Tree-shape Uncertainty for Analyzing the Inherent Branching Bias of Unsupervised Parsing Models

Taiga Ishii and **Yusuke Miyao**
The University of Tokyo
{taigarana,yusuke}@is.s.u-tokyo.ac.jp

Abstract

This paper presents the formalization of tree-shape uncertainty that enables us to analyze the inherent branching bias of unsupervised parsing models using raw texts alone. Previous work analyzed the branching bias of unsupervised parsing models by comparing the outputs of trained parsers with gold syntactic trees. However, such approaches do not consider the fact that texts can be generated by different grammars with different syntactic trees, possibly failing to clearly separate the inherent bias of the model and the bias in train data learned by the model. To this end, we formulate tree-shape uncertainty and derive sufficient conditions that can be used for creating texts that are expected to contain no biased information on branching. In the experiment, we show that training parsers on such unbiased texts can effectively detect the branching bias of existing unsupervised parsing models. Such bias may depend only on the algorithm, or it may depend on seemingly unrelated dataset statistics such as sequence length and vocabulary size.

1 Introduction

In unsupervised parsing, a model receives raw texts as training data and produces trained parsers. The *branching bias* of an unsupervised parsing model is the bias in the branching direction of tree structures it is likely to learn (Li et al., 2020a), where branching direction refers to whether trees grow deeper on the left or right side. Such a bias is important in applications; for example, a model with a right-branching bias is likely to be more accurate for a right-branching language such as English but less accurate for a left-branching language like Japanese. A theoretical bias analysis was done by Dyer et al. (2019), but their method is specific to certain models, such as PRPN (Shen et al., 2018), and not general in nature. Instead, the branching bias of a model is observed by empirically comparing the performances of trained parsers for languages with different gold tree branching directions, e.g., different natural-language treebanks such as English and Japanese (Li et al., 2020b), original and reversed treebanks (Li et al., 2020a), and synthetic languages (Jin et al., 2018).

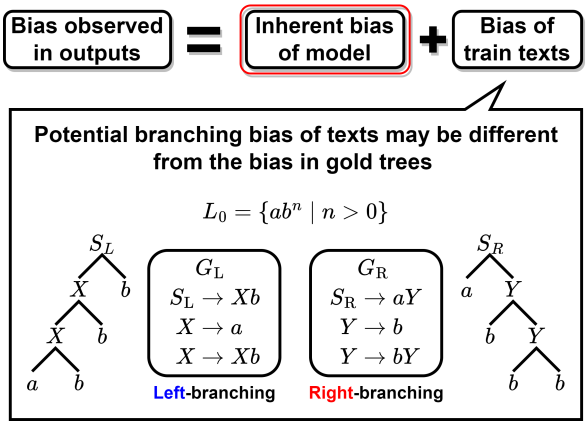


Figure 1: An illustration of the critical problem in branching bias analysis of unsupervised parsing models

However, performance comparison based on gold syntactic trees is theoretically incomplete as bias analysis. In principle, to analyze the inherent bias of a model in a model-agnostic way, we need to examine the bias in the output tree structures of trained parsers. Yet, to make this procedure theoretically valid, it needs to be clarified what information the train texts can provide to the models regarding branching direction because the bias observable in the parser outputs are two folds: the inductive bias inherent in the model and the bias in the train data that can be learned by a parser. We call the latter the *potential branching bias of texts*. The problem with previous work is that the bias in the branching directions of gold trees may not be equal to the potential branching bias of the texts. For example, Jin et al. (2018) assumed the language $L_0 \equiv \{ab^n \mid n > 0\}$ is left-branching because it can be generated by a left-branching grammar (Figure 1: G_{L}); similarly, they assumed

Proceedings of the 27th Conference on Computational Natural Language Learning (CoNLL), pages 532–547
December 6–7, 2023.

$L_1 \equiv \{a^n b \mid n > 0\}$ is right-branching. They train parsers on each language and compare the likelihoods to show that models have no bias. However, in fact, L_0 can also be generated by a right-branching grammar (Figure 1: G_{R}), and L_1 by a left-branching grammar. Therefore, it is not trivial to claim that the texts drawn from L_0 provide left-branching information to models and, hence, that the performance gap between L_0 and L_1 reveals the models' inherent branching bias. This points out that assumptions about gold trees may lead to a misestimate of the potential branching bias of texts and, thus, the branching bias of the models.

How can we avoid such problems? One solution is to use texts that contain no potential branching bias. Similar to how Kharitonov and Chaabouni (2020) study the inductive bias of sequence-to-sequence models by training them on non-informative data, if we train parsers on unbiased texts, we can directly observe the model's inherent branching bias as the bias in the outputs of trained parsers without the need to compare with gold trees. In other words, parsers must decide the branching directions based solely on the bias inherited from the model if the train texts give no information about the branching directions. In this paper, we first introduce the concept of *tree-shape uncertainty*, which formulates the property of certain texts that can be produced by syntactic trees of different shapes. Then, we revisit the work by Li et al. (2020a) and derive sufficient conditions for tree-shape uncertainty to construct texts with no potential branching bias.

In the experiments, we analyze the inherent branching bias of models by constructing unbiased texts based on natural language corpora. To examine the biases in the parser outputs, we extend existing tree imbalance measures (Fischer et al., 2021) to capture branching directions. The experiments on popular unsupervised parsing models DIORA (Drozdov et al., 2019), PRPN (Shen et al., 2018), and URNNG (Kim et al., 2019b) demonstrate that our method can effectively detect the different branching biases of these models. We also find that the bias of URNNG may be sensitive to seemingly unrelated dataset statistics such as sequence length and vocabulary size.

2 Measuring Branching Direction

In this section, we describe the measures for branching direction. We denote by $\mathcal{T}$ the set of all, possibly non-binary, unlabeled trees and formalize the requirements for a branching measure as follows.

Definition 1. We call a function $B : \mathcal{T} \to [-1, 1]$ *branching measure* if it meets the following requirements:

1. $B(t) = -1$ and 1 when t is a complete left and right-branching tree, respectively.
2. $B(t) = 0$ when t is a complete n-ary tree.
3. $B(t) = -B(t^{-1})$ for any t and its flip t^{-1}.

Here, flipping a tree is defined as reversing the order of the child subtrees for all internal nodes.

