j) being in the correct tree, this formulation gives the tree with the highest expected number of correct constituents (Goodman, 1996). This formulation has been used recently by several unsupervised constituency parsing algorithms (Kim et al., 2019b,a; Cao et al., 2020; Li et al., 2020a).

3 Training Algorithm

At the core of our approach lies the notion of *inside* and *outside* strings. For a given sentence $x = x_1 \cdots x_n$ and a span (i, j), the inside string of span (i, j) is the sequence $x_i \cdots x_j$ while the outside string is the pair $(x_1 \cdots x_{i-1}, x_{j+1} \cdots x_n)$. We denote by $\boldsymbol{h}_{\text{in}}(i, j)$ representations for inside strings and $\boldsymbol{h}_{\text{out}}(i, j)$ representations for outside strings. Both are vectors derived from a PLM (RoBERTa (Liu et al., 2019), as we see later).

These two types of strings provide two views of a given possible splitting point in the syntax tree. We offer three ways, with increasing complexity, to bootstrap a score function that helps identify whether a node should dominate a given span. The main idea behind this bootstrapping is to start with a small seed set of training examples (x, i, j, b) where (i, j) is a span in a sentence x and $b \in \{0, 1\}$, depending on whether the span (i, j) is dominated by a node in the syntactic tree or not. Bootstrapping the seed set is dependent only on either the inside string or the outside string, and the corresponding classifier built from this bootstrapped seed set returns a probability $p(b \mid x, i, j)$.

Once a classifier is learned using the bootstrapping seed set, the classifier is applied on the training set, and the seed set is added to more examples where the classifier is confident of the label b. This is also known as self-training (McClosky et al., 2006, 2008).

In the next three sections, we present three learning algorithms of increasing complexity in their use of inside and outside strings.

3.1 Modeling Using Inside Strings

The inside model m_{in} which is modeled at a sentence level, computes an inside score $s_{\text{in}}(i, j)$ from the inside vector representation $\boldsymbol{h}_{\text{in}}(i, j)$ of each span in the unlabeled input training sentence $\mathbf{U}$. To compute $\boldsymbol{h}_{\text{in}}(i, j)$, we fine-tune the sequence classification model that encodes a fixed-vector representation for each token in the dataset. This captures the phrase information of the inner content in the span. In order to prepare the features for the inside model, we make use of a seed bootstrapping technique (Section 4.2.1). Once we build the inside model m_{in}, we get the most confidently-classified inside strings from $\mathbf{U}$ based on a set threshold $\tau = (\tau_{\text{min}}, \tau_{\text{max}})$. Here, τ_{min} and τ_{max}, form the confidence bounds to select distituents and constituents respectively. We select a random sample of c constituents and d distituents with appropriate labels from these most confident inside strings comprising the labeled inside set $\mathbf{I}$.

3.2 Modeling Using Inside and Outside Strings

To perform the iterative self-training procedure, we follow the steps as detailed in Figure 2. While building the outside model, we extract the tokens at the span boundaries of the pair of outside strings, which is of the form consisting of the triple (x_{i-1}, `[MASK]`, x_{j+1}). The outside model computes an outside score $s_{\text{out}}(i, j)$ from the outside vector representation $\boldsymbol{h}_{\text{out}}(i, j)$ of each span, which models the contextual information of the span. To compute $\boldsymbol{h}_{\text{out}}(i, j)$, we extract the triple for every span (i, j) in the dataset and fine-tune another sequence classification model that encodes a fixed-vector representation for each triple.

3.3 An Iterative Co-training Algorithm

Co-training (Blum and Mitchell, 1998) is a classic multi-view training method, which trains a classifier by exploiting two (or more) views of the training instances. Our final learning algorithm is indeed inspired by it, where we consider the inside and the outside strings to be the two views. Once we have the inside m_{in} and the outside classifiers m_{out} that are trained on their respective conditionally independent inside $\boldsymbol{h}_{\text{in}}(i, j)$ and outside $\boldsymbol{h}_{\text{out}}(i, j)$ feature sets, we can make use of an iterative approach. At each iteration, only the inside strings $\hat{\mathbf{I}}$ that are confident to be likely the insides of constituents and distituents according to the outside model are moved to the labeled training set of the inside model $\mathbf{I}$. Thus, the outside model (teacher) provides the labels to the inside strings on which the inside model (student) is uncertain. Similarly, only the outside strings $\hat{\mathbf{O}}$ that are confident to be the likely outsides of constituents and distituents according to the inside model are moved to the labeled training set of the outside model $\mathbf{O}$. Thus, the inside model provides the labels to the outside strings on which the outside model is uncertain. We describe the steps in Figure 3. Finally, we combine the scores obtained by the inside and the outside model to get the score $s(i, j)$ for each span:

$$s(i, j) = s_{\text{in}}(i, j) \cdot s_{\text{out}}(i, j).$$

Co-training requires the two views to be independent of each other conditioned on the label of the training instance. This is the type of assumption that, for example, PCFGs satisfy, when breaking a tree into an outside and inside tree: the two trees are conditionally independent given the nonterminal that connects them. In our case, we satisfy this

Inputs: $\mathbf{I}$ represents the labeled inside set; $\mathbf{U}$ is a set of Unlabeled training sentences;

Algorithm:

- Loop for K iterations:
  1. Learn the inside classifier m_{in} based on $\boldsymbol{h}_{\text{in}}(i, j)$ derived from $\mathbf{I}$
  2. Use m_{in} to label $\mathbf{U}$ to get the predicted inside strings $\hat{y}_{\text{in}}$
  3. If $\hat{y}_{\text{in}} > \tau_{\text{max}}$, extract c constituents randomly and add it to the set of pseudo-constituents $\mathcal{X}_c$
  4. If $\hat{y}_{\text{in}} < \tau_{\text{min}}$, extract d distituents randomly and add it to the set of pseudo-distituents $\mathcal{X}_d$
  5. $\mathbf{I} = \mathcal{X}_c \cup \mathcal{X}_d$
- Get outside strings for each $\mathbf{I}$; Assign to the set of labeled output sentences $\mathbf{O}$
- Learn outside model m_{out} based on $\boldsymbol{h}_{\text{out}}(i, j)$ derived from $\mathbf{O}$

Output: inside model m_{in}, outside model m_{out}

Figure 2: Our self-training algorithm.

Inputs: $\mathbf{I}$ is the set of labeled inside sentences; $\mathbf{O}$ is the set of labeled outside sentences; $\mathbf{U}$ is a set of unlabeled sentences.

Algorithm: Loop for K iterations:

- Choose c pseudo-constituents and d pseudo-distituents from the most confidently predicted outside strings $\hat{y}_{\text{out}}$ from $\mathbf{U}$ based on τ
- Extract the inside strings $\hat{\mathbf{I}}$ corresponding to the c pseudo-constituents and d pseudo-distituents of outside
- $\mathbf{I} = \mathbf{I} \cup \hat{\mathbf{I}}$
- Train the inside model m_{in} based on $\boldsymbol{h}_{\text{in}}(i, j)$ derived from $\mathbf{I}$
- Choose c pseudo-constituents and d pseudo-distituents from the most confidently predicted inside strings $\hat{y}_{\text{in}}$ from $\mathbf{U}$ based on τ
- Extract the outside strings $\hat{\mathbf{O}}$ corresponding to the c pseudo-constituents and d pseudo-distituents of inside
- $\mathbf{O} = \mathbf{O} \cup \hat{\mathbf{O}}$
- Train the outside model m_{out} based on $\boldsymbol{h}_{\text{out}}(i, j)$ derived from $\mathbf{O}$

Output: Two models m_{in}, m_{out}, that predict the inside and outside scores for unlabeled sentences. We combine these predictions by multiplying together and optionally re-normalizing their class probability scores.

Figure 3: Our co-training algorithm.

assumption by creating inside and outside string representations separately, as we see later in Section 4.

4 Experimental Setup

4.1 Data

We evaluate our methodology on the Penn Treebank (PTB; Marcus et al. 1993) with the standard splits (2-21 for training, 22 for validation, 23 for test). For preprocessing, we keep all punctuation and remove any trailing punctuation. To maintain the unsupervised nature of our experiments, we avoid the common practice of using gold parses of the validation set for either early stopping (Shen et al., 2018, 2019; Drozdov et al., 2019) or hyperparameter tuning (Kim et al., 2019a). Additionally, we experiment on Chinese with version 5.1 of the Chinese Penn Treebank (CTB; Xue et al. 2005) with the same splits as in Chen and Manning (2014), and the Japanese Keyaki Treebank (KTB; Butler et al. 2012). For KTB, we shuffle the corpus and use 80% of the sentences for training, 10% for validation, and 10% for testing.

4.2 Multi-view Learning

In this section, we devise the task of identifying constituents in a sentence by training two models with different views of the data. Ideally, these views complement each other and help each model improve the performance of the other.

4.2.1 Seed Bootstrapping

We treat identifying constituents from unlabeled sentences as a sequence classification task. To generate the *constituent* class, we take the complete sentence `(start:end)`, as a sentence in itself is a constituent, and also the largest among all of its other constituents. To generate the *distituent* class, we take `(start:end-1)`, $\cdots$, `(start:end-6)` slices, where `start` and `end` denote the 0$^{\text{th}}$ and N$^{\text{th}}$ position (sentence length) respectively. We select the distituents in this manner because the longer the sentence, there would be a significantly unlikely chance that the span of the constituents extends till the very end of the sentence. Additionally, we make use of casing-specific information by adding contiguous title-case words while allowing only the apostrophe mark. Since all of the sentences for the constituent class start with capital letters, we identify the most common first word and generate lower-case equivalents of contiguous title-case words, which starts with it to account for bias due to the casing of spans. While we do use a fixed template to perform the seed bootstrapping process, this is part of the inductive bias of the algorithm, and is relatively easy to acquire. In our analysis, we assume the language is already known before and thereby its *structure* (left/right-branching), a form of weak supervision.

For CTB, we follow the exact same process as PTB for preparing the input data for the first-level sequence classifier, but we do not rely on case-specific information and perform no post-processing. Meanwhile, since KTB is a treebank of a strongly left-branching language, we design our modeling approach slightly differently compared to before, although along the same style. To prepare the data for the sequence classifier, we choose the slice `(start:end)` in the sentence to label the *constituent* class, whereas, `(start+1:end)`, $\cdots$, `(start+4:end)` slices are chosen to label the *distituent* class. We also split the sentences on "$\star$" mark and treat the resulting fragmented parts as constituents too. Our training does not depend on the development set with the gold-standard annotated trees since we base the necessary string slicing decision on the feedback from the validation split after the bootstrapping procedure in an iterative fashion (increment/decrement the value of slice counter by 1) until we see a degradation in performance (measured using F_1 score) on the synthetic set of seed constituents and distituents.

4.2.2 Inside Model

We fine-tune the RoBERTa model with a sequence classification layer on top using a cross-entropy loss (see Section A.1 in Appendix for training and hyperparameter details). As we supply input data, the entire pre-trained RoBERTa$_{\text{BASE}}$ model and the additional untrained classification layer is trained on our specific downstream task. To compute $\boldsymbol{h}_{\text{in}}(i, j)$, we run the RoBERTa$_{\text{BASE}}$ model and retrieve the `[CLS]` token representation for the span enclosed between the i^{th} and the j^{th} element. The inside model is evaluated on MCC (Matthews Correlation Coefficient) as well as F_1 because the classes are imbalanced. After fine-tuning, our best inside model achieves 0.28 MCC and 0.42 F_1 on the internal validation set. Finally, we fine-tune the inside model on the unlabeled training sentences that generates an inside score $s_{\text{in}}(i, j)$ for every span. Since our major focus was on PTB, we have listed a few *heuristics* that inject further bias into the algorithm acting as the another form of weak supervision. Moreover, incorporating such rules was not necessary for CTB and KTB as our models showed superior performance without them.

Once we compute the inside score, $s_{\text{in}}(i, j)$, we use the following refinement strategies to prune out false constituents: We treat punctuation characters to mark the boundaries of a span and penalize any span that crosses its demarcated punctuation region (indicated by a span) by assigning a negative penalty of 0.25. Additionally, we delete any constituent if it starts or ends with the most common word succeeding the comma punctuation. Next, we take the most common starting word and check if its accompanying word does not belong to either the stop word or is present in the top 20 most frequent tokens of the PTB training set. We assign the scores of these corresponding spans in the CYK chart cell to the maximum value. Intuitively, from the linguistic definition of constituents, we refrain from bracketing if we identify a contiguous group of rare words (tokens not in the top 1000 most frequent list). These heuristics only contribute to a certain extent in making the parser strong, and should be considered as a standard post-processing step. Overall, we observe 3.8 F_1 improvements in the case of the inside model. We further note that the contribution due to additional heuristics is much less than the combined self-training and co-training gains since their effect becomes insignificant after multiple iterations of the self-training process due to the predictions approximately following the template rules. As described in Figure 2, we perform self-training on the inside model for three iterations.[2]

4.2.3 Outside Model

We extract the outside strings of spans having the inside score satisfying a pre-determined cutoff value. The Constituent-Context Model (Klein and Manning, 2002) use a smoothing ratio of 1:5 (constituents to distituents) for the WSJ-10 section to take into account the skewness of random spans more likely to represent distituents. In the same vein, the values of lower and upper bounds of the threshold are chosen to ensure the distribution of class labels is about 1:10 (with the distituent class being the majority) which is a crude estimate considering much larger sentence lengths in the WSJ-Full section. Moreover, from a linguistic standpoint, we can be certain that the distituents must necessarily outnumber the constituents. For the self-training experiments, we set the thresholds, τ_{min} as 0.0005 and τ_{max} as 0.995. We treat the outside strings satisfying the upper and lower bounds of the threshold as *gold-standard outside* of constituents and distituents respectively. To compute $\boldsymbol{h}_{\text{out}}(i, j)$, we run the RoBERTa$_{\text{BASE}}$ model on left-outside, i.e., $(i-1)^{\text{th}}$ element and right-outside, i.e., $(j+1)^{\text{th}}$ element, along with a `[MASK]` placeholder token separating the two, and extract the `[CLS]` token representation. As done previously, we fine-tune the outside model on the unlabeled training sentences that generates an outside score $s_{\text{out}}(i, j)$ for every span.

4.2.4 Jointly learning with Inside and Outside Models

Once we have the outside model, we run it on the training sentences and choose the outside string that the classifier is highly confident about. We extract their inside strings again using the same bounds of the threshold as done previously and re-train the inside model on the *old highly confident inside strings* along with the *new inside strings obtained from the highly confident outside strings*. Similarly, the same technique can be applied to the

[2] We only use the top 5K inside strings for self-training to cover maximum possible iterations as it is representative of the whole training set in terms of the average sentence length and punctuation marks.

Model	WSJ-Full Mean	WSJ-Full Max	WSJ-10 Mean	WSJ-10 Max
Trivial Baselines:				
Left Branching (LB)	8.7		17.4	
Balanced	18.5			
Right Branching (RB)	39.5		58.5	
Unsupervised Parsing approaches:				
PRPN† (Shen et al., 2018)	37.4	38.1	58.4	–
URNNG⋆ (Kim et al., 2019b)	–	45.4	–	–
ON† (Shen et al., 2019)	47.7	49.4	63.9	–
Tree Transformer†⋆ (Wang et al., 2019)	50.5	52.0	66.2	–
Neural PCFG† (Kim et al., 2019a)	50.8	52.6	64.6	–
DIORA⋆ (Drozdov et al., 2019)	–	58.9	60.5	–
Compound PCFG† (Kim et al., 2019a)	55.2	60.1	70.5	–
S-DIORA†⋆ (Drozdov et al., 2020)	57.6	64.0	71.8	–
Constituency Test⋆ (Cao et al., 2020)	62.8	65.9	68.1	–
Ours⋆ (using inside)	55.9	57.2	64.2	–
Ours⋆ (using inside w/ self-training)	61.4	64.2	66.9	–
Ours⋆ (using inside and outside w/ co-training)	**63.1**	66.8	**73.1**	–
Oracle Binary Trees	84.3		82.1	

Table 1: Unlabeled sentence-level F_1 on the full as well as sentences of length ≤ 10 of the PTB test set without punctuation or unary chains. We evaluate each model using the evaluation script provided by Kim et al. (2019a) and take the baseline numbers of certain models from (Kim et al., 2019a; Cao et al., 2020). † denotes models trained without punctuation and ⋆ denotes models trained on additional data.

Model	CTB Mean	CTB Max
Trivial Baselines:		
Left Branching (LB)	9.7	
Random Trees	15.7	16.0
Right Branching (RB)	20.0	
Unsupervised Parsing approaches:		
PRPN (Shen et al., 2018)	30.4	31.5
ON (Shen et al., 2019)	25.4	25.7
Neural PCFG (Kim et al., 2019a)	25.7	29.5
Compound PCFG (Kim et al., 2019a)	36.0	39.8
Ours (using inside)	37.8	38.4
Ours (using inside w/ self-training)	40.6	41.7
Ours (using inside and outside w/ co-training)	**41.8**	43.3
Oracle Binary Trees	81.1	

Table 2: Unlabeled sentence-level F_1 on the CTB test set. We evaluate each model using the evaluation script provided by Kim et al. (2019a) and take the baseline numbers also from Kim et al. (2019a).

outside model to augment its input data too. We repeat this process twice (Figure 3).

4.3 Evaluation

We report the F_1 score with reference to gold trees in the PTB test set (section 23). Following prior work (Kim et al., 2019a; Shen et al., 2018, 2019; Cao et al., 2020), we remove punctuation and collapse unary chains before evaluation, and calculate F_1 ignoring trivial spans, i.e., single-word spans and whole-sentence spans, and we perform the averaging at sentence-level (macro average) rather than span-level (micro average), which means that we compute F_1 for each sentence and later average over all sentences. We also mention the oracle upper bound, which is the highest possible score with binarized trees since we compare them against non-binarized gold trees according to the convention, as most unsupervised parsing methods output fully binary trees. We additionally use the standard PARSEVAL metric computed by the `evalb` program.[3] Although `evalb` calculates the micro average F_1 score, it differs from our micro average metric in that it will count the whole sentence spans and duplicated spans are calculated and not removed. Following the recommendations put forth by previous work that has done a comprehensive empirical evaluation on this topic (Li et al., 2020b), we report results on both length ≤ 10 as well as all-length test data.

5 Results and Discussion

Table 1 shows the unlabeled F_1 scores for our model compared to existing unsupervised parsers on PTB. The vanilla inside model is in itself competitive and is already in the range of previous best models like DIORA (Drozdov et al., 2019), Compound PCFG (Kim et al., 2019a).[4] See Appendix A.5 to assess our model's performance on unsupervised labeled parsing.

We further evaluate how our method works for languages with different branching types – Chinese (right-branching) and Japanese (left-branching). We use Transformer models for the representations of the spans for both Chinese and Japanese. See Section A.1 in the Appendix for training details. Tables 2 and 3 shows the results for CTB and KTB respectively. Moreover, we do not include a few models chosen previously for PTB during our analysis, as extending those models for CTB or KTB is non-trivial due to several reasons: such as lack of domain-related datasets (as DIORA uses SNLI and MultiNLI for training), and lack of linguistic

[3]https://nlp.cs.nyu.edu/evalb

[4]We do not include the results of Shi et al. (2021) in our analysis because their boost in the performance is contingent on the nature of the supervision data (especially the QA-SRL dataset) rather than on the actual learning process itself. Furthermore, the authors mention that a vast amount of hyperlinks match syntactic constituents, hence restricting the scope for the actual algorithm to derive meaningful trees.

Model	KTB-40 Mean	KTB-40 Max	KTB-10 Mean	KTB-10 Max
Trivial Baselines:				
Left Branching (LB)	29.4		51.6	
Right Branching (RB)	9.8		22.9	
Unsupervised Parsing approaches:				
PRPN (Shen et al., 2018)	27.2	31.8	30.1	33.6
URNNG (Kim et al., 2019b)	10	10.2	22.7	22.7
DIORA (Drozdov et al., 2019)	24.9	26.0	42.3	43.3
DIORA-all (Hong et al., 2020)	36.4	40.0	47.1	48.9
Ours (using inside)	33.7	36.3	53.8	55.9
Ours (using inside w/ self-training)	37.6	39.8	55.5	58.2
Ours (using inside and outside w/ co-training)	**39.2**	41.1	**56.7**	59.1
Upper Bound	76.5		76.6	

Table 3: Evalb F_1 on the full (F_1-all) and length ≤ 10 (F_1-10) sentences of the KTB test set discarding punctuation corresponding to KTB-40 and KTB-10, respectively. We take the baseline numbers of models from Li et al. (2020b). See Table 7 to view the hyperparameters used for `evalb`.

knowledge expertise (not easily cross-lingual transferable notion for designing constituency tests).

Figure 7 in the Appendix shows step-wise qualitative analysis for a sample sentence taken from the PTB training set. See Figures 8 and 9 in Appendix to see the visualization for an example tree at every stage of the pipeline for CTB and KTB respectively. As we can observe from all the example tree outputs, the parser using the inside and outside models after the co-training stage produces fewer crossing brackets than the vanilla inside model.

5.1 Effect of Self-Training

PLMs that possess rich contextualized textual representations can assist parsing when we have a large volume of unlabeled data. For this reason, we might expect that self-training in combination with pre-training adds no extra information to the fine-tuned parser. However, we find that self-training improves the performance of the parser by about 9.8%, demonstrating that self-training provides advantages complementary to the pre-trained contextualized embeddings (see Table 5 in Appendix for a more detailed analysis at different stages).

5.2 Effect of Co-training

The question of how to integrate multi-view information is important. One of the options would be to concatenate both the inside and outside vectors while performing training and inference. With this approach, we see negligible improvement. This corroborates the effectiveness of co-training compared with concatenation: the simple concatenation strategy cannot fully harvest the information corresponding to each view and indeed render the optimization intractable. After co-training, the parser achieves 63.1 F_1 averaged over four runs, outperforming the previous best-published result (see Table 6 in Appendix to view the improvement at each step). Figure 5 in Appendix compares the performance of different models over varying sentence length (see Figure 4 in Appendix to understand the extent to which bootstrapping helps compared to the vanilla inside model).

	PRPN	ON	URNNG	Compound PCFG	S-DIORA	Constituency Test	Our Best Parser
SBAR	50.0	51.2	74.8	56.1	59.2	66.1	**81.7**
NP	59.2	64.5	39.5	74.7	78.0	**79.4**	73.5
VP	46.7	41.0	76.6	41.7	**78.9**	68.2	70.4
PP	57.2	54.4	55.8	68.8	67.1	**86.2**	77.8
ADJP	44.3	38.1	33.9	40.4	49.1	**62.6**	40.9
ADVP	32.8	31.6	50.4	52.5	59.9	63.9	**70.4**

Table 4: Average recall per constituent category (i.e. label recall) in (%). The results of PRPN, ON, URNNG, and Compound PCFG are taken from Kim et al. (2019a), S-DIORA from Drozdov et al. (2020), and Constituency Test from Cao et al. (2020).

5.3 Effect of Distituent Selection

To understand the extent to which the type of the disitituent selection impacts the performance, we assess two settings on the PTB – random and left-branching bias. In the random setting, we select distituents from the slice `(start:`r`)`, where r is a random number generated between `start+1` and `end-1`, both inclusive. This produces 19.3 F_1 for the inside model. Whereas, in the left-branching bias setting, we prepare the seed bootstrapping process as explained in the Section 4.2.1 similar to KTB (a left-branching treebank). This results in 11.2 F_1 score for the inside model. Hence, the manner in which we perform the initial classification has a strong impact on the final tree structures.

5.4 Linguistic Error Analysis

Table 4 shows that our model achieves strong accuracy while predicting all the phrase types except for the Adjective Phrase (ADJP). We list some of the most common mistakes our parse makes and suggest likely explanations for each:

Bracketing inner NP of a definite Noun Phrase. When a definite article is linked with a singular noun, the inner spans need to be shelved, accommodating the larger span with the definite article. E.g.: *the* [*stock market*]

Grouping NP too early overlooking broader context. Due to the way it is trained, the parser aggressively groups rare words in the corpus. Building a better outside model can fix this type of error to a considerable extent. Eg: *Shearson* [*Lehman Hutton*] *Inc.*

Omitting conjunction joining two phrases. It shows poor signs of understanding co-ordination cases in which conjunction is an adjacent sibling of the nodes being shifted, or is the leftmost or rightmost node being shifted. E.g.: *Notable* [*& Quotable*]

Confusing contractions with Possessives. Due to the presence of a lot of contraction phrases like {*they're*, *it's*}, the parser confuses it with that of the Possessive NPs, causing unnecessary splitting. Expanding the contractions can be a good way to correct these systematic errors. E.g.: *the company* [*'s $ 488 million in 1988*]

6 Related Work

Our weakly-supervised parser is comparable in behavior to a fully unsupervised parser as it does not rely on syntactic annotations.

Learning from distant supervision: A related work to ours (Shi et al., 2021) uses answer fragments and webpage hyperlinks to mine syntactic constituents for parsing. Many previous studies depend on punctuation as a strong signal to detect constituent boundaries (Spitkovsky et al., 2013; Parikh et al., 2014).

Incorporating bootstrapping techniques: Co-training (Yarowsky, 1995; Blum and Mitchell, 1998) and self-training (McClosky et al., 2006; Steedman et al., 2003) are bootstrapping methods that attempt to convert a fully unsupervised learning problem to a semi-supervised learning form. More recently, Mohananey et al. (2020); Shi et al. (2020); Steedman et al. (2003) have shown the benefits of using self-training as a standard post-hoc processing step for unsupervised parsing models.

Using Inside-Outside representations constructed with a latent tree chart parser: Drawing inspiration from the inside-outside algorithm (Baker, 1979), DIORA (Drozdov et al., 2019) optimizes an autoencoder objective and computes a vector representation for each node in a tree by combining child representations recursively. To recover from errors and make DIORA more robust to local errors when computing the best parse in the bottom-up chart parsing, an improved variant of DIORA, S-DIORA (Drozdov et al., 2020) achieves it.

Inducing tree structure by introducing an inductive bias to recurrent neural networks: PRPN (Shen et al., 2018) introduces a neural parsing network that has the ability to make differentiable parsing decisions using structured attention mechanism to regulate skip connections in an RNN. ON-LSTM (Shen et al., 2019) enables hidden neurons to learn information by a combination of gating mechanism as well as activation function. In URNNG, Kim et al. (2019b) employs parameterized function over latent trees to handle intractable marginalization and inject strong inductive biases for the unsupervised learning of the recurrent neural network grammar (RNNG) (Dyer et al., 2016). Peng et al. (2019) introduces PaLM that acts as an attention component on top of RNN.

Enhancing PCFGs: Compound PCFG (Kim et al., 2019a) which consists of a Variational Autoencoder (VAE) with a PCFG decoder, found the original PCFG is fully capable of inducing trees if it uses a neural parameterization. Jin et al. (2019) show that the flow-based PCFG induction model is capable of using morphological and semantic information in context embeddings for grammar induction. Zhu et al. (2020) proposes neural L-PCFGs to simultaneously induce both constituents and dependencies.

Concerning PLMs: Tree Transformer (Wang et al., 2019) adds locality constraints to the Transformer encoder's self-attention such that the attention heads resemble a tree structure. More recently, Kim et al. (2020) extract trees from pre-trained transformers.

Refining based on constituency tests: With the help of transformations and RoBERTa model to make grammaticality decisions, (Cao et al., 2020) were able to achieve strong performance for unsupervised parsing.

7 Conclusion

We propose a simple yet effective method which is the first of its kind in achieving performance comparable to the supervised binary tree RNNG model and setting a new SOTA for unsupervised parsing using weak supervision. Our model generalizes to multiple languages of known treebanks. We have done comprehensive linguistic error analysis showing a step-by-step breakdown of the F_1 performance for the inside model versus the inside-outside model with a co-training-based approach. The effectiveness of our multi-view learning strategy is clearly evident in our experiments.

References

J. K. Baker. 1979. Trainable grammars for speech recognition. In *Speech communication papers presented at th 97th Meeting of the Acoustical Society of America*, pages 547–550, Boston, MA.

Avrim Blum and Tom Mitchell. 1998. Combining labeled and unlabeled data with co-training. In *Proceedings of the Eleventh Annual Conference on Computational Learning Theory*, COLT' 98, page 92–100, New York, NY, USA. Association for Computing Machinery.

Rens Bod. 2006. An all-subtrees approach to unsupervised parsing. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 865–872, Sydney, Australia. Association for Computational Linguistics.

Alastair Butler, Tomoko Hotta, Ruriko Otomo, Kei Yoshimoto, Zhen Zhou, and Hong Zhu. 2012. Keyaki treebank: phrase structure with functional information for japanese. In *Proceedings of Text Annotation Workshop*.

Steven Cao, Nikita Kitaev, and Dan Klein. 2020. Unsupervised parsing via constituency tests. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4798–4808, Online. Association for Computational Linguistics.

Danqi Chen and Christopher Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 740–750, Doha, Qatar. Association for Computational Linguistics.

Alexander Clark and Nathanaël Fijalkow. 2020. Consistent unsupervised estimators for anchored PCFGs. *Transactions of the Association for Computational Linguistics*, 8:409–422.

Shay B. Cohen, Karl Stratos, Michael Collins, Dean P. Foster, and Lyle Ungar. 2012. Spectral learning of latent-variable PCFGs. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 223–231, Jeju Island, Korea. Association for Computational Linguistics.

Shay B. Cohen, Karl Stratos, Michael Collins, Dean P. Foster, and Lyle Ungar. 2013. Experiments with spectral learning of latent-variable PCFGs. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 148–157, Atlanta, Georgia. Association for Computational Linguistics.

Shay B. Cohen, Karl Stratos, Michael Collins, Dean P. Foster, and Lyle Ungar. 2014. Spectral learning of latent-variable pcfgs: Algorithms and sample complexity. *Journal of Machine Learning Research*, 15(69):2399–2449.

Andrew Drozdov, Subendhu Rongali, Yi-Pei Chen, Tim O'Gorman, Mohit Iyyer, and Andrew McCallum. 2020. Unsupervised parsing with S-DIORA: Single tree encoding for deep inside-outside recursive autoencoders. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4832–4845, Online. Association for Computational Linguistics.

Andrew Drozdov, Patrick Verga, Yi-Pei Chen, Mohit Iyyer, and Andrew McCallum. 2019. Unsupervised labeled parsing with deep inside-outside recursive autoencoders. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1507–1512, Hong Kong, China. Association for Computational Linguistics.

Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A. Smith. 2016. Recurrent neural network grammars. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 199–209, San Diego, California. Association for Computational Linguistics.

Yoav Goldberg. 2019. Assessing bert's syntactic abilities. *ArXiv*, abs/1901.05287.

Joshua Goodman. 1996. Parsing algorithms and metrics. In *34th Annual Meeting of the Association for Computational Linguistics*, pages 177–183, Santa Cruz, California, USA. Association for Computational Linguistics.

John Hewitt and Christopher D. Manning. 2019. A structural probe for finding syntax in word representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4129–4138, Minneapolis, Minnesota. Association for Computational Linguistics.

Ruyue Hong, Jiong Cai, and Kewei Tu. 2020. Deep inside-outside recursive autoencoder with all-span objective. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 3610–3615, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. 2019. What does BERT learn about the structure of language? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3651–3657, Florence, Italy. Association for Computational Linguistics.

Lifeng Jin, Finale Doshi-Velez, Timothy Miller, Lane Schwartz, and William Schuler. 2019. Unsupervised learning of PCFGs with normalizing flow.

In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2442–2452, Florence, Italy. Association for Computational Linguistics.

Taeuk Kim, Jihun Choi, Daniel Edmiston, and Sang goo Lee. 2020. Are pre-trained language models aware of phrases? simple but strong baselines for grammar induction. In *International Conference on Learning Representations*.

Yoon Kim, Chris Dyer, and Alexander Rush. 2019a. Compound probabilistic context-free grammars for grammar induction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2369–2385, Florence, Italy. Association for Computational Linguistics.

Yoon Kim, Alexander Rush, Lei Yu, Adhiguna Kuncoro, Chris Dyer, and Gábor Melis. 2019b. Unsupervised recurrent neural network grammars. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1105–1117, Minneapolis, Minnesota. Association for Computational Linguistics.

Dan Klein and Christopher D. Manning. 2002. A generative constituent-context model for improved grammar induction. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 128–135, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Dan Klein and Christopher D. Manning. 2005. Natural language grammar induction with a generative constituent-context model. *Pattern Recognit.*, 38:1407–1419.

Bowen Li, Taeuk Kim, Reinald Kim Amplayo, and Frank Keller. 2020a. Heads-up! unsupervised constituency parsing via self-attention heads. In *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*, pages 409–424, Suzhou, China. Association for Computational Linguistics.

Jun Li, Yifan Cao, Jiong Cai, Yong Jiang, and Kewei Tu. 2020b. An empirical comparison of unsupervised constituency parsing methods. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3278–3283, Online. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

David McClosky, Eugene Charniak, and Mark Johnson. 2006. Effective self-training for parsing. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 152–159, New York City, USA. Association for Computational Linguistics.

David McClosky, Eugene Charniak, and Mark Johnson. 2008. When is self-training effective for parsing? In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 561–568, Manchester, UK. Coling 2008 Organizing Committee.

Anhad Mohananey, Katharina Kann, and Samuel R. Bowman. 2020. Self-training for unsupervised parsing with PRPN. In *Proceedings of the 16th International Conference on Parsing Technologies and the IWPT 2020 Shared Task on Parsing into Enhanced Universal Dependencies*, pages 105–110, Online. Association for Computational Linguistics.

Ankur P. Parikh, Shay B. Cohen, and Eric P. Xing. 2014. Spectral unsupervised parsing with additive tree metrics. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1062–1072, Baltimore, Maryland. Association for Computational Linguistics.

Hao Peng, Roy Schwartz, and Noah A. Smith. 2019. PaLM: A hybrid parser and language model. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3644–3651, Hong Kong, China. Association for Computational Linguistics.

Yikang Shen, Zhouhan Lin, Chin wei Huang, and Aaron Courville. 2018. Neural language modeling by jointly learning syntax and lexicon. In *International Conference on Learning Representations*.

Yikang Shen, Shawn Tan, Alessandro Sordoni, and Aaron Courville. 2019. Ordered neurons: Integrating tree structures into recurrent neural networks. In *International Conference on Learning Representations*.

Haoyue Shi, Karen Livescu, and Kevin Gimpel. 2020. On the role of supervision in unsupervised constituency parsing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7611–7621, Online. Association for Computational Linguistics.

Tianze Shi, Ozan İrsoy, Igor Malioutov, and Lillian Lee. 2021. Learning syntax from naturally-occurring bracketings. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2941–2949, Online. Association for Computational Linguistics.

Valentin I. Spitkovsky, Hiyan Alshawi, and Daniel Jurafsky. 2013. Breaking out of local optima with count transforms and model recombination: A study in grammar induction. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1983–1995, Seattle, Washington, USA. Association for Computational Linguistics.

Mark Steedman, Miles Osborne, Anoop Sarkar, Stephen Clark, Rebecca Hwa, Julia Hockenmaier, Paul Ruhlen, Steven Baker, and Jeremiah Crim. 2003. Bootstrapping statistical parsers from small datasets. In *10th Conference of the European Chapter of the Association for Computational Linguistics*, Budapest, Hungary. Association for Computational Linguistics.

Yaushian Wang, Hung-Yi Lee, and Yun-Nung Chen. 2019. Tree transformer: Integrating tree structures into self-attention. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1061–1070, Hong Kong, China. Association for Computational Linguistics.

Naiwen Xue, Fei Xia, Fu-Dong Chiou, and Martha Palmer. 2005. The penn chinese treebank: Phrase structure annotation of a large corpus. *Nat. Lang. Eng.*, 11(2):207–238.

David Yarowsky. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In *33rd Annual Meeting of the Association for Computational Linguistics*, pages 189–196, Cambridge, Massachusetts, USA. Association for Computational Linguistics.

Hao Zhu, Yonatan Bisk, and Graham Neubig. 2020. The return of lexical dependencies: Neural lexicalized PCFGs. *Transactions of the Association for Computational Linguistics*, 8:647–661.

A Appendix

A.1 Training Details

We use the Adam optimizer and, on the bootstrapped dataset, fine-tune `roberta-base` consisting of default 125M trainable parameters with a learning rate $3e-5$, batch size 32, epochs 3, maximum sequence length 128, for all our models. The values were chosen as default based on sequence classification tasks on the GLUE benchmark[5] as mentioned in HuggingFace Transformers.[6] We use a train/validation random split of 80/20 on the bootstrapped dataset which contains 100,000 sentences (50,152 for the distituent class and 49,848 for the constituent class) to monitor the validation loss and perform early stopping. The average sentence length is about 22 tokens. Note that the development set of PTB is kept untouched. We set the patience value at 3. Model checkpointing, as well as logging, is carried out after every 100 steps.

We use a single GPU, Nvidia GeForce RTX 2070 (8GB GDDR5 RAM) to conduct all our experiments. The estimated training time for the inside model is about 0.2h, inside model with self-training (3 iterations) is about 36h, and inside-outside model with co-training (2 iterations) is about 45h. While the inference time for all the models is roughly 0.2h.

For the Chinese monolingual experiment, we use `bert-base-chinese` which is trained on cased Chinese Simplified and Traditional text, and for Japanese monolingual experiment, we use `cl-tohoku/bert-base-japanese` which is trained on Japanese Wikipedia available at https://huggingface.co/models.

Training Data We tried several strategies to augment the *distituent* class for our models, but without concrete gains. Some of those include word deletion (randomly selects tokens in the sentence and replace them by a special token), span deletion (Same as word deletion, but puts more focus on deleting consecutive words), reordering (randomly sample several pairs of span and switch them pairwise) and substitution (sample some words and replace them with synonyms).

[5] https://gluebenchmark.com/

[6] https://huggingface.co/transformers/v2.3.0/examples.html#glue

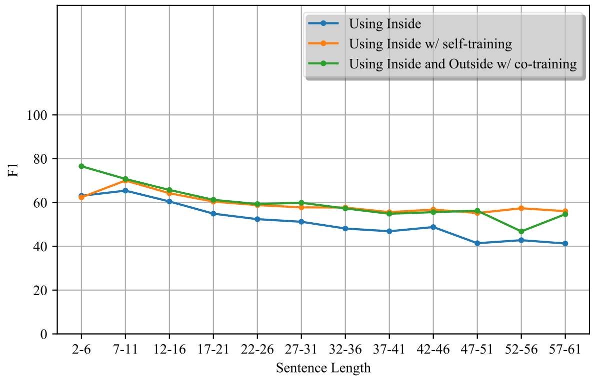

Figure 4: F_1 grouped by sentence length on the PTB test set for different strategies.

Model	#ST-steps			
	0	1	2	3
Inside	55.9	57.7	59.5	61.4

Table 5: Unlabeled sentence-level F_1 on the full PTB test set after applying the iterative Self-Training algorithm on the Inside model.

A.2 Effect of Bootstrapping

As shown in Figure 4, the final model with co-training identifies constituents from shorter sentences (WSJ-10) much more precisely compared to the rest of the models. There is a lower performance in F_1 around sentence length of 50-55 zone, but that improves for longer sentences.[7]

A.3 Stages of Self-Training

Self-training boosts the performance of the inside model by 5.5 F_1 points as shown in Table 5. As can be seen, the effect of the initial set of candidate constituents and distituents on the final performance is 55.9 F_1 which is not insignificant.[8]

A.4 Stages of Co-Training

After co-training, the performance of the inside-outside joint model increases by 1.7 F_1 points as shown in Table 6. Compared to using self-training, one of the reasons the benefit is not significant may be attributed to the fact that the inside vectors (built upon Transformer architecture) inher-

[7] For evaluating PTB and CTB, we use Yoon Kim's script available at https://github.com/harvardnlp/compound-pcfg. Whereas for evaluating KTB, we use Jun Li's script available at https://github.com/i-lijun/UnsupConstParseEval.

[8] For analysis purposes, we use the test set instead of the standard validation set to avoid tuning on the test set based on feedback received from the validation set to keep the nature of our experiments *purely* unsupervised.

Model	#CT-steps		
	0	1	2
Inside and Outside	61.4	62.9	63.1

Table 6: Unlabeled sentence-level F_1 on the full PTB test set after applying the iterative Co-Training algorithm on the joint Inside and Outside model.

ently possesses contextual knowledge due to being trained on a large corpus.

A.5 Unsupervised Labeled Parsing

We explore unsupervised labeled constituency parsing to identify meaningful constituent spans such as Noun Phrases (NP) and Verb Phrases (VP) to see if the parser can extract such labels. Labeled parsing is usually evaluated on whether a span has the correct label. We can effectively induce span labels using the clustering of the learned phrase vectors from the inside and outside strings. When labeling a gold bracket, our method achieves 61.2 F_1 on the full PTB test set and is comparable with the current best model, DIORA. See Figure 6 to view the visualization of induced and linguistic alignment. RoBERTa does not strictly output word-level vectors. Rather, the output are subword vectors which we aggregate with mean-pooling to achieve a word-level representation using `SentenceTransformers`.[9] We use 600 codes while doing the clustering initially, such that we are left with about 25 clusters after the most common label assignment process, i.e., the number of distinct phrase types. The phrase clusters are assigned to {'NP': 7, 'PP': 5, 'WHPP': 3, 'ADVP': 3, 'ADJP': 2, 'S': 2, 'WHADVP': 1, 'UCP': 1, 'VP': 1, 'PRN': 1, 'QP': 1, 'SBAR': 1, 'WHNP': 1, 'CONJP': 1} according to the majority gold labels in that cluster. These 14 assigned phrase types correspond with the 14 most frequent labels. Table 8 lists the induced non-terminal grouped across different clusters and also their correctness in identifying the gold labels. The further course of action would be to have a joint single model that is capable of achieving both bracketing and labeling. Further, these induced labels can function as features for the inside and outside models to achieve even better predictive ability. It also warrants a multi-lingual exploration in this area.

[9] https://github.com/UKPLab/sentence-transformers

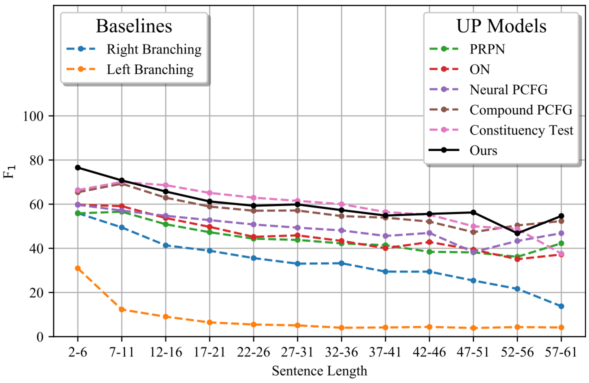

Figure 5: F_1 of different models grouped by sentence length on PTB test set.

A.6 Non-Terminal Label Alignment

Figure 6 shows the alignment between gold and induced labels. We observe that some of the induced non-terminals clearly align to linguistic non-terminals. For instance, S-2 non-terminal has a high resemblance with NP. Similarly, S-8 has a high resemblance with ADVP.

```
DEBUG 0
MAX_ERROR 1
CUTOFF_LEN 10
LABELED 0
DELETE_LABEL_FOR_LENGTH -NONE-
EQ_LABEL ADVP PRT
```

Table 7: The hyperparameters used for `evalb`

.

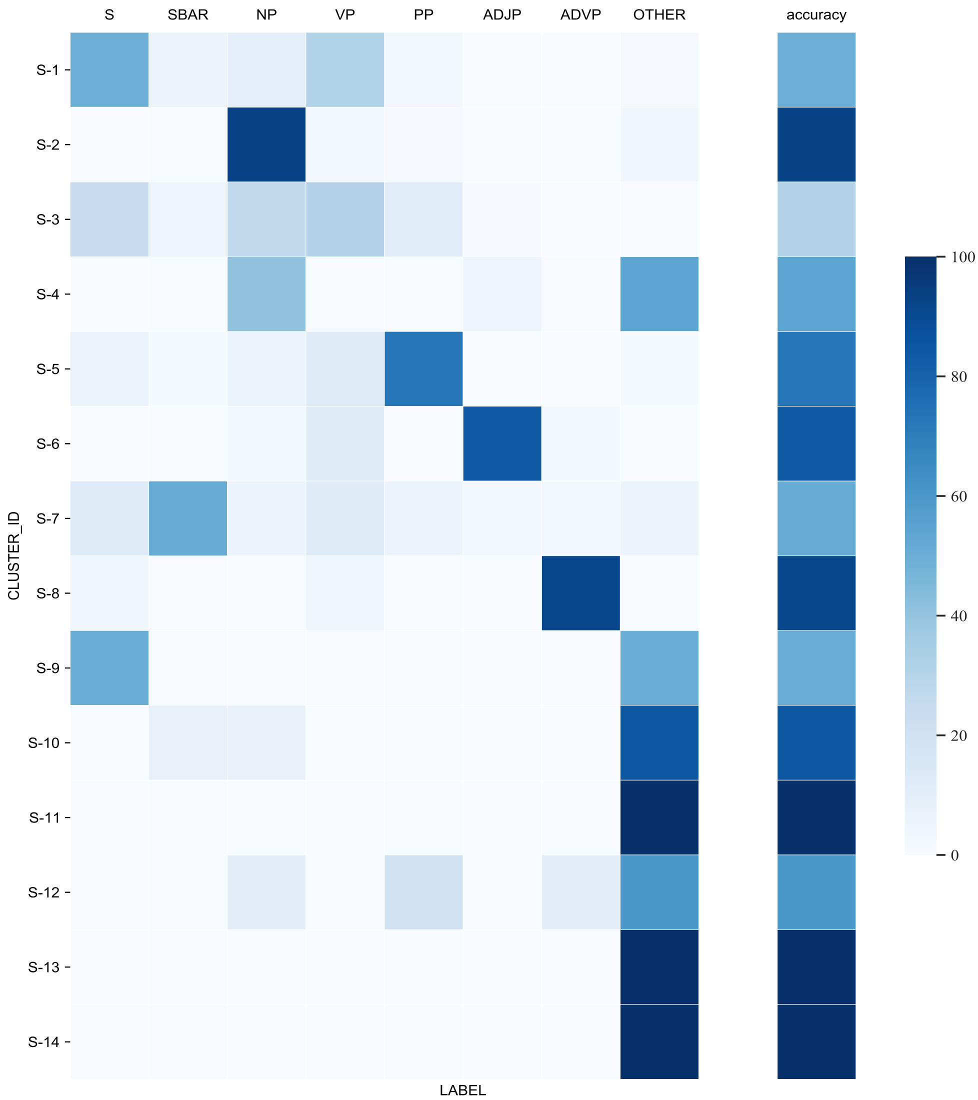

Figure 6: Alignment between induced and gold labels of the top-performing clusters. We cluster the constituent inside vectors derived from the ground truth parse (without labels) using the K-Means algorithm and assign each constituent with the most common label within its cluster. Accuracy is the probability of correctly predicting the most common label.

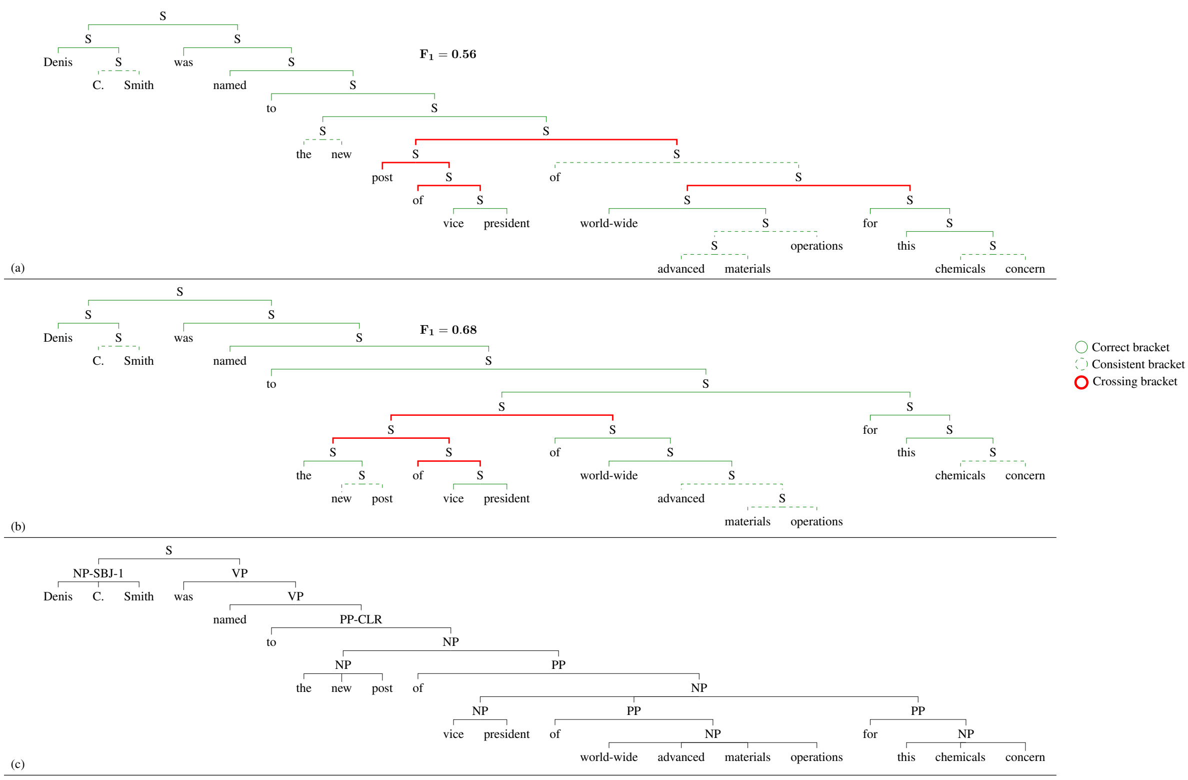

Figure 7: Displays the parse tree output for a sample sentence: (a) Using Inside (b) Using Inside and Outside (c) Gold Tree. After the co-training procedure (b), the parser correctly identifies constituents *"the new post"* and *"of world-wide advanced materials operations"* which were earlier identified as distituents by the inside model (a). It makes two errors due to crossing brackets - namely *"of vice president"*, *"the new post of vice president"*, and *"the new post of vice president of world-wide advanced materials operations"*.

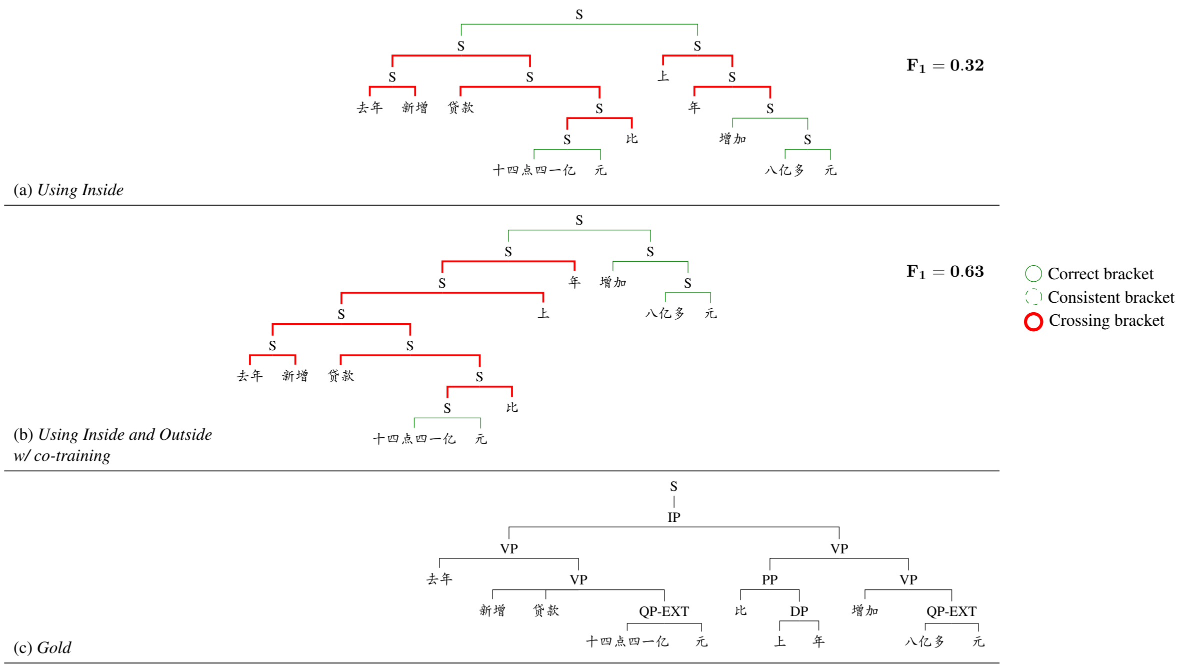

Figure 8: Example tree taken from the CTB training set. After the co-training procedure (b), the parser correctly identifies constituents “十四点四一亿元”, “新增贷款十四点四一亿元”, and “去年新增贷款十四点四一亿元” compared to the previous step using the inside model (a). It only makes 3 errors due to crossing brackets at “贷款十四点四一亿元”, “年增加八亿多元”, and “上年增加八亿多元”.

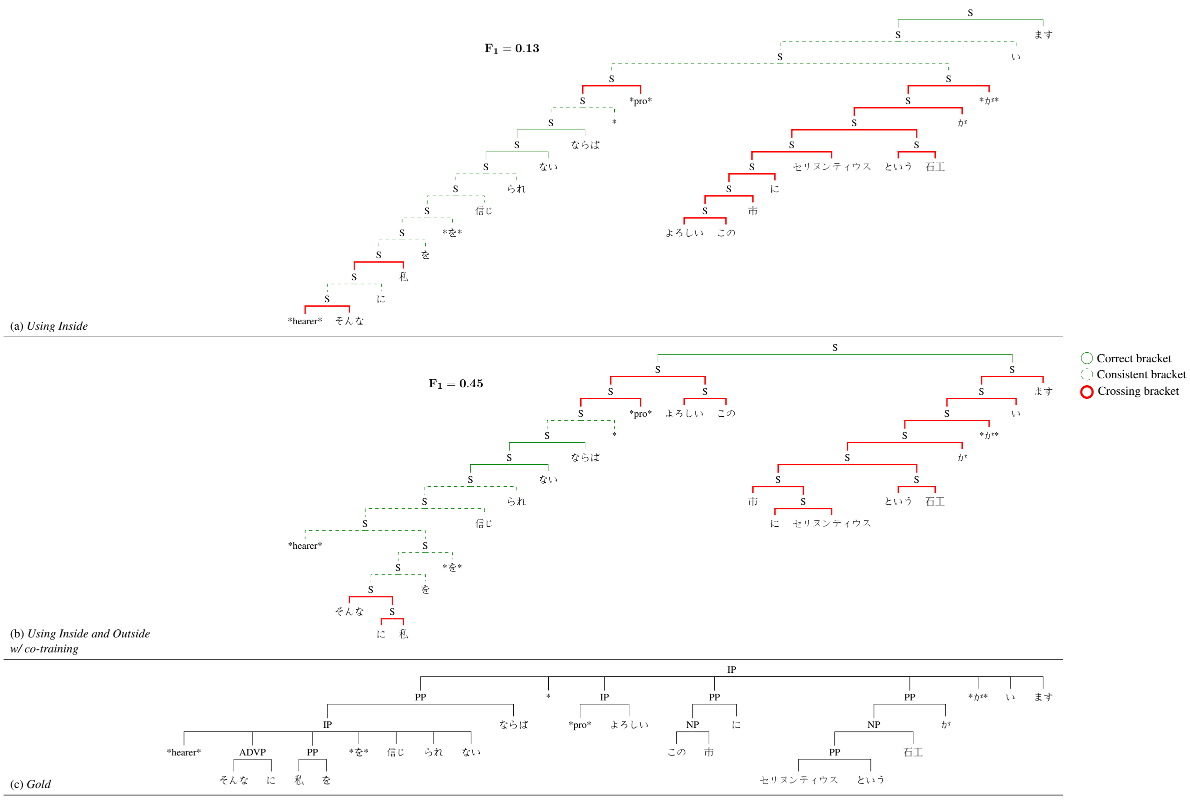

Figure 9: Example tree taken from the KTB training set. After the co-training procedure (b), the parser correctly identifies constituents “そんな に”, “私 を”, “*hearer* そんな に 私 を *を* 信じ られ ない ならば”, “*pro* よろしい”, “この 市”, and “この 市 に”, while incorrectly tagging “セリヌンティウス という 石工 が” as a distituent compared to the previous step using the inside model (a).

Cluster ID	Label	Constituent	Predicted	Status
0	NP	the space shuttle Atlantis	NP	✓
	NP	Once the chief beneficiaries	NP	✓
	PP	in the offing	NP	✗
	PP	in the thrift	NP	✗
	S	the dollar was weak	NP	✗
	SBAR	If the new Cheer sells well	NP	✗
1	ADJP	higher than most anticipated	NP	✗
	NP	more than one billion Canadian dollars 851 mil...	NP	✓
	QP	at least 600 to 700	NP	✗
12	NP	A. Boyd Simpson	NP	✓
	NP	Justice John Harlan	NP	✓
	NP	Robert D. Cardillo	NP	✓
	NP	James D. Awad	NP	✓
	NP	Clark S. Spalsbury Jr	NP	✓
	NP	L.J. Hooker	NP	✓
30	NP	one 's testimony	NP	✓
	NP	the stock market 's plunge Friday	NP	✓
	PP	in the market 's decline	NP	✗
75	ADVP	two years ago	ADVP	✓
	ADVP	two weeks ago	ADVP	✓
	PP	just like two years ago	ADVP	✗
	PP	between now and two years ago	ADVP	✗
310	NP	action on capital gains	VP	✗
	NP	the three airlines being dropped	VP	✗
	NP	news footage of the devastated South Bronx	VP	✗
	NP	the prospect of a fight with GEC for Ferranti	VP	✗
	PP	before declining again trapping more investors	VP	✗
	S	This small Dallas suburb 's got trouble	VP	✗
	S	the earnings picture confuses	VP	✗
	SBAR	it acquired 5 % of the shares in Jaguar PLC	VP	✗
	SBAR	the market is going through another October '87	VP	✗
	VP	may be dubbed Eurodynamics	VP	✓
	VP	resuscitate the protagonist of his 1972 work A...	VP	✓
	VP	said after the 1987 crash	VP	✓
	VP	has a base of 100 set in 1983	VP	✓
514	NP	its two classes of preferred stock	PP	✗
	NP	Oil company refineries	PP	✗
	PP	to depository institutions	PP	✓
	PP	of Remic mortgage securities	PP	✓
	PP	of the preferred-share issue	PP	✓
	PP	in the patent-infringement proceedings	PP	✓
	PP	of mainframe computers	PP	✓
	PP	from mature conventional fields in western Canada	PP	✓
	PP	of its North American vehicle capacity	PP	✓
	VP	have big commodity-chemical operations	PP	✗
533	NP	Bateman Eichler Hill Richards	NP	✓
	NP	KLM Royal Dutch Airlines	NP	✓
	NP	owners Anna and Morris Snezak	NP	✓
	NP	Mehta & Isaly	NP	✓
	PP	at Hambrecht & Quist in San Francisco	NP	✗

Table 8: Investigation of phrase clusters that shows several syntactic properties. Clearly, there are patterns surrounding identification of people/organization names, time-related signals, quantities etc.