# Structural Optimization Ambiguity and Simplicity Bias in Unsupervised Neural Grammar Induction

**Anonymous ACL submission**

## Abstract

Neural parameterization has significantly advanced unsupervised grammar induction. However, training these models with a traditional likelihood loss for all possible parses exacerbates two issues: 1) *structural optimization ambiguity* that arbitrarily selects one among structurally ambiguous optimal grammars despite the specific preference of gold parses, and 2) *structural simplicity bias* that underutilizes available grammar rules. These challenges subject unsupervised neural grammar induction (UNGI) to inevitable prediction errors, high variance, and the necessity for extensive grammars to achieve accurate predictions. This paper tackles these issues, offering a comprehensive analysis of their origins. As a solution, we introduce *sentence-wise parse-focusing* to reduce the parse pool per sentence for loss evaluation, using the structural bias from pre-trained parsers on the same dataset. In unsupervised parsing benchmark tests, our method significantly improves performance while effectively reducing variance and enhancing rule utilization. Our research promotes the learning of more compact, accurate, and consistent explicit grammar, facilitating better interpretability.

## 1 Introduction

Grammar induction (GI) has been intensively studied due to its critical role in understanding, controlling, and utilizing the structural information of internal symbols called constituents in the case of linguistic analysis.

The unsupervised approach to GI is continuously studied for its benefits of avoiding structure data as parsing, which requires enormous human efforts and expertise in generating supervised data. In the literature in this field, neural parameterization to represent grammars showed significant improvement (Kim et al., 2019a) than well-founded probabilistic modeling as Probabilistic Context-Free Grammars (PCFGs) (Klein and Manning, 2002, 2001; Clark, 2001), and advanced models applying issues introduced in UGIs using lexicalization (Zhu et al., 2020; Yang et al., 2021a), or symbol expansion (Yang et al., 2021b, 2022) have been proposed.

However, these Unsupervised Neural Grammar Inductions (UNGIs) lack analysis of the traditional probabilistic learning challenges, particularly about setting the training loss. Some studies, such as (Kim et al., 2019a), have highlighted the difficulty of optimization presented by the complex loss landscape in UGIs (Lari and Young, 1990; Carroll and Charniak, 1992; Klein and Manning, 2001). Subsequent efforts (Yang et al., 2021b, 2022) have alleviated the issues by only enhancing its expressiveness via over-parameterization rather than the cause analysis.

In this paper, we aim to reveal two unhandled problems in UNGIs: 1) *structural optimization ambiguity* to increase the number of the same quality of optima preferring different parse structures, leading to prediction errors by random selection of the structural preference and high variance, and 2) *structural simplicity bias* to induce the neural grammar to use only a few rules and result in inaccurate expression even in its given large capacity. Both properties are caused by training a neural grammar with traditional sentence probability, adding probability mass of all possible parses of the same observation weight as one.

To address the issues, we propose a simple but effective method to inject focusing-bias that accounts for only a few selected parses for training loss in the inside algorithm, which reduces the ambiguity by selecting more distinct parses across sentences and restricts the simplicity bias via separating desired rules with the bias by sentences. We also introduce a method of generating structural bias by parse selection from single or multi-parsers pre-trained on the same training data.

Our method outperforms state-of-the-art implicit and explicit grammar induction models in English

parsing tasks of Penn Treebank (PTB) (Marcus et al., 1993), as well as in ten languages of Chinese Penn Treebank (CTB) (Xue et al., 2005) and SPMRL datasets (Seddah et al., 2014). In-depth analysis shows the reduction of the high variance and the low utilization. We provide the investigation results of various structural biases, recommending heterogeneous multi-parsers.

Our contributions are summarized as follows:

- We raise and clarify *structural optimization ambiguity* and *structural simplicity bias*, which lead to high variance and low rule utilization problems in UNGIs.
- We propose simple yet effective *sentence-wise parse-focusing* with focusing biases from pre-trained parsers on the same training data, which enables stable learning of more compact and accurate explicit grammar.
- Through in-depth empirical analysis, we demonstrate the effectiveness of our approach in reducing the identified causes, investigating various focusing biases, and achieving significant performance improvements compared to state-of-the-art UNGIs.

## 2 Limits on Unsupervised Learning of Neural Grammar

UNGIs based on neural parameterization of grammars using sentence likelihood loss over all possible parses still inherit the limits of probabilistic UGIs: structural optimization ambiguity and structural simplicity bias. For precise analysis, we focus on inducing explicit neural grammars based on PCFGs, mainly Neural PCFGs (N-PCFGs).

**Notations of Probabilistic Context-Free Grammar** In the following analysis, we use the notations below for a PCFG $G = (S, N, P, \Sigma, R, \Pi)$ for root symbol $S$, finite sets of nonterminal $N$, preterminal $P$, terminal words $\Sigma$, production rules $R$, and production rule probabilities $\Pi$. $R$ consists of three types of rules:

$$
\begin{aligned}
&S \rightarrow A && \text{where } A \in N \\
&A \rightarrow \alpha\ \beta && \text{where } \alpha, \beta \in (N \cup P) \\
&B \rightarrow \omega && \text{where } B \in P \text{ and } \omega \in \Sigma
\end{aligned}
$$

Note that the left-hand side (parent) of the rules is distinguished between nonterminal and preterminal categories.

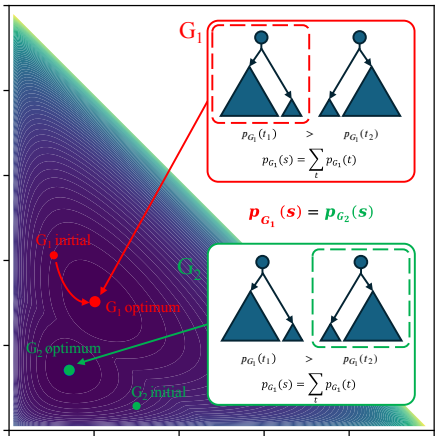



Figure 1: Two different grammars, $G_1$ and $G_2$, have different preferences for parse trees for a given sentence by parse tree probability; however, they have the same sentence probability. The two initial points go to the different optimums close to their own.

### 2.1 Structural Optimization Ambiguity

In this section, we define *structural optimization ambiguity* resulting from grammars of the same training loss for each sample but different preferences of its parse structures. We elaborate on the conditions where the ambiguity appears and its significant intensification.

**Structure Optimization Ambiguity by UGI Loss** We demonstrate the optimization ambiguity and its F1 score variance amplification via analysis from an optimization perspective.

For clear analysis, we define the ambiguity as follows.

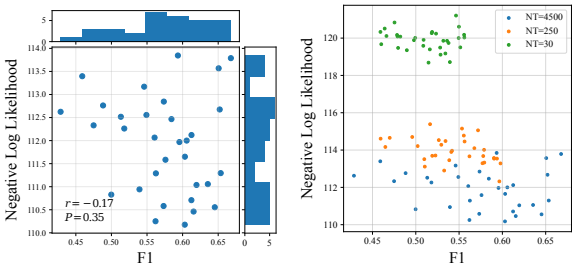
**Definition 1.** Given $G_1$ and $G_2$ that assign different rule probabilities from the same $G$, the grammar $G$ is *structural-optimization-ambiguous* if and only if $p(s|G_1) = p(s|G_2)$ for all sentence $s \in S$ in training data, and there exists $s$ whose best parses respectively derived through $G_1$ and $G_2$ are different.

Figure 1 represents the optimization landscape for an arbitrary sentence, highlighting two optimal points for grammars that are equally probable yet structurally divergent. At the red optimum, the parse tree $t_1$ has a more significant probability than the parse tree $t_2$, whereas, at the blue optimum, the situation is reversed, with $t_2$ having a lower probability than $t_1$. This discrepancy leads to each grammar favoring a different parse tree for the given sentence at each optimal point. Consequently, it becomes unclear which optimum will be chosen

based solely on sentence probabilities from an optimization perspective. Moreover, it is intuitively understood that the reachable optimum may vary with changes in the initial state.

**Amplification of Structural Optimization Ambiguity by Symbol Size** Optimization ambiguity is a common problem in learning on grammar induction, but it is more critical in UGIs due to its properties in training. Detail proof for this effect is in Appendix A.1. In a brief proof sketch, UGIs use all possible binary trees for given sentences, and all nonterminals can appear in any position of parses equally. For clear analysis, we assume that preterminals have all the same probability distribution over words regarded as only one preterminal, which often practically happens in uniform initialization of rule probabilities or assigning only one preterminal to unknown words. Under the properties and conditions, the frequency of a parse with a decision for branching a nonterminal in a parse tree is always equal to that of the other decision (i.e., horizontally flipped parse always exists.). Then, given two grammars based on the same grammar $G$ except rule probabilities, they can have different probabilities for branching on a nonterminal while maintaining the same probabilities for all the other rules. Such grammars still have the same sentence probability in calculation via the inside algorithm, which implies that $G$ has structural optimization ambiguity for the sentence. This problem can be critical because any nonterminal has the ambiguity in UGIs, and therefore, the best parse of a grammar for any sentence causes the ambiguity problem.

**Practical Limit: High Variance and Amplification by Symbol Size** The structural optimization ambiguity leads to a high variance of F1 scores of grammars trained with random seeds. In Figure 2a, we evaluate the correlation of F1 score in test and negative log likelihood in training for 32 models of N-PCFGs after sufficient epochs. The two metrics near the convergence points of the models are almost uncorrelated, showing 0.35 P-value, -0.17 Pearson coefficient, and large change up to more than 20% of F1 scores across models. This low correlation is evidence of the ambiguity, because grammar has no intended and specific bias derived from training loss in selecting one of branching on the ambiguous condition. Therefore, the branching decision relies on randomness in initialization and optimization. Then, the grammars of similar training loss have different parse structures and show




(a) Correlation of F1 score and likelihood (b) Amplification for the variance of F1 by symbol size

Figure 2: Low correlation between F1 score and UGI likelihood; the right illustrates variance amplification by symbol size increasing.

different scores in the test by comparing them with a single gold parse, preferring only one of them. Moreover, Figure 2b shows the amplification of F1 score variance by symbol size increase. This is another evidence of the ambiguity because it appears for any nonterminals in UGIs, and therefore, more nonterminals create more local optima of similar quality to derive different parses, which leads to more variety of different parses compared to a single gold parse.

## 2.2 Structural Simplicity Bias

Another problem is *structural simplicity bias* to induce a grammar to learn a smaller number of production rules by training loss and its amplification in learning for the same parse sets across sentences.

**Rule Simplification by Likelihood Loss** Likelihood loss of grammar induction is well-known to increase the probability mass of only observed samples (Chi and Geman, 1998) and also maximize rule probability for representing correct parses of sentences (Corazza and Satta, 2006).

The likelihood of UGI for a given sentence $s$ (Chi and Geman, 1998) is:

$$L(s) = -\log \sum_{\tau \in \mathrm{T}(s)} \exp \sum_{r \in R(\tau)} f(r|\tau) \log p(r|G)$$

where the exponential term is the probability of a parse tree $\tau$, and $f(r|\tau)$ denotes the frequency of rules used in the tree. The loss reaches its minimum when the exponent, rewritten as $\sum f(A \to \alpha) \log P(A|\alpha) \simeq 0$, and each term within the summation, which is always negative or equal to zero, approaches zero as well.

In supervised grammar induction (SGI), the minimum is a challenging goal because the various gold parses are assigned across sentences. However,

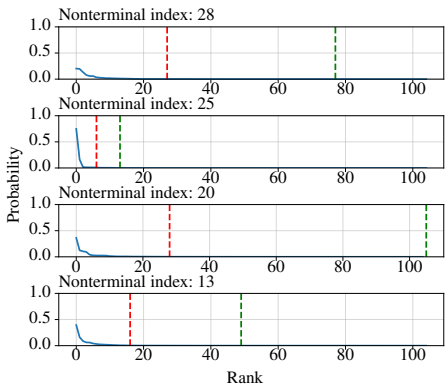



Figure 3: The sorted probabilities in probability distribution for 4 nonterminal symbols. The red and green vertical lines denote the rank of probability that exceeds 95% and 99% portion of the sum of probabilities.

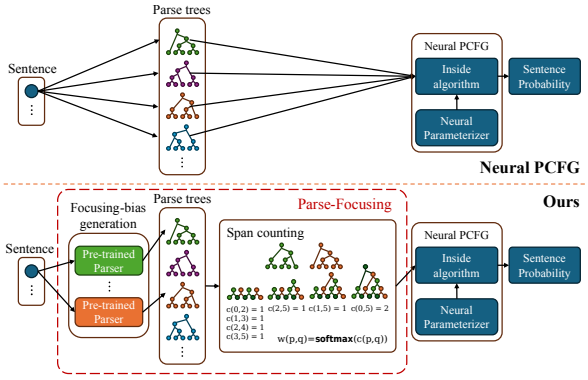



Figure 4: Overview of N-PCFG (above) and Ours (below). The red box shows the parse-focusing method.

UGIs assign the same frequency to all possible parses equally generated for all sentences, at least in the same length. This large number of parses affects the loss scale because the insider term has the cross-entropy form, and its reduction effectively decreases the loss in optimization. Under the single preterminal condition of the structural optimization ambiguity, this behavior is crucial, showing the minimum when it assigns all parse probability mass to a single parse across all sentences. Note that this simplicity bias simplifies the rules of parses not only for each sentence and across sentences.

**Practical Limit: Low Rule Utilization of Huge Grammar** The simplicity bias of UGIs leads to low rule utilization. Figure 3 shows the probability of the top 100 rules among 8100 rules for randomly selected four nonterminals, generated by 30 nonterminals and 60 preterminals of N-PCFG. In the figure, only the top 20 rules (0.2%) have 95% of total probability mass. This represents a significantly small fraction of rules compared to the overall rule count. Such low utilization of rules is a characteristic observed across all rule distributions, as shown in Appendix A.2. The low rule utilization is also reported in parsing tasks (Yang et al., 2021b), but the simplicity bias of UGIs has never been raised as the primary cause.

# 3 Method

## 3.1 Preliminaries

**Motivation** The structural optimization ambiguity is an existing problem of conventional probabilistic GIs (Klein and Manning, 2001), but it has yet to be reported as a critical problem. A possible reason is that they can avoid equal frequency assignment to overall parses, which is the basic condition to causes the ambiguity. For example, the EM algorithm in UGIs re-assigns different rule frequencies at every iteration (Petrov et al., 2006), and SGIs use only gold parse trees while implicitly assigning zero probability to all the incorrect parses. Motivated by this reason, we propose *sentence-wise parse-focusing* to learn from a few parses for each sentence and *structural bias generation* to select the parses from unsupervised multi-parsers pre-trained on the same training data.

**Overview of Parse-Focusing** Figure 4 presents the example application of sentence-wise parse-focusing to N-PCFGS. The distinguished feature compared to N-PCFG is to select a few parses from unsupervised parses pre-trained on the same training data and to derive training loss from them.

## 3.2 Sentence-wise Parse-Focusing

**Learning from Focused Parses** To learn grammar from a few selected parsers, we can simply modify the inside algorithm to count constituent frequency only for the parses. However, this strict bias injection completely excludes unselected parses, resulting in losing potential structures. To alleviate the limits, we use soft-weight derived from softmax over all constituent counts, which maintain the focus on the parses and also leave small weights on potential structures. In detail, the process of applying the parse-focusing is shown in Algorithm 1. For given a finite set of pre-trained parsers $G_p \in \mathcal{G}_p$ and a finite set of sentence $s \in \mathcal{S}$, we represent a finite set of parse trees from parsers as $\mathcal{T}_p$ that derived by $\mathcal{S} \xrightarrow{\mathcal{G}_p}$. We denote frequency for span $(p, q)$ of word $p$ to $q$ as $f(c(p, q)|\tau_{p,i})$ for each parse trees

$\tau_{G_p} \in \mathcal{T}_{G_p}$. Then, the soft-weight is

$$w(p,q|s_i) = \frac{\exp f(c(p,q)|\tau_{G_p}(s_i))}{\sum_{p,q} \exp f(c(p,q)|\tau_{G_p}(s_i))}$$

**Why does it alleviate high variance and low rule utilization issues?** Focusing on only a few parses for each sentence, a model uses a significantly small parse set across the sentences compared to all parses. For example, the reduced parse set may be under ten, but all parses are $O(n^l)$ that is more than a million even for a small grammar using 30 nonterminals and 60 preterminals and two-word sentences. Then, the sentences share the same parses less, which violates the base condition of conflicting rules to have the same frequency in structural optimization ambiguity. In detail, the violation is represented as assigning different weights not to satisfy $\forall I_i(p,q)\ \alpha = 1$, which is a derived condition to cause the ambiguity in the inside algorithm in Equation 6 of Appendix A.1. When weights $w(p,q)$ are applied to spans, the conflicting rule pair also satisfies the weight condition $\frac{w(p,q-1)}{w(p+1,q)} = 1$, which can meet the requirement that $\alpha = 1$. This condition is always fulfilled when equal consideration is given to spans. However, parse-focusing has no specific dynamics to consider them equally, and therefore, the rate is not guaranteed. The sentence-wise separation of parse sets can also enhance the rule utilization because the separation retains the structures of the selected parses, which restricts the simplification across the sentences.

### 3.3 Focusing-Bias Generation

**Bias Generation from Pre-Trained Unsupervised Parsers** Even after applying sentence-wise parse-focusing to reduce the problems, the performance of grammars critically relies on the quality of selected parses. In unsupervised learning, however, finding parses close to gold parse trees without supervision-related information is a challenging issue. As a secondary solution, we use additional parsers pre-trained on the same data with unsupervised learning. Note that our method allows any type of parser as long as it provides parse span information.

**Data-Specific Bias Induction via Heterogeneous Multi-Parsers** A potential approach to generate good parses is to use structurally heterogeneous parsers. According to (Ishii and Miyao, 2023), model-specific inherent bias and data-specific inductive bias appear together in branching bias. Heterogeneous parser models are supposed to have different inherent biases, but the data-specific biases are shared over parsers. In frequency-based learning, the shared structures should have a higher frequency than structures specialized for each parser, and therefore, the data-specific bias dominates structural bias on the selected parses. In Table 1, the structural difference of homogeneous pairs of parsers varying only random seeds with the same parsing method and heterogeneous pairs of parsers using different methods. To directly quantify measuring of the structural heterogeneity, we use IoU scores on parse spans defined as:

$$\frac{\bigcap_{G_p \in \mathcal{G}_p} \mathrm{T}_{G_p}}{\bigcup_{G_p \in \mathcal{G}_p} \mathrm{T}_{G_p}}$$

The overall low IoU scores of all parser pairs is evidence of strong model-inherent bias. The consistently lower IoU scores of heterogeneous parsers imply that their parser combination provides more distinguished model-inherent bias, and therefore, their shared part is more likely to indicate data-specific bias.

| Homogeneous | | Heterogeneous | |
|---|---|---|---|
| Parser Pair | IoU | Parser Pair | IoU |
| (SF, SF) | $56.3_{\pm 7.0}$ | (SF, NBL) | $32.8_{\pm 9.8}$ |
| (FGG, FGG) | $47.0_{\pm 7.3}$ | (SF, FGG) | $38.9_{\pm 5.6}$ |
| (NBL, NBL) | $38.5_{\pm 19.7}$ | (NBL, FGG) | $37.9_{\pm 12.3}$ |

Table 1: IoU scores of homogeneous and heterogeneous parser combinations.

## 4 Experiments

### 4.1 Settings

**Datasets and hyperparameters** We use Penn Treebank (PTB) (Marcus et al., 1994), the English constituency parsing task dataset, to evaluate the parsing performance. Additionally, we use CTB (Xue et al., 2005) and SPMRL dataset (Seddah et al., 2014) for multi-lingual evaluation.

We used all hyperparameters the same as Yang et al. (2022); in the case of analysis for the number of symbols, we follow the ratio 1:2 between nonterminal and preterminal. The details for the dataset and training are in Appendix A.5

**Inference** We basically evaluate grammars with MBR decoding (Yang et al., 2021b). In the other

hand, we use CYK decoding when we need to distinguish each rule in the grammar.

**Base Model** We use the model from Yang et al. (2022), which we refer to as Factor graph grammar-based TN-PCFG (FGG-TNPCFGs), as our base model. Our model is implemented based on open-sourced code of Yang et al. (2022)[1]. There are two reasons for choosing this model: 1) Strict assumption of PCFGs is preserved. In the case of compound PCFGs and lexicalized PCFGs (Zhu et al., 2020; Yang et al., 2021a), they are dependent on words or sentences. Therefore, induced grammar is complex to understand. We avoid it to verify that we get better performance without relaxed assumptions. 2) FGG-TNPCFGs train a vast size of grammar rapidly through efficient usage of memory and computation. This is favorable for getting varied results in various conditions. Also, we notice that FGG-TNPCFGs provide efficient computation, but the fundamental mathematical principle for grammar has stayed the same.

**Selected Parsers for Focusing-Bias** We adopt three parsers that use different algorithms; 1) Structformer (Shen et al., 2021) is a transformer-based and implicit grammar model. 2) NBL-PCFG (Yang et al., 2021a) is a lexicalized explicit grammar model. 3) FGG-TNPCFG (Yang et al., 2022) is the biggest and the state-of-the-art grammar model. Note that all of these parsers are trained with only PTB. The primary trained model by parse-focusing consists of these three parsers, which shows the best performance. Additionally, we use DIORA (Drozdov et al., 2019), which is a self-supervised model utilizing recursive autoencoder architecture, but which trained with additional dataset MultiNLI (Williams et al., 2018) except PTB.

### 4.2 Performance

**Performance in English** We evaluated the performance of state-of-the-art UNGI methods on English PTB data, as shown in Table 2. Ours is significantly outperforming than state-of-the-art performance (FGG-TNPCFG) with only 30 nonterminals and 60 preterminals. In the case of using 4500 nonterminals as FGG-TNPCFG, it again significantly increases the performance. Note the difference between reproduced models and the variance of models. Instead of running only four runs, we reproduce with 32 runs, resulting in lower performance and high variance, especially in FGG-TNPCFG and NBL-PCFG. The significant performance difference between DIORA and DIORA† is due to the differing evaluation criteria. While DIORA utilizes a test set of binarized trees by Stanford CoreNLP, DIORA† employs non-binarized trees. This approach is intended to assess performance accurately within the evaluation environment of Neural PCFG.

[1]https://github.com/sustcsonglin/TN-PCFG

| grammar type | model | S-F1 mean | S-F1 max |
|---|---|---|---|
| implicit grammar | Structformer | 54.0 | - |
| | Structformer† | $52.3_{\pm 2.3}$ | 54.2 |
| | DIORA | 55.7 | 56.8 |
| | DIORA† | $43.6_{\pm 0.9}$ | 44.8 |
| explicit grammar | N-PCFG | $50.9_{\pm 2.3}$ | 52.3 |
| | N-PCFG w/ MBR | $52.3_{\pm 2.3}$ | 55.8 |
| | C-PCFG | $55.4_{\pm 2.2}$ | 59.0 |
| | C-PCFG w/ MBR | $56.3_{\pm 2.1}$ | 60.1 |
| | NL-PCFG | 55.3 | - |
| | NBL-PCFG | 60.4 | - |
| | NBL-PCFG† | $53.3_{\pm 11.5}$ | 62.3 |
| | TN-PCFG | $57.7_{\pm 4.2}$ | 55.6 |
| | FGG-TNPCFG | 64.1 | - |
| | FGG-TNPCFG† | $57.4_{\pm 6.0}$ | 66.8 |
| | Ours (NT=30) | $67.4_{\pm 0.9}$ | 68.4 |
| | Ours (NT=4500) | $69.6_{\pm 0.6}$ | 70.3 |

Table 2: Performance of UNGIs on English trained on PennTreeBank (†: reproduced result). All statistics are from 32 runs with random seeds. Ours trained by randomly and separately generated bias from Structformer†, NBL-PCFG†, and FGG-TNPCFG†.

**Performance in Multilingual** Table 3 shows the performance of explicit grammar-based UNGIs in multilingual parsing on CTB and SPMRL data. In the results, ours significantly outperforms other models in overall languages. A a few languages such as Hungarian, Polish and Swedish show the second or third best but ranks over languages is the best in our model. Note that ours has consistently and significantly low variance compared to all other methods, which implies a better estimator.

### 4.3 In-depth Analysis

**Variance with Random Seeds** We investigated the impact of our method on high variance problem, the main limit of UNGIs, as shown in Table 4. We first prepare random, left-only branching, and right-

| Model | Basque | Chinese | English | French | German | Hebrew | Hungarian | Korean | Polish | Swedish | Mean rank |
|---|---|---|---|---|---|---|---|---|---|---|---|
| N-PCFG* | $35.1_{\pm 2.0}$ | $26.3_{\pm 2.5}$ | $52.3_{\pm 2.3}$ | $45.0_{\pm 2.0}$ | $42.3_{\pm 1.6}$ | $45.7_{\pm 2.2}$ | $43.5_{\pm 1.2}$ | $28.4_{\pm 6.5}$ | $43.2_{\pm 0.8}$ | $17.0_{\pm 9.9}$ | 4.3 |
| C-PCFG* | $36.0_{\pm 1.2}$ | $38.7_{\pm 6.6}$ | $56.3_{\pm 2.1}$ | $45.0_{\pm 1.1}$ | $43.5_{\pm 1.2}$ | $45.2_{\pm 0.5}$ | $\mathbf{44.9_{\pm 1.5}}$ | $30.5_{\pm 4.2}$ | $43.8_{\pm 1.3}$ | $33.0_{\pm 15.4}$ | 3.2 |
| TN-PCFG* | $36.0_{\pm 3.0}$ | $39.2_{\pm 5.0}$ | $57.7_{\pm 4.2}$ | $39.1_{\pm 4.1}$ | $47.1_{\pm 1.7}$ | $39.2_{\pm 10.7}$ | $43.1_{\pm 1.1}$ | $35.4_{\pm 2.8}$ | $\mathbf{48.6_{\pm 3.1}}$ | $\mathbf{40.0_{\pm 4.8}}$ | 3.1 |
| FGG-TNPCFG† | $38.4_{\pm 7.3}$ | $31.0_{\pm 8.4}$ | $59.6_{\pm 7.7}$ | $43.9_{\pm 3.1}$ | $48.0_{\pm 1.4}$ | $46.2_{\pm 4.1}$ | $42.2_{\pm 0.7}$ | $31.5_{\pm 4.0}$ | $41.6_{\pm 4.3}$ | $\mathbf{40.0_{\pm 0.6}}$ | 3.0 |
| Ours | $\mathbf{45.9_{\pm 0.3}}$ | $\mathbf{46.1_{\pm 0.9}}$ | $\mathbf{69.7_{\pm 0.9}}$ | $\mathbf{50.5_{\pm 0.5}}$ | $\mathbf{49.1_{\pm 0.3}}$ | $\mathbf{49.5_{\pm 0.2}}$ | $43.7_{\pm 0.2}$ | $\mathbf{42.1_{\pm 0.3}}$ | $47.9_{\pm 0.3}$ | $33.4_{\pm 1.2}$ | **1.4** |

Table 3: Performance (F1 score) in multilingual parsing on CTB and SPRML datasets. * indicates reported by Yang et al. (2021b), † indicates reproduced using open-sourced code provided by Yang et al. (2022).

| Bias | S-F1 | NLL |
|---|---|---|
| LBranching | $8.9_{\pm 0.0}$ | $116.5_{\pm 2.3}$ |
| RBranching | $39.7_{\pm 0.0}$ | $116.1_{\pm 1.6}$ |
| Random | $34.9_{\pm 1.2}$ | $113.5_{\pm 1.0}$ |
| SF | $58.9_{\pm 0.2}$ | $112.0_{\pm 0.9}$ |
| NBLPCFG | $64.7_{\pm 0.6}$ | $111.5_{\pm 0.7}$ |
| FGG | $65.3_{\pm 0.4}$ | $113.4_{\pm 1.3}$ |
| DIORA | $50.0_{\pm 1.4}$ | $111.6_{\pm 0.3}$ |
| FGGs | $57.4_{\pm 6.0}$ | $111.9_{\pm 1.1}$ |

Table 4: S-F1 and NLL for parse-focused grammar by each bias. LBranching and RBranching mean left-branching and right-branching. Random and pre-trained biases are generated once and used for each run. FGGs denote reproduced performance to compare variance.

only branching parse sets for obtaining the variance change by only parse-focusing. In the results, their variance of F1 scores is significantly reduced, and pre-trained bias cases also show a similar reduction. These reduced variances are significantly lower than FGGs, while their structural loss has a similar bias. The results of implying parse-focusing reduce the variance problem effectively and support the idea that its cause is structural optimization ambiguity under uncorrelated F1 and NLL.

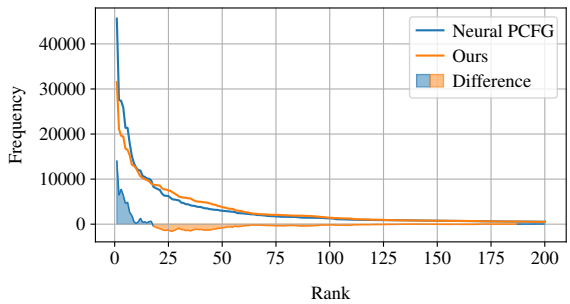



Figure 5: Frequency distribution sorted in descending order for contained rules in parsed train dataset. The colored fill area represents the frequency difference between Neural PCFG and Ours.

**Rule Utilization** Rule utilization is another main limit of UNGIs to verify. Figure 5 shows the sorted frequency of all rules in a grammar. After applying our method, the frequency of the most frequent 17 rules only decreases from 41.8% to 34.4%, and the reduced amount is redistributed to the next top 169 rules. These results imply the effective enhancement of rule utilization.

| Nonterminal | Preterminal | S-F1 | |
|---|---|---|---|
| | | Ours | FGGs |
| 1 | 2 | $44.3_{\pm 0.1}$ | $39.7_{\pm 0.0}$ |
| 5 | 10 | $60.4_{\pm 1.9}$ | $41.6_{\pm 9.6}$ |
| 15 | 30 | $65.5_{\pm 0.7}$ | $43.2_{\pm 0.8}$ |
| 30 | 60 | $67.4_{\pm 0.9}$ | $51.2_{\pm 3.1}$ |
| 250 | 500 | $69.7_{\pm 0.9}$ | $54.3_{\pm 3.9}$ |
| 4500 | 9000 | $69.6_{\pm 0.7}$ | $57.4_{\pm 6.0}$ |

Table 5: The S-F1 score for each number of symbols for FGG-TNPCFGs and induced grammars by parse-focusing.

**Performance and Variance by Symbol Size** Performance change by rule utilization and variance reduction are evaluated in symbol size change in Table 5. The results demonstrate that applying our method to FGG-TNPCFGs significantly enhances performance in the smallest grammar model and even boosts performance in the largest model. Overall variance is significantly reduced. These results imply that our method can stably learn a more compact grammar model while maintaining the same accuracy and a more accurate model while using the same model capacity.

**Quality of Focusing-Bias by Multi-Parser Size** In Figure 6, common structures across the selected parsers are more similar to gold parse structures, which supports the explanation.

**Practical Multi-Parser Combinations** The graph at the bottom of Figure 7 shows that the performance increase among heterogeneous parsers is generally more considerable compared to homogeneous ones, and there is also a more significant variance in performance, which supports the pre-

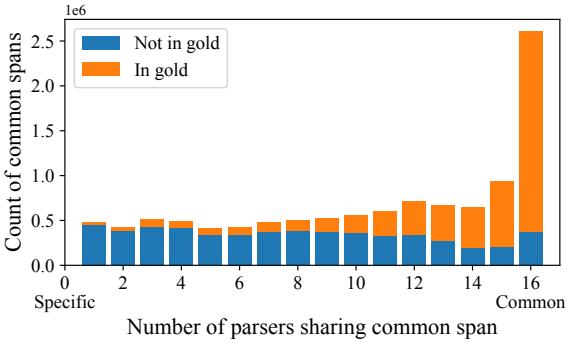



Figure 6: Frequency ($10^6$ scale) of common parse spans in gold parse trees by the number of combined parsers. More parsers show significantly more spans in gold parse spans without supervision.

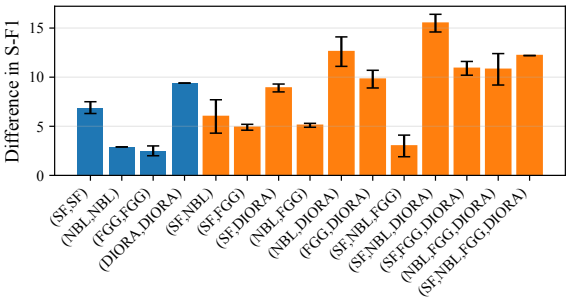



Figure 7: The difference between the mean of S-F1 scores for pre-trained parsers and Multi-parser. The blue bars indicate homogeneous parsers, and the oranges indicate heterogeneous parsers. The pure F1 scores is in Figure 10

vious claim. Consequently, the performance of the parser combinations on parsed-focusing N-PCFG also implies that heterogeneity is a practically helpful strategy.

## 5 Conclusion

In this paper, we tackle the challenges of high variance and low rule utilization in unsupervised neural grammar induction (UNGI). We identify *structural optimization-ambiguity* and *structural simplicity bias* as primary causes, arising from UGI loss on all possible parses across all sentence parses. We introduce a parse-focusing approach that narrows down the parse set per sentence and incorporates biases from parses generated by pre-trained parsers on the same dataset. This method has been shown to effectively mitigate these issues, significantly surpassing state-of-the-art methods in both English and multilingual parsing benchmarks. Our contributions facilitate the consistent learning of more compact, explicit, accurate, and therefore interpretable grammar models in unsupervised learning environment.

**Limitation and Future work** Our work is limited to offline learning due to additional computational costs for pre-training parsers and for the hyper-parameter search of better combinations. In the future, more light and adaptive structural bias generation is required.

## 6 Related Works

**TN-PCFG/FGGs** Yang et al. (2021b) and Yang et al. (2022) verify that overparameterization lead better grammars with many symbols. However, not highlighted, the variance of the F1 score of induced grammar is increased by the number of symbols. In this paper, we show that our approach resolves variance, and the number of symbols is related to performance.

**S-DIORA** Drozdov et al. (2020) refer that structural ambiguity between subtrees can affect to locally greed training, and resolve the problem with single tree encoding by beam search. This shows that choosing a promising tree structure while training can be beneficial.

**Ensemble distillation** Shayegh et al. (2023) introduce a variant of the CYK algorithm that gains refined parse trees through parse trees from the different pre-trained parsers. Our approach differs in searching structure through training in a relaxed condition than a strict algorithm for trees by using soft weight derived from the count of spans. It utilizes the search capability of the model not the algorithm. In addition, it is different that we resolve the optimization problem for loss function.

**Tree-shape uncertainty** The Tree-shape uncertainty referred to in Ishii and Miyao (2023) is the existence of similar grammars that assign different tree structures for a given sentence. We discuss this uncertainty from the optimization perspective. In our perspective, the grammars having equivalent probabilistic optimums may assign different probabilities to parse trees of a given sentence. However, we cannot distinguish these grammars. Consequently, optimization ambiguity differs in discussing that learning becomes unstable due to indistinguishable grammars.

## References


Glenn Carroll and Eugene Charniak. 1992. Two experiments on learning probabilistic dependency grammars from corpora. In *AAAI Workshop on Statistically-Based NLP Techniques*.

Zhiyi Chi and Stuart Geman. 1998. Estimation of probabilistic context-free grammars. *Computational linguistics*, 24(2):299–305.

Alexander Clark. 2001. Unsupervised induction of stochastic context-free grammars using distributional clustering. In *Proceedings of the 2001 Workshop on Computational Natural Language Learning - Volume 7*, ConLL '01, USA. Association for Computational Linguistics.

Anna Corazza and Giorgio Satta. 2006. Cross-entropy and estimation of probabilistic context-free grammars. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 335–342.

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, Pat Verga, Mohit Yadav, Mohit Iyyer, and Andrew McCallum. 2019. Unsupervised latent tree induction with deep inside-outside recursive autoencoders.

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.

Taiga Ishii and Yusuke Miyao. 2023. Tree-shape uncertainty for analyzing the inherent branching bias of unsupervised parsing models. In *Proceedings of the 27th Conference on Computational Natural Language Learning (CoNLL)*, pages 532–547, Singapore. Association for Computational Linguistics.

Yoon Kim, Chris Dyer, and Alexander M 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.

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. 2001. Natural language grammar induction using a constituent-context model. *Advances in neural information processing systems*, 14.

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.

Karim Lari and Steve J Young. 1990. The estimation of stochastic context-free grammars using the inside-outside algorithm. *Computer speech & language*, 4(1):35–56.

Mitch Marcus, Grace Kim, Mary Ann Marcinkiewicz, Robert MacIntyre, Ann Bies, Mark Ferguson, Karen Katz, and Britta Schasberger. 1994. The penn treebank: Annotating predicate argument structure. In *Human Language Technology: Proceedings of a Workshop held at Plainsboro, New Jersey, March 8-11, 1994*.

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.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32.

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 433–440.

Djamé Seddah, Sandra Kübler, and Reut Tsarfaty. 2014. Introducing the SPMRL 2014 shared task on parsing morphologically-rich languages. In *Proceedings of the First Joint Workshop on Statistical Parsing of Morphologically Rich Languages and Syntactic Analysis of Non-Canonical Languages*, pages 103–109, Dublin, Ireland. Dublin City University.

Behzad Shayegh, Yanshuai Cao, Xiaodan Zhu, Jackie C. K. Cheung, and Lili Mou. 2023. Ensemble distillation for unsupervised constituency parsing.

Yikang Shen, Zhouhan Lin, Chin-Wei Huang, and Aaron Courville. 2018. Neural language modeling by jointly learning syntax and lexicon.

Yikang Shen, Shawn Tan, Alessandro Sordoni, and Aaron Courville. 2019. Ordered neurons: Integrating tree structures into recurrent neural networks.

Yikang Shen, Yi Tay, Che Zheng, Dara Bahri, Donald Metzler, and Aaron Courville. 2021. Structformer:

Joint unsupervised induction of dependency and constituency structure from masked language modeling. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 7196–7209.

Yau-Shian Wang, Hung-Yi Lee, and Yun-Nung Chen. 2019. Tree transformer: Integrating tree structures into self-attention.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122, New Orleans, Louisiana. Association for Computational Linguistics.

Nianwen Xue, Fei Xia, Fu-Dong Chiou, and Martha Palmer. 2005. The penn chinese treebank: Phrase structure annotation of a large corpus. *Natural Language Engineering*, 11:207–238.

Songlin Yang, Wei Liu, and Kewei Tu. 2022. Dynamic programming in rank space: Scaling structured inference with low-rank HMMs and PCFGs. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4797–4809, Seattle, United States. Association for Computational Linguistics.

Songlin Yang, Yanpeng Zhao, and Kewei Tu. 2021a. Neural bi-lexicalized PCFG induction. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 2688–2699, Online. Association for Computational Linguistics.

Songlin Yang, Yanpeng Zhao, and Kewei Tu. 2021b. PCFGs can do better: Inducing probabilistic context-free grammars with many symbols. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1487–1498, Online. 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 Proving Structural Optimization Ambiguity with the Inside Algorithm

We incorporate the three stipulated conditions into the conventional inside algorithm as per Equation 1: 1) distinguishing between nonterminals and preterminals, 2) allowing identical positions for all symbols in parse trees, and 3) encompassing all structurally possible binary trees for given sentences. Consequently, we can reformulate the inside algorithm as shown in Equation 2.

In this revised equation, the right-hand side of Equation 1 distinguishes between rules that involve preterminal symbols as children. Owing to conditions 2) and 3), we are able to account for all possible positional cases of symbols, which is reflected in the summation across all symbols within the equation.

We examine a special case where a grammar contains only one preterminal symbol, as detailed in Equation 3, to investigate the potential for structural optimization ambiguity. We recursively derive the conditions leading to ambiguity.

Before beginning the proof, let us consider two grammars, $G_1$ and $G_2$. The relationship between them is defined as follows:

$$\begin{aligned} p_{G_1}(N_i \to N_j T) &\neq p_{G_2}(N_i \to N_j T) \\ p_{G_1}(N_i \to T N_j) &\neq p_{G_2}(N_i \to T N_j) \\ \forall r \in R^c,\ p_{G_1}(r) &= p_{G_2}(r) \end{aligned}$$

Here, $N_i, N_j$ represent nonterminal symbols, $T$ denotes a preterminal symbol, $p_{G_1}(N_i \to N_j T)$ represents the probability of the rule $N_i \to N_j T$ as assigned by $G_1$, and $R^c$ is the complementary set of rules that do not involve $N_i \to N_j T$ and $N_i \to T N_j$.

Initially, we can determine the probability of a span with width $w = 2$ directly:

$$I_{G,i}(p, p+1) = p_G(N_i \to TT) p_G(T \to w_p) p_G(T \to w_{p+1}) \tag{7}$$

For both grammars $G_1$ and $G_2$, the equality $I_{G_1,i}(p, p+1) = I_{G_2,i}(p, p+1)$ consistently holds for all $p$ because the probability of a span with width $w = 2$ does not include the rules $N_i \to N_j T$ and $N_i \to T N_j$.

For spans with width $w = 3$, we refer to Equation 4. We equate the inside probabilities for grammars $G_1$ and $G_2$ as $I_{G_1,i}(p, p{+}2) = I_{G_2,i}(p, p{+}2)$. Noting that only the probabilities of $N_i \to N_j T$ and $N_i \to T N_j$ differ, the other terms cancel out as shown in Equation 5. We can then simplify Equation 5 to Equation 6, given that $I_{G_1,i}(p, p+1) = I_{G_2,i}(p, p+1)$. Consequently:

$$\alpha = \frac{p_{G_1}(T \to w_1) I_{G_1,j}(2, r)}{I_{G_1,j}(1, r-1) p_{G_1}(T \to w_r)} \tag{8}$$

| **Conventional inside algorithm** | |
|---|---|
| $$I_i(1,l) = \sum_i^n \sum_j^n \sum_k^n p(N_i \to N_j N_k) I_j(1,p) I_k(p+1,l)$$ | (1) |
| **Inside algorithm in Unsupervised Neural Grammar Induction** | |
| $$\begin{aligned} I_i(1,l) = &\sum_j^n \sum_k^m p(N_i \to N_j T_k) I_j(1,l-1) p(T_k \to w_l) + p(N_i \to T_k N_j) p(T_k \to w_1) I_j(2,l) \\ &+ \sum_j^n \sum_k^n \sum_{a=2}^{n-2} p(N_i \to N_j N_k) I_j(1,a) I_k(a+1,l) \end{aligned}$$ | (2) |
| **Inside algorithm in UNGI with one preterminal symbol** | |
| $$\begin{aligned} I_i(1,l) = &\sum_j^n p(N_i \to N_j T) I_j(1,l-1) p(T \to w_l) + p(N_i \to T N_j) p(T \to w_1) I_j(2,l) \\ &+ \sum_j^n \sum_k^n \sum_{a=2}^{n-2} p(N_i \to N_j N_k) I_j(1,a) I_k(a+1,l) \end{aligned}$$ | (3) |
| **Inside probability for span width 3** | |
| $$I_{G,i}(1,l) = \sum_j^n p_G(N_i \to N_j T) I_{G,j}(1,l-1) p_G(T \to w_l) + p_G(N_i \to T N_j) p_G(T \to w_1) I_{G,j}(2,l)$$ | (4) |
| **Equivalence between two different grammars** | |
| $$\begin{aligned} &p_{G_1}(N_i \to N_j T) I_{G_1,j}(1,r-1) p_{G_1}(T \to w_r) + p_{G_1}(N_i \to T N_j) p_{G_1}(T \to w_1) I_{G_1,j}(2,r) \\ &= p_{G_2}(N_i \to N_j T) I_{G_2,j}(1,r-1) p_{G_2}(T \to w_r) + p_{G_2}(N_i \to T N_j) p_{G_2}(T \to w_1) I_{G_2,j}(2,r) \end{aligned}$$ | (5) |
| $$\begin{aligned} &p_G(N_i \to N_j T) + p_G(N_i \to T N_j)\alpha \\ &= p_{G'}(N_i \to N_j T) + p_{G'}(N_i \to T N_j)\alpha \end{aligned}$$ | (6) |

Table 6: The equation of inside algorithm and derived equation for proof.

It must be noted that $p_{G_1}(T \rightarrow w_1) I_{G_1,j}(2, r) = I_{G_1,j}(1, r-1) p_{G_1}(T \rightarrow w_r)$ since the preterminal symbol $T$ is unified, leading to $\alpha = 1$.

This process can be applied recursively to spans of any width. As a result, $p_{G_1}(N_i \rightarrow N_j T) + p_{G_1}(N_i \rightarrow T N_j) = p_{G_2}(N_i \rightarrow N_j T) + p_{G_2}(N_i \rightarrow T N_j)$ if and only if $I_{G_1,i}(p, q) = I_{G_2,i}(p, q)$. In simpler terms, the sentence probabilities calculated by $G_1$ and $G_2$ will be identical as long as the sum of probabilities $p_{G_1}(N_i \rightarrow N_j T) + p_{G_1}(N_i \rightarrow T N_j)$ matches that of $G_2$'s.

### A.2 Low Utilization of Rule Probability Distribution

It is essential to note that our analysis pertains to a Grammar consisting of 30 nonterminals and 60 preterminals to assess the extent of low utilization, which includes 8100 rules associated with a single nonterminal symbol parent. Figure 8 displays the sorted probability distributions of binary rules. The red line marks the rank where 95% of the cumulative probabilities are concentrated. On average, this concentration threshold occurs at the 22.7th rank, indicating that around 23 rules are primarily used in parsing sentences. When considering the total rule count, this represents merely about 0.3% of all rules. Therefore, the grammar allocates 95% of its probability mass to just 0.3% of the rules, highlighting an exceedingly low utilization rate for the vast majority of the remaining rules.

### A.3 Algorithm of Proposed Method

Algorithm 1 presents the pseudocode for our entire methodology. We calculate the weights to be applied in the Inside Algorithm using softmax, based on the frequency of each span existing within a sentence as provided by the parse trees from pretrained parsers.

**Algorithm 1** Parse-Focusing Inside algorithm

**function** PARSE-FOCUSING($s \in S, \mathcal{T}_p$)
  $\mathcal{T}_p(s) \leftarrow \mathcal{T}_p$ ▷ Subset of parse trees
  $C(s) \leftarrow$ COUNTSPANS($\tau_{p_i}, |s|$)
  $W(s) \leftarrow \dfrac{\exp f(c(p,q)|\tau_{G_p}(s_i))}{\sum_{p,q} \exp f(c(p,q)|\tau_{G_p}(s_i))}$
  **for** $w \leftarrow 2$ to $|s|$ **do**
    **for** $n \leftarrow 0$ to $|s|$ **do**
      $I_i(n, n+w) \leftarrow \sum_{j,k} \sum_m p(i,j,k) I_j(n,m) I_k(m+1, n+w)$
      $I_i(n, n+w) \leftarrow w_s(n, n+w) I_i(n, n+w)$
  **return** $p(s)$
**function** COUNTSPANS($\tau_{p_i}, |s|$)
  Initialize $c(i,j) = 0$, for all $0 \leq i, j \leq |s|$
  **for** $i \leftarrow 0$ to $|s|$ **do**
    **for** $j \leftarrow i+2$ to $|s|$ **do**
      **for each** $\tau_{p_i} \in \mathrm{T}_p$ **do**
        **if** span $(i,j)$ in $\tau_{p_i}$ **then**
          $c(i,j) \leftarrow c(i,j) + 1$
  **return** $\mathcal{C}(s)$

### A.4 Background

With the dramatic growth of Neural Networks, various approaches have been developed to integrate the concepts of Neural Networks and Grammars. (Shen et al., 2018, 2019) utilized tree structures within their models, hierarchically learning implicit structural information from sentences through tree structures. Meanwhile, Wang et al. (2019); Shen et al. (2021) aimed to leverage structural information within transformers, inferring structural information from sentences and using it to mask the attention matrix to control relationships between words. Notably, Shen et al. (2021) proposed a method for simultaneously learning constituency and dependency parsing.

There have been studies applying structural concepts, either by mimicking grammars using recurrent neural networks Dyer et al. (2016); Kim et al. (2019b) or by combining the inside algorithm with the concept of autoencoders (Drozdov et al., 2019).

In particular, Kim et al. (2019a) proposed neural parameterization, assigning rule probabilities using neural networks from grammar component embeddings. Kim et al. (2019a) used explicit grammatical structures and the inside algorithm to induce grammars, enabling traditional grammatical approaches. This led to the emergence of lexicalized Neural PCFGs (Zhu et al., 2020; Yang et al., 2021a), leveraging lexicons. However, the inside algorithm still demanded high computational complexity, limiting the expansion of grammar sizes.

Yang et al. (2021b, 2022) addressed the high computational complexity of the inside algorithm by utilizing tensor decomposition and FGGs, allowing for the learning of larger grammars. This approach, solving issues through the number of parameters, aligns with the methods of Petrov et al. (2006).

### A.5 Experiment Details

**Dataset Detail** We follow a standard setup and preprocessing for the PTB[2], utilizing sections 02-21 as the training set, section 22 as the validation set, and section 23 as the test set while excluding punctuation and trivial constituents. The most

[2]The license of PTB is LDC User Agreement for Non-Members. https://catalog.ldc.upenn.edu/LDC99T42

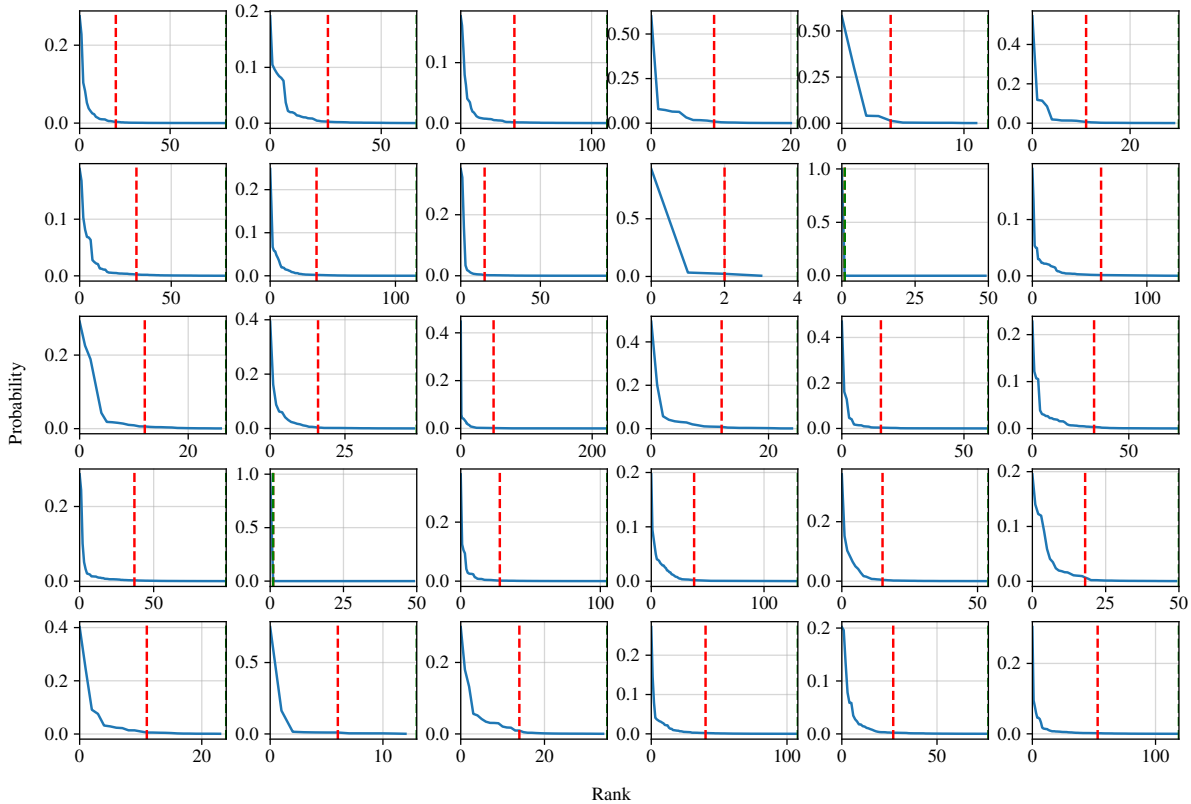



Figure 8: The probability distribution of $A \rightarrow \alpha\beta$ sorted in descending order. It show all figure for each nonterminal of grammar. Figure 3 is parts of this.

frequent 10,000 words were selected as the vocabulary, and the remaining words were treated as <unk>. To process the data, we followed the pipeline used by models employing base models and parsers as outlined in (Kim et al., 2019a; Yang et al., 2022; Shen et al., 2021; Yang et al., 2021a; Drozdov et al., 2019). We also use CTB[3] and Basque, French, German, Hebrew, Hungarian, Korean, Polish, Swedish dataset in SPMRL[4].

**Implementation Detail** To implement our methodology on top of the base model, FGG-TNPCFGs, we utilized PyTorch version 2.2 (Paszke et al., 2019). For the smooth processing and analysis of tree structures, we employed NLTK[5].

**Training Detail** The hyperparameter details are identical to those found in Yang et al. (2022). The ratio of nonterminal to preterminal symbols is primarily set at 1:2, and to analyze performance based on the number of symbols, experiments were conducted using a minimum of 1 to a maximum of 4500 nonterminal symbols. Training was carried out up to 10 epochs, with early stopping applied based on valid likelihood. For optimization, the Adam optimizer was utilized with a learning rate of $2e^{-3}$, $\beta_1 = 0.75$, and $\beta_2 = 0.999$. The model's hyperparameters were set to a rank size of 1000, a symbol embedding size of 256, and a word embedding size of 200. The same hyperparameters were used for all languages without specific tuning. All experiments were performed using an NVIDIA RTX 2080ti, and 32 experiments were conducted to verify variance for both the original FGG-TNPCFGs and Ours, as shown in Table 2, while the rest of the experiments were carried out four times each. Each run takes approximately 3 hours.

### A.6 Total difference of rule frequencies

Figure 9 illustrates the whole frequency distribution of Figure 5 on the log scale. We already verify that the rules having the highest rank occupy almost a portion of frequencies. We find that existing models allocate more frequencies on the tails of the frequency distribution.

[3]The license of CTB is LDC User Agreement for Non-Members. https://catalog.ldc.upenn.edu/LDC2005T01.

[4]The license of SPMRL is Creative Commons Attribution 4.0 International License. https://www.spmrl.org/spmrl2013-sharedtask.html

[5]https://www.nltk.org/

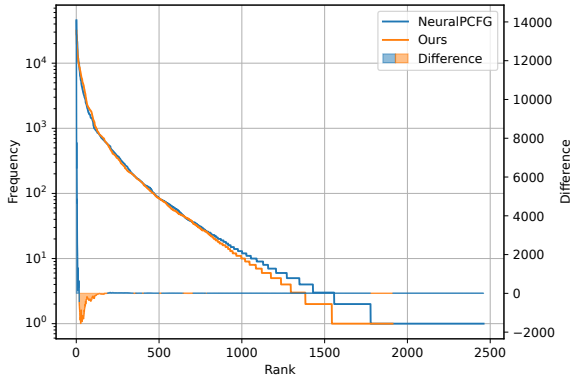



Figure 9: Log scale frequency distribution expanded for whole used rules for Figure 5.

### A.7 Focusing-Bias of Single Parser

To validate the learning of various types of focusing bias in the Neural PCFG, we assessed the Intersection over Union (IoU) scores between the pre-trained single parsers and our parse-focusing Neural PCFG, as shown in Table 7. The results in Table 7 indicate that our parse-focusing Neural PCFGs can replicate the tree structures generated by the pre-trained parsers. They show the highest congruence with FGG-TNPCFGs and the least congruence with structformer. Despite this, the structformer, which has the lowest IoU score, still surpasses the highest IoU score listed in Table 1. This indirectly indicates the extent of similarity between the parse trees derived from the pre-trained parsers and those obtained from the Parse-focusing Neural PCFG.

| Bias | S-F1 | | IoU |
|---|---|---|---|
| | Pre-trained | Injected | |
| SF | $52.32_{\pm 2.25}$ | $56.35_{\pm 2.03}$ | $57.18_{\pm 2.81}$ |
| NBL | $52.11_{\pm 15.32}$ | $53.04_{\pm 16.03}$ | $61.37_{\pm 2.39}$ |
| FGGs | $59.65_{\pm 7.73}$ | $58.67_{\pm 7.15}$ | $74.49_{\pm 6.67}$ |

Table 7: S-F1 score of Pre-trained parsers, Parse-focusing N-PCFG, and IoU score between their results.

### A.8 Numerical Result for Figure 10

In Table 8, the S-F1 and IoU scores on combinations of homogeneous parsers and heterogeneous parsers. However, heterogeneous parsers show significantly lower IoU scores overall combinations than homogeneous parsers. A lower IoU score signifies that the two grammars make different decisions for more spans, allowing for handling a wider variety of structural information in parse-focusing.

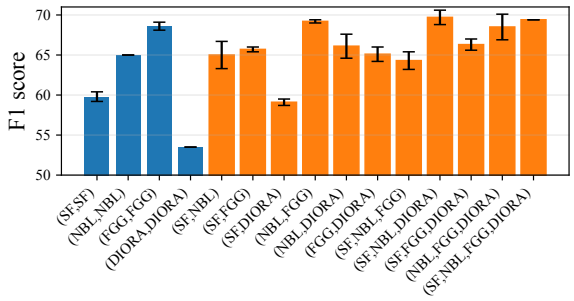



Figure 10: S-F1 scores of the combination of homogeneous and heterogeneous parsers.

The model can select the most advantageous structural information from the given options, which opens up opportunities for exploration, potentially leading to higher performance.

| Parser Pair | S-F1 | IoU |
|---|---|---|
| Homogeneous (same method, different seeds) | | |
| (SF,SF) | $59.8_{\pm 0.6}$ | 50.6 |
| (NBL,NBL) | $65.0_{\pm 0.2}$ | 57.1 |
| (FGG,FGG) | $68.6_{\pm 0.5}$ | 40.8 |
| (DIORA,DIORA) | $53.5_{\pm 0.7}$ | 43.9 |
| Heterogeneous (different method, different seeds) | | |
| (SF,NBL) | $65.0_{\pm 1.7}$ | 40.3 |
| (SF,FGG) | $65.7_{\pm 0.3}$ | 39.6 |
| (SF,DIORA) | $59.1_{\pm 0.4}$ | 36.2 |
| (NBL,FGG) | $69.2_{\pm 0.2}$ | 40.6 |
| (NBL,DIORA) | $66.1_{\pm 1.5}$ | 32.9 |
| (FGG,DIORA) | $65.1_{\pm 0.9}$ | 34.3 |
| (SF,NBL,DIORA) | $64.3_{\pm 1.1}$ | 21.0 |
| (SF,FGG,NBL) | $69.7_{\pm 0.9}$ | 24.7 |
| (SF,FGG,DIORA) | $66.3_{\pm 0.7}$ | 21.7 |
| (FGG,NBL,DIORA) | $68.5_{\pm 1.6}$ | 20.7 |

Table 8: IoU scores of homogeneous and heterogeneous parser combinations. S-F1 is the test performance of ours with the parser combination on PTB. FGG denotes FGG-TNPCFG.