In the field of phylogeny, a number of tree-shape metrics have been proposed based on leaf depths (Kirkpatrick and Slatkin, 1993; Coronado et al., 2020; Fischer, 2021), number of leaves (Heard, 1992; Mooers and Heard, 1997), and number of inner vertices satisfying certain conditions (Rogers, 1996; Kersting and Fischer, 2021; Norström et al., 2012). However, these metrics are mostly about the (im)balance of tree structures and do not address branching directions. For this reason, we pick up and modify three metrics, namely, the corrected Colles index (Heard, 1992), the equal weights Colles index (Mooers and Heard, 1997), and the Rogers J index (Rogers, 1996). Furthermore, since these three metrics are only defined for binary trees, we naively generalize them to apply to non-binary trees. As can be seen in Table 1, all the modified branching measures used in this paper satisfy the requirements in Definition 1 .

2.1 Corrected Colles Index

First, the Colles index (Colless, 1982; Shao and Sokal, 1990) is an imbalance measure for binary trees defined as the sum of the absolute difference in the number of leaves of left and right subtrees of each inner vertex: $\sum_{v \in V_t^{\mathrm{in}}} ||t_{v_0}| - |t_{v_1}||$. Here, V_t^{in} is the set of inner vertices of t, v_0, v_1 are the left and right children of v, t_v is the subtree rooted at v, and $|t|$ denotes the number of leaves of a tree t. One problem with the Colles index is that its maximum value is dependent on tree size, making it impossible to compare the values between trees with different numbers of leaves. The corrected Colles index (Heard, 1992) remedies such a problem by normalizing the Colles index with its maximum value of $\frac{(|t|-1)(|t|-2)}{2}$.

Since the original formula for the Colles index is defined only for binary trees, we cannot extend

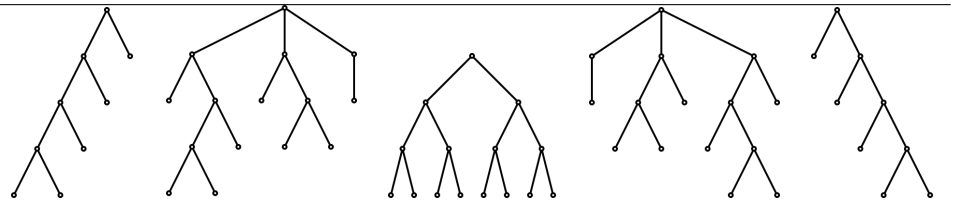
	t_0	t_1	t_2	t_1^{-1}	t_0^{-1}
B					
$\mathrm{CC}^{\pm}$	-1	$-\frac{1}{21}$	0	$\frac{1}{21}$	1
$\mathrm{EWC}^{\pm}$	-1	$\frac{1}{12}$	0	$-\frac{1}{12}$	1
$\mathrm{RJ}^{\pm}$	-1	0	0	0	1

Table 1: An example of trees and corresponding branching scores for $\mathrm{CC}^{\pm}$, $\mathrm{EWC}^{\pm}$, and $\mathrm{RJ}^{\pm}$

it to a branching measure for n-ary trees by simply removing the absolute operator. Hence, we modify the corrected Colles index by substituting the absolute difference with a weighted relative difference. Let $|v|$ be the number of children of a vertex v; we consider the following weights for the child node v_i indexed from the left:

$$w_v(i) \equiv \begin{cases} g(i - (\frac{|v|-1}{2})) \cdot \frac{1}{\lfloor |v|/2 \rfloor} & |v| > 1, \\ 0 & \text{otherwise}, \end{cases}$$

where $g(x) \equiv \mathrm{sign}(x) \cdot \lceil |x| \rceil$ is a rounding toward infinity. For example, when $|v| = 5$, the weights are $(-1, -\frac{1}{2}, 0, \frac{1}{2}, 1)$; note that unary nodes always assign weight 0 to their children. The weighted relative difference h is then calculated as follows:

$$h(v) \equiv \sum_{i=0}^{|v|-1} w_v(i) \cdot |t_{v_i}|.$$

Finally, the modified version of the corrected Colles index is described in the following:

$$\mathrm{CC}^{\pm}(t) \equiv \frac{2}{(|t|-1)(|t|-2)} \cdot \sum_{v \in V_t^{\mathrm{in}}} h(v).$$

2.2 Equal Weights Colles Index

One of the characteristics of the (corrected) Colles index is that branches closer to the root are evaluated more heavily than those closer to the leaves. Instead of simply summing up the absolute difference in the number of leaves for the inner vertices, the equal weights Colles index (Mooers and Heard, 1997) sums up the normalized values to treat the inner vertices equally.

We denote by $\mathrm{EWC}^{\pm}$ the extended version of the equal weights Colles index:

$$\mathrm{EWC}^{\pm}(t) \equiv \frac{1}{|t|-2} \cdot \sum_{v \in V_t^{\mathrm{in}}: |t_v| > 2} \frac{h(v)}{|t_v|-2}.$$

2.3 Rogers J Index

As Zhang et al. (2022) determined whether a phrase is left-branching or not by simply comparing the sizes of the left and right subtrees, we can also employ such phrase-level binary decisions to a whole sentence. The Rogers J index (Rogers, 1996) computes the degree of tree imbalance simply by counting the number of inner vertices that are not balanced. Compared to the Colles index-based metrics above, such count-based metrics can evaluate tree imbalance more coarsely.

In this paper, we normalize the Rogers J index by dividing it by its maximal value of $|t|-2$ and extend it to capture the branching direction as follows:[1]

$$\mathrm{RJ}^{\pm}(t) \equiv \frac{1}{|t|-2} \cdot \sum_{v \in V_t^{\mathrm{in}}} \mathrm{sign}(h(v)).$$

3 Formalizing Texts with No Potential Branching Bias

Linguistically, branching directions in natural language syntactic trees reflect the relative position of the head and modifier in a phrase. For example, Figure 2 shows that the syntactic tree of the same phrase is right-branching in English and left-branching in Japanese. In this way, we can observe the branching bias in natural language as a bias in the shape of syntactic trees if they are given. But what if we do not assume any underlying syntactic

[1] An imbalance metric staircase-ness (Norström et al., 2012) divides the Rogers J index by $|t|-1$, but obviously, it does not assign 1 to completely right/left-branching trees.

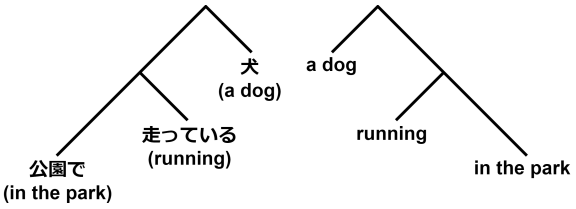


Figure 2: An example of syntactic trees of the same phrase in Japanese (left) and English (right)

trees? In fact, for some texts, whether they belong to left or right-branching language cannot be decided on their own. To formalize such a textual property, we use probabilistic context-free grammar (PCFG), a well-established and widely used grammar formalism in natural language processing (Johnson et al., 2007; Liang et al., 2007; Wang and Blunsom, 2013; Kim et al., 2019a). In section 3.2, we first define uncertainty in general tree shapes and then specialize it to branching direction.

3.1 Probabilistic Context-free Grammar

As is clear from the definition below, PCFG itself does not have any preference for left or right-branching structures. At this point, PCFG is a suitable tool for formalizing the potential branching bias of texts without assuming gold syntactic trees.

A probabilistic context-free grammar (PCFG) G is defined as a tuple $(\Sigma, N^G, S^G, R^G, \pi^G)$ consisting of a finite set of terminal symbols Σ, a finite set of nonterminal symbols N^G, a start symbol $S^G \in N^G$, a finite set of production rules R^G, and the rule probabilities $\pi^G \equiv \{\pi^G_r \in (0,1] \mid r \in R^G\}$. We consider production rules in a general form:

$$A \to \beta \quad A \in N^G, \beta \in (\Sigma \cup N^G)^+.$$

Besides, the rule probabilities must sum up to 1 for each nonterminal: $\forall A \in N^G. \sum_{r:A\to*\in R^G}(\pi^G_r) = 1$. Given a grammar G, the joint probability of a string $s \in \Sigma^*$ and an unlabeled tree $t \in \mathcal{T}$ is calculated by $p_G(s,t) \equiv \sum_{\mathfrak{t}\in\mathcal{T}_G(s,t)} \prod_{r\in R^G_\mathfrak{t}}(\pi^G_r)$, where $\mathcal{T}_G(s,t)$ is the set of derivation trees of s with the shape t, and $R^G_\mathfrak{t}$ is the enumeration of the rules used in the derivation tree $\mathfrak{t}$. We denote by $\mathcal{G}$ the set of all PCFGs with terminals Σ.

3.2 Tree-shape Uncertainty

In order to formalize texts that have no potential branching bias, we first abstract branching direction and define uncertainty in general tree shapes.

Given a text corpus, i.e., a finite multiset of texts, D, its corresponding tree structure assignment $T : \Sigma^* \to \mathcal{T}$, and a PCFG G, we denote by $G \xrightarrow[P]{T} D$ that, the corpus D is generated by G with T with probability $P \in [0,1]$, that is, $P = \prod_{s\in D} p_G(s, T(s))$.[2]

Definition 2. A text corpus D is said to be *tree-shape uncertain* with respect to $\mathcal{N}_\mathcal{G}$ and $\overline{\mathcal{N}}_\mathcal{T}$ if the following proposition holds:

$$\forall G.\ \forall T.\ G \xrightarrow[P]{T} D \implies$$
$$\exists G' \in \mathcal{N}_\mathcal{G}(G).\ \exists T' \in \overline{\mathcal{N}}_\mathcal{T}(T, D).\ G' \xrightarrow[P]{T'} D,$$

where $\mathcal{N}_\mathcal{G}$ and $\overline{\mathcal{N}}_\mathcal{T}$ define the neighborhood and non-neighborhood of grammar and tree structure assignment, respectively.

Intuitively, tree-shape uncertainty illustrates that no matter what grammar and syntactic tree underlie the texts, there is always a grammar that is similar in terms of $\mathcal{N}_\mathcal{G}$ but generates the same texts differently in terms of tree shapes. Here, $\overline{\mathcal{N}}_\mathcal{T}(T, D)$ can be considered as generally defining the "differently shaped trees" for T and D. Note that tree-shape uncertainty is different from *ambiguity in grammar* (Hopcroft et al., 2001). Whereas the latter concerns the ambiguity of derivation trees within a single grammar, the former is rather broad and allows trees from different grammars.

Now, we define $\overline{\mathcal{N}}_\mathcal{T}$ specific to the branching direction so that tree-shape uncertainty describes the uncertainty in the branching directions of texts.

Definition 3. A tree non-neighborhood $\overline{\mathcal{N}}_\mathcal{T}$ is called a *branching non-neighborhood* if there is a branching measure B and

$$\overline{\mathcal{N}}_\mathcal{T}(T, D) = \left\{ T' \middle| \sum_{s\in D} \frac{B(T(s))}{|D|} = -\sum_{s\in D} \frac{B(T'(s))}{|D|} \right\}.$$

We denote such non-neighborhood by $\overline{\mathcal{N}}^B_\mathcal{T}$.

For example, if T assigns right-branching trees to D, then any $T' \in \overline{\mathcal{N}}^B_\mathcal{T}(T, D)$ has the opposite branching directions on average, specifically left-branching, measured by average B.[3]

[2] Note that there is no restriction on T for $s \in \Sigma^* \setminus D$.

[3] Note that, by Definition 3 , T is included in $\overline{\mathcal{N}}^B_\mathcal{T}(T, D)$ when $\sum_{s\in D} B(T(s)) = 0$. Nevertheless, this won't be problematic since the T in Definition 2 is universally quantified, and the underlying branching direction of D must be uncertain with respect to T s.t. $\sum_{s\in D} B(T(s)) \neq 0$. Developing more sophisticated non-neighborhoods is left for future work.

Definition 4. We call a grammar neighborhood $\mathcal{N}_\mathcal{G}$ *complexity neighborhood* if there is a grammar complexity measure $C : \mathcal{G} \to \mathbb{R}_{\geq 0}$ and

$$\mathcal{N}_\mathcal{G}(G) = \{G' \mid C(G) = C(G')\}$$

We denote such grammar neighborhood by $\mathcal{N}^C_\mathcal{G}$.

Moreover, we call $\mathcal{N}^C_\mathcal{G}$

- *production-flip invariant* iff $\forall G.\forall \widetilde{G} \in \mathcal{F}(G).\ C(G) = C(\widetilde{G})$,
- *symbol-mapping invariant* iff $\forall G.\forall \phi \in \mathrm{Aut}(\Sigma).\ C(G) = C(G_\phi)$,

where $\mathcal{F}(G)$ is the set of grammars $\widetilde{G}$ that can be obtained by flipping the right-hand side of some production rules of G, $\mathrm{Aut}(\Sigma)$ is the set of automorphisms on Σ, and G_ϕ denotes the grammar whose terminal symbols are remapped by ϕ.[4]

For instance, commonly used grammar complexity measures such as the number of nonterminals, the number of production rules, etc. (Gruska, 1971; Ginsburg and Lynch, 1976), all induce production-flip and symbol-mapping invariant complexity neighborhoods.

4 Sufficient Conditions for Unbiased Texts

In this section, we revisit the approach taken by Li et al. (2020a) and extend it to derive sufficient conditions for tree-shape uncertainty, which is useful for branching bias analysis.

Li et al. (2020a) analyzed the branching bias of the syntactic trees extracted from pre-trained language models such as BERT (Devlin et al., 2019; Liu et al., 2019) and GPT2 (Radford et al., 2019). To do this, they trained language model m on natural language treebank corpus D and m' on the reversed corpus D^{-1}. Let $\mathrm{F1}(m, T_{\text{gold}})$ be the F1 score of m for the gold syntactic trees T_{gold} of D; they measured the branching bias by the difference in accuracy $\mathrm{F1}(m, T_{\text{gold}}) - \mathrm{F1}(m', {T_{\text{gold}}}^{-1})$, based on the intuition that reversing the text of a right-branching language yields the text of a left-branching language. However, such bias evaluation is highly dependent on the choice of gold trees. It becomes problematic when D can be generated by trees with different shapes from T_{gold}, potentially over/underestimating the bias of the models.

The problem above is that the potential branching bias of texts is not necessarily the same as that of gold trees. On the other hand, if we can train parsers on texts that contain no potential branching bias, we can directly observe the inherent branching bias of unsupervised parsing models without worrying about the choice of gold trees. To construct such unbiased texts, we can extend the intuition of Li et al. (2020a). That is, reversing given texts yields texts of completely opposite underlying branching directions, and if the reversed texts coincide with the original, the text should not contain left-right branching direction bias. For instance, we can combine a corpus Z and its flip, i.e., $D \equiv Z \cup Z^{-1}$. If a grammar G generates D, then the flipped grammar G^{-1} generates D^{-1} with the same probability but with flipped derivations, which leads to completely the opposite branching directions for the same texts D $(= D^{-1})$. The following theorem further generalizes such construction by allowing re-mappings of terminal symbols.

Theorem 1. *The following holds for any text corpus D:*

$$\exists \phi \in \mathrm{Aut}(\Sigma).\ \exists Z \subset D.\ D = \bigcup_{k=0}^{|\phi|-1} f^k(Z) \wedge \exists n \in \mathbb{N}_{>0}.|\phi| = 2n$$

$\Longrightarrow$

D is tree-shape uncertain with respect to any $\overline{\mathcal{N}}^B_\mathcal{T}$ and any $\mathcal{N}^C_\mathcal{G}$ that is production-flip and symbol-mapping invariant,

where $|\phi|$ denotes the order of ϕ, and $f(Z) \equiv \phi(Z^{-1})$ flips each sequence in Z and remaps each symbol by ϕ.[5]

Proof. First, we show that if $D = \phi(D^{-1})$, then D is tree-shape uncertain with respect to $\mathcal{N}^C_\mathcal{G}$ and $\overline{\mathcal{N}}^B_\mathcal{T}$. Take any G and T. For any sequence s and tree t, it can be seen that

$$\begin{aligned}
p_G(s,t) &= \sum_{\mathfrak{t} \in \mathcal{T}_G(s,t)} \prod_{r \in R^G_\mathfrak{t}} (\pi^G_r) \\
&= \sum_{\mathfrak{t} \in \mathcal{T}_G(s,t)} \prod_{r \in R^G_\mathfrak{t}} (\pi^{G^{-1}_\phi}_{\phi(r^{-1})}) \\
&= \sum_{\phi(\mathfrak{t}^{-1}) \in \mathcal{T}_{G^{-1}_\phi}(\phi(s^{-1}),t^{-1})} \prod_{\phi(r^{-1}) \in R^{G^{-1}_\phi}_{\phi(\mathfrak{t}^{-1})}} (\pi^{G^{-1}_\phi}_{\phi(r^{-1})}) \\
&= p_{G^{-1}_\phi}(\phi(s^{-1}), t^{-1})
\end{aligned}$$

[4] An automorphism ϕ on Σ is a bijective function $\Sigma \to \Sigma$.

[5] $|\phi|$ is defined as the smallest $k \in \mathbb{N}_{>0}$ s.t. $\phi^k = 1$.

holds.[6][7] Thus, $G_\phi^{-1} \xrightarrow[P]{T_\phi^{-1}} D$ follows since we have

$$\begin{aligned}
&\prod_{\phi(s^{-1})\in D} p_{G_\phi^{-1}}(\phi(s^{-1}), T_\phi^{-1}(\phi(s^{-1})))\\
&\quad= \prod_{\phi(s^{-1})\in D} p_G(s, T(s))\\
&\quad= \prod_{s\in D} p_G(s, T(s))\\
&\quad= P,
\end{aligned}$$

where we denote by $T_\phi^{-1} : \phi(s^{-1}) \mapsto T(s)^{-1} \in \mathcal{T}$ the flipped tree structure assignment.[8] The following equations show $T_\phi^{-1} \in \overline{\mathcal{N}}_{\mathcal{T}}^{B}(T, D)$; that is, T_ϕ^{-1} is a member of the branching non-neighborhood:

$$\begin{aligned}
\sum_{\phi(s^{-1})\in D} \frac{B(T_\phi^{-1}(\phi(s^{-1})))}{|D|} &= \sum_{s\in D} \frac{B(T(s)^{-1})}{|D|}\\
&= -\sum_{s\in D} \frac{B(T(s))}{|D|}.
\end{aligned}$$

Since $\mathcal{N}_{\mathcal{G}}^{C}$ is production-flip and symbol-mapping invariant, we also have $G_\phi^{-1} \in \mathcal{N}_{\mathcal{G}}^{C}(G)$, which leads to the tree-shape uncertainty of D.

Therefore, to prove the theorem, it suffices to show $D = \phi(D^{-1})$:

$$\begin{aligned}
\phi(D^{-1}) &= \bigcup_{k=0}^{|\phi|-1} f^{k+1}(Z)\\
&= \phi^{|\phi|}(Z^{-|\phi|}) \cup \bigcup_{k=1}^{|\phi|-1} f^k(Z)\\
&= Z \cup \bigcup_{k=1}^{|\phi|-1} f^k(Z) = D,
\end{aligned}$$

since $|\phi|$ is the order of ϕ, and we have, by definition, $\phi^{|\phi|} = 1$. We also have $Z^{-|\phi|} = Z^{-2n} = Z$ because a string does not change when flipped an even number of times. □

[6]The second line follows from the fact that the rule probabilities do not change by flipping and remapping terminal symbols on the right-hand side of the rules: $\pi^{G}_{A\to\beta} = \pi^{G^{-1}}_{A\to\beta^{-1}} = \pi^{G_\phi^{-1}}_{A\to\phi(\beta^{-1})}$ $(\equiv \pi^{G_\phi^{-1}}_{\phi(r^{-1})})$.

[7]The third line follows because $\phi(\cdot^{-1})$ induces one-to-one mappings $\mathcal{T}_G(s,t) \to \mathcal{T}_{G_\phi^{-1}}(\phi(s^{-1}), t^{-1})$ and $R^{G}_{\mathrm{t}} \to R^{G_\phi^{-1}}_{\phi(\mathrm{t}^{-1})}$.

[8]Note that since $\phi(\cdot^{-1})$ induces a one-to-one mapping on Σ^*, T_ϕ^{-1} is well-defined. Besides, we always have $\prod_{\phi(s^{-1})\in D} * = \prod_{\phi(s^{-1})\in\phi(D^{-1})} * = \prod_{s\in D} *$ as we assume $D = \phi(D^{-1})$. Similar equations also hold for $\sum$.

The intuition behind considering automorphisms on terminal symbols is that when we use one-hot encoding or randomly initialize word embedding, exchanging the embedding between different words does not make any essential difference to models.[9] In our formalization, such intuition is formulated as the symbol-mapping invariance of the grammar neighborhood. Thus, Theorem 1 can be interpreted as indicating that we can construct a text corpus D with any base texts Z and vocabulary automorphism ϕ ($|\phi| = 2n$) such that the underlying branching direction cannot be identified from the texts alone when using one-hot encoding or randomly initialized word embedding.

Consideration for Natural Language One might wonder if natural language texts satisfy the sufficient conditions introduced in Theorem 1 . The answer, in short, is probably no. This can be seen from a very simple example. Consider texts $D = \{x = \text{“S V”}, y = \text{“S V O”}\}$. If there is an automorphism ϕ such that $D = \phi(D^{-1})$, then it is clear that for x, S must map to V, but for y, S must map to O, contradicting that ϕ is an automorphism. However, Theorem 1 shows only sufficient conditions, and whether natural language texts are tree-shape uncertain or not is an open problem. Moreover, it is still difficult to design toy languages that are not tree-shape uncertain. This is because, to prove that given texts are not tree-shape uncertain, we must construct a grammar and show that any similarly complex grammar does not generate the texts with the same probability or with differently shaped syntactic trees, which is not trivial.

5 Experimental Settings

To analyze the inherent branching bias of unsupervised parses, we utilize Theorem 1 . More concretely, we create $D_\phi^Z \equiv \bigcup_{k=0}^{|\phi|-1} f^k(Z)$ based on some base texts Z and morphism ϕ; we then use D_ϕ^Z to train unsupervised parsers.[10]

5.1 Datasets

5.1.1 Base Text Z

As the choice of base text Z, we use natural language corpora. In order to verify whether D_ϕ^Z can be used for branching bias analysis regardless of

[9]This is not the case for pre-trained word embedding.

[10]The URL for the codes: https://github.com/mynlp/tree-shape-uncertainty

the underlying branching direction of Z, we follow Li et al. (2020b) and use the English Penn Treebank (PTB) (Marcus et al., 1993) as a corpus for right-branching language and the Japanese Keyaki Treebank (KTB) (Butler et al., 2012) for left-branching language. For preprocessing, we use the same script used in Li et al. (2020b).[11] For PTB, sections 02-21 are used as train split, 22 as dev split, and 23 as test split. The KTB corpus is randomly split into train, dev, and test in an 8-1-1 ratio. Then, punctuation is removed, and the sentences in train and dev splits are filtered by the maximum length of 10 and 40.[12] In addition, numbers are replaced by the "<num>" token, and words that occur only once are replaced by the "<unk>" token.[13] We denote by PTB10, PTB40, KTB10, and KTB40 the preprocessed datasets for PTB and KTB with maximum lengths of 10 and 40, respectively.

5.1.2 Morphism ϕ

After obtaining Z, we randomly generate vocabulary automorphisms ϕ for each Z. Since the size of D_ϕ^Z is $|\phi|$ times the size of Z, we only consider the morphisms such that $|\phi| = 2$ to save computational resources, where 2 is the smallest order satisfying condition $|\phi| = 2n$ $(n > 0)$.

To generate such morphisms, we first collect all the words from train, dev, and test splits; we then randomly shuffle the vocabulary list V to obtain $\phi(V[i]) = V[-i]$.[14] In this way, we randomly generate three morphisms for each of PTB10 and KTB10, but two morphisms for each of PTB40 and KTB40 due to computational resource limit. Table 2 summarizes the size of the generated datasets D_ϕ^Z. Note that although the train vocabulary size of D_ϕ^Z may differ depending on the randomly generated ϕ, the vocabulary sizes of the generated datasets turn out to be mostly the same across different random seeds in our setting.

5.2 Models

In this paper, we analyze three popular unsupervised parsing models: DIORA (Drozdov et al., 2019), PRPN (Shen et al., 2018), and URNNG (Kim et al., 2019b). DIORA is an auto-encoder-based discriminative model using inside-outside dynamic programming. PRPN is a neural language model that jointly learns syntactic structures by utilizing a gate mechanism. URNNG is a transition-based model, an unsupervised version of RNNG (Dyer et al., 2016) that explicitly models top-down generation in language modeling.

We use the implementations released by the authors of the models.[15][16][17] As for the hyperparameters, we basically use those from the original papers and author implementations.[18] Whereas DIORA originally uses pre-trained word embedding such as ELMo (Peters et al., 2018), we instead use one-hot encoding for our analysis.[19] To reduce learning time and amount of computation, training is terminated when the training loss converges. In addition, we apply early stopping when the validation loss is not improved for five epochs. We train parsers with 15 different random seeds for each dataset. For each training, we save the best-performing model in terms of validation loss and use it for analysis.

Dataset	Train	Dev	Test	Vocab
D_*^{PTB10}	11.5K	0.5K	4.8K	7.8K
D_*^{PTB40}	76.5K	3.2K	4.8K	19.0K
D_*^{KTB10}	29.5K	3.8K	7.3K	14.1K
D_*^{KTB40}	56.9K	7.1K	7.3K	14.3K

Table 2: Summary of dataset size. The Vocab column is the vocabulary size of train data. The vocabulary sizes are mostly the same for randomly generated different ϕ.

5.3 Evaluation

First, for each trained parser m, we compute the average $\bar{B}_m$ of branching scores $B(t)$ over the output tree structures for the test data.[20] Next, for each dataset and unsupervised parsing model, we calculate the mean of $\bar{B}_m$ over the parsers trained with different random seeds. Note that while each trained parser m may be biased, there is equally likely to be another trained parser m' that exhibits the opposite score $\bar{B}_{m'} = -\bar{B}_m$ and cancels out the mean of $\bar{B}_m$ to zero if an unsupervised parsing

[11] https://github.com/i-lijun/UnsupConstParseEval

[12] The sentences in the test split are not filtered.

[13] The preprocessing procedure specific to each target model is also applied to D_ϕ^Z.

[14] The morphisms must be consistent across the train, dev, and test splits and cannot be generated for each of these splits.

[15] https://github.com/iesl/diora

[16] https://github.com/yikangshen/PRPN

[17] https://github.com/harvardnlp/urnng

[18] Details are shown in Appendix D.

[19] In the implementation, the pre-trained word embeddings are multiplied by a trainable matrix. In our case, since we use one-hot encoding, the matrix can be viewed as randomly initialized trainable word embeddings.

[20] Trivial sentences of length ≤ 2 are not included in the evaluation.

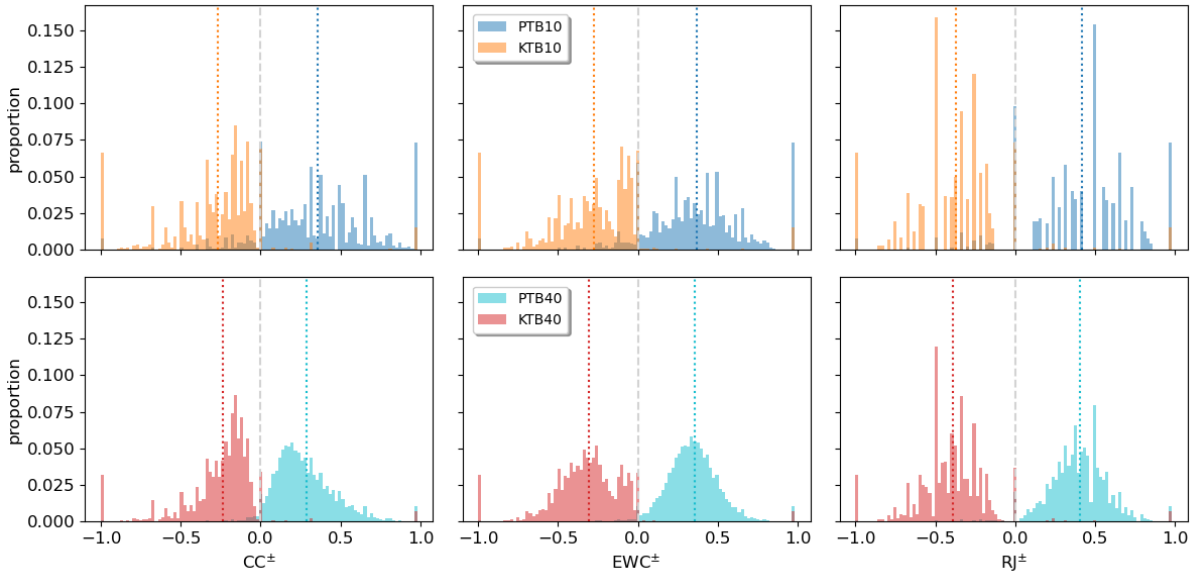


Figure 3: Histograms of branching measures calculated for the gold trees. The top row is for PTB10 and KTB10; the bottom row is for PTB40 and KTB40. Each dotted line shows the mean value for the corresponding dataset. Note that the negative and positive values correspond to left and right-branching structures, respectively.

model is not biased.

6 Results and Discussion

6.1 Branching of Gold Trees

In section 2, we extended the existing imbalance measures for binary trees to measures for branching directions of general n-ary trees. First, we examine whether these branching measures can successfully quantify the branching directions of syntactic trees of natural languages. Figure 3 shows the histogram of the branching scores calculated for the preprocessed treebanks PTB10, PTB40, KTB10, and KTB40 using $CC^{\pm}$, $EWC^{\pm}$, and $RJ^{\pm}$. In Figure 3, it can be seen that, for all branching measures, the gold trees of KTB10 and KTB40 show negative branching scores indicating, that the trees are left-branching, while those of PTB10 and PTB40 are mostly positive and hence right-branching. This supports that our extended branching measures can capture the difference in the branching direction of natural languages.

In Figure 3, for PTB40 and KTB40, the means (dotted lines) and the modes are mostly consistent, but for $CC^{\pm}$, the modes are closer to 0 than the means. This may be due to the fact that $CC^{\pm}$ puts more weight on the branches near the root, and the branches near the leaves are evaluated more weakly than the other two measures. It is also interesting to note that, even though the word order is not completely reversed between Japanese and English (SOV and SVO, respectively), the distributions in Figure 3 are line-symmetric with little overlap.

6.2 Branching of Unsupervised Parsers

Figure 4 shows the branching scores for the three unsupervised parsing models, DIORA, PRPN, and URNNG, averaged over different random seeds.[21] The y-axes show the datasets used for training and testing. In Figure 4, it can be seen that DIORA, PRPN, and URNNG show different results. The branching scores for DIORA are close to 0 for all the datasets and branching measures, suggesting that it has no inherent branching bias. On the other hand, PRPN consistently shows a right-branching bias for all datasets and measures. In fact, Dyer et al. (2019) point out the right-branching bias of PRPN by theoretically proving that PRPN cannot parse certain structures. Although the proof by Dyer et al. (2019) is model-specific, the fact that the right-branching bias of PRPN was also observed in our experiment suggests that our branching bias analysis utilizing tree-shape uncertainty is valid and effective while being model-agnostic. Interestingly, URNNG shows different branching biases depending on the datasets, unlike DIORA and PRPN. For example, URNNG shows branching scores close to 1, i.e., completely right-branching, for D_*^{PTB10}, while it has smaller scores for D_*^{KTB10}

[21] More detailed plots are shown in Appendix F.

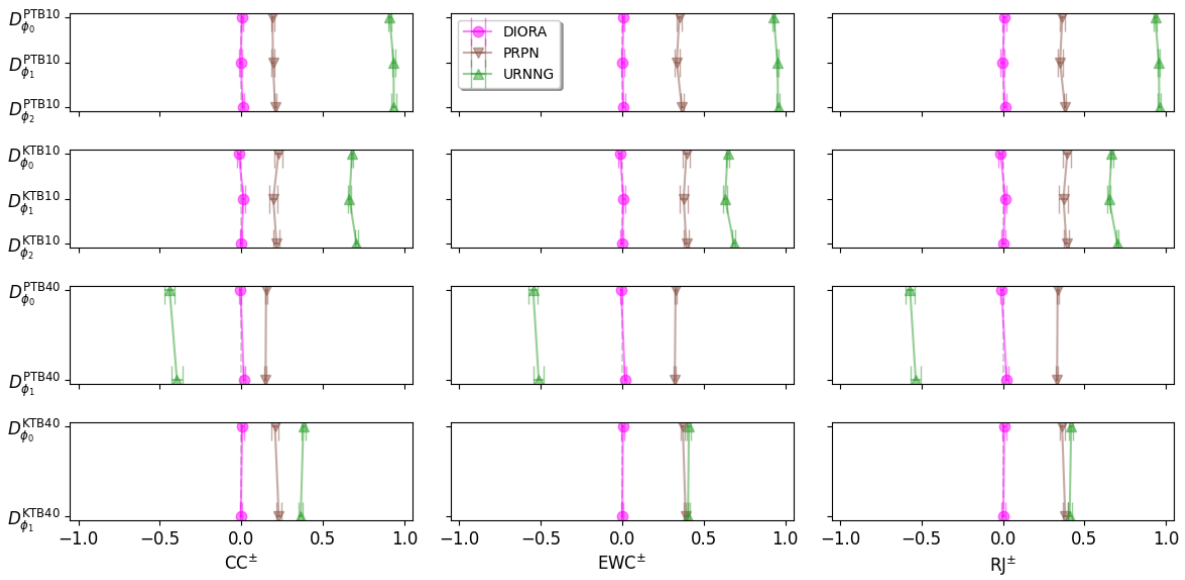


Figure 4: Average branching scores of DIORA, PRPN, and URNNG trained on each D_ϕ^Z. The scores are calculated on the parser outputs on test splits. The top row is for the datasets created based on PTB10 and KTB10. The bottom row is for those based on PTB40 and KTB40. Note that each ϕ_i in D_ϕ^Z is a morphism generated randomly with a seed i for Z. Error bars show standard errors.

and D_*^{KTB40}, and even negative scores, i.e., left-branching, for D_*^{PTB40}. Since we can reasonably expect the branching direction of D_ϕ^Z to be uncertain from Theorem 1 , we conjecture that the branching bias of URNNG is sensitive to factors other than the branching direction of the texts, such as dataset size, vocabulary size, word frequency, sentence length, and so on.

Following Li et al. (2020b), we also evaluate the models on shorter sequences by setting the maximum length to 10 for the test data.[22] While the results for DIORA and PRPN are mostly the same, URNNG shows slightly more right-branching results for D_*^{PTB40} compared to when the maximum length is not set for test data. This also indicates the URNNG's sensitivity to sentence length.

6.3 Practical Implication

One important application of bias analysis is correct model performance evaluations by, for example, rescaling or reranking the parsing scores with respect to the biases. However, using the bias observed in Figure 4 for such a "model performance correction" is theoretically non-trivial for two reasons. Firstly, the numerical relation, e.g., whether it can be approximated linearly, between bias scores and model performance scores, e.g., F1 parsing score and likelihood, is not clear yet. Secondly, since what we know from this experiment is the bias for the texts that contain no potential branching bias, it is possible that models show different biases for the base text Z. At least, there is currently no theoretical guarantee that the bias is the same for Z and D_ϕ^Z for any model. Nevertheless, the results in Figure 4 still prove that the models are somehow biased, and they are still useful as a milestone in developing and using unsupervised parsing models.

7 Conclusion

This paper proposes a theoretically founded branching bias analysis of unsupervised parsing models. We consider the possibility of the same texts being generated by PCFGs that assign differently shaped tree structures, which we formalize as tree-shape uncertainty. We derive sufficient conditions for tree-shape uncertainty with respect to branching direction under a reasonable grammar complexity assumption and use it to construct text corpora that are expected to contain no potential branching bias. By training unsupervised parsers on such unbiased texts, we demonstrate that the inherent branching bias of models can be directly observed by quantifying the branching direction of the output tree structures without the need to compare them with gold trees.

[22] The results are shown in Appendix E.

Acknowledgements

We are grateful to the anonymous reviewers and Junjie Chen for their helpful feedback. This work was partially supported by JST SPRING, Grant Number JPMJSP2108, and by JST CREST, Grant Number JPMJCR2114, Japan.

References

Alastair Butler, Zhu Hong, Tomoko Hotta, Ruriko Otomo, Kei Yoshimoto, and Zhen Zhou. 2012. Keyaki treebank: phrase structure with functional information for japanese. In *Proceedings of Text Annotation Workshop*, page 41.

Donald H. Colless. 1982. Review of Phylogenetics: The Theory and Practice of Phylogenetic Systematics. *Systematic Zoology*, 31(1):100–104.

Tomás M. Coronado, Arnau Mir, Francesc Rosselló, and Lucía Rotger. 2020. On Sackin's original proposal: The variance of the leaves' depths as a phylogenetic balance index. *BMC Bioinformatics*, 21(1):154.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. 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 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Andrew Drozdov, Patrick Verga, Mohit Yadav, Mohit Iyyer, and Andrew McCallum. 2019. Unsupervised Latent Tree Induction with Deep Inside-Outside Recursive Auto-Encoders. 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 1129–1141, Minneapolis, Minnesota. 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.

Chris Dyer, Gábor Melis, and Phil Blunsom. 2019. A Critical Analysis of Biased Parsers in Unsupervised Parsing. *arXiv:1909.09428 [cs]*.

Mareike Fischer. 2021. Extremal Values of the Sackin Tree Balance Index. *Annals of Combinatorics*, 25(2):515–541.

Mareike Fischer, Lina Herbst, Sophie Kersting, Luise Kühn, and Kristina Wicke. 2021. Tree balance indices: A comprehensive survey.

Seymour Ginsburg and Nancy Lynch. 1976. Size complexity in context-free grammars forms. *Journal of the ACM*, 23(4):582–598.

J. Gruska. 1971. Complexity and unambiguity of context-free grammars and languages. *Information and Control*, 18(5):502–519.

Stephen B. Heard. 1992. Patterns in Tree Balance Among Cladistic, Phenetic, and Randomly Generated Phylogenetic Trees. *Evolution*, 46(6):1818–1826.

John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman. 2001. Introduction to automata theory, languages, and computation, 2nd edition. *ACM SIGACT News*, 32(1):60–65.

Lifeng Jin, Finale Doshi-Velez, Timothy Miller, William Schuler, and Lane Schwartz. 2018. Unsupervised Grammar Induction with Depth-bounded PCFG. *Transactions of the Association for Computational Linguistics*, 6:211–224.

Mark Johnson, Thomas Griffiths, and Sharon Goldwater. 2007. Bayesian Inference for PCFGs via Markov Chain Monte Carlo. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 139–146, Rochester, New York. Association for Computational Linguistics.

Sophie J. Kersting and Mareike Fischer. 2021. Measuring tree balance using symmetry nodes — A new balance index and its extremal properties. *Mathematical Biosciences*, 341:108690.

Eugene Kharitonov and Rahma Chaabouni. 2020. What they do when in doubt: A study of inductive biases in seq2seq learners. 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.

Mark Kirkpatrick and Montgomery Slatkin. 1993. Searching for Evolutionary Patterns in the Shape of a Phylogenetic Tree. *Evolution*, 47(4):1171–1181.

Huayang Li, Lemao Liu, Guoping Huang, and Shuming Shi. 2020a. On the Branching Bias of Syntax Extracted from Pre-trained Language Models. In

Findings of the Association for Computational Linguistics: EMNLP 2020, pages 4473–4478, Online. 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.

Percy Liang, Slav Petrov, Michael Jordan, and Dan Klein. 2007. The Infinite PCFG Using Hierarchical Dirichlet Processes. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 688–697, Prague, Czech Republic. 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.

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.

Arne O. Mooers and Stephen B. Heard. 1997. Inferring Evolutionary Process from Phylogenetic Tree Shape. *The Quarterly Review of Biology*, 72(1):31–54.

Melissa M. Norström, Mattia C.F. Prosperi, Rebecca R. Gray, Annika C. Karlsson, and Marco Salemi. 2012. PhyloTempo: A Set of R Scripts for Assessing and Visualizing Temporal Clustering in Genealogies Inferred from Serially Sampled Viral Sequences. *Evolutionary Bioinformatics*, 8:EBO.S9738.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep Contextualized Word Representations. 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 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

James S. Rogers. 1996. Central Moments and Probability Distributions of Three Measures of Phylogenetic Tree Imbalance. *Systematic Biology*, 45(1):99–110.

Kwang-Tsao Shao and Robert R. Sokal. 1990. Tree Balance. *Systematic Zoology*, 39(3):266–276.

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*.

Pengyu Wang and Phil Blunsom. 2013. Collapsed Variational Bayesian Inference for PCFGs. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 173–182, Sofia, Bulgaria. Association for Computational Linguistics.

Xiaohan Zhang, Shaonan Wang, Nan Lin, and Chengqing Zong. 2022. Is the Brain Mechanism for Hierarchical Structure Building Universal Across Languages? An fMRI Study of Chinese and English. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 7852–7861, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

A Limitations and Future Work

First, as described in section 4, one of the major limitations of this study is that it is not clear yet whether natural language corpora are tree-shape uncertain or not. One solution to this problem is to quantify the degree of tree-shape uncertainty instead of considering it as a binary true/false value. For example, in a Bayesian framework, one can consider a prior distribution over grammars and calculate the expected branching scores for a text corpus D.

Next, we only considered the tree-shape uncertainty with respect to branching directions in this paper. However, the definition of tree-shape uncertainty (Definition 2) is general and not limited to branching direction. Extension to other tree shapes, such as the degree of center embedding, is left for future work.

To consider the potential syntactic trees of text corpora, we used PCFG as a grammar formalism. However, while PCFG can generate any finite text corpus D, it has been pointed out that PCFG has a strong independence assumption and does not fully capture the grammatical features of natural languages (Kim et al., 2019a). Considering grammar formalization other than PCFG is an important future work.

B Ethics Statement

Our research focuses on the analysis of the branching bias of unsupervised parsing models, and we do not propose any models to be used in practice. We believe our research does not raise any ethical issues.

C Dataset License

Here, we describe the licenses of the natural language corpora used in this paper. We download the PTB corpus from Linguistic Data Consortium and use it as LDC members.[23] The KTB corpus is published under CC BY 4.0 license.[24]

We confirmed that all the above licenses allow us to use the datasets in our experiment.

D Models

Here, we show the hyperparameter settings for the target unsupervised parsing models. Table 3 shows the hyperparameters for DIORA. Table 4 shows the hyperparameters for PRPN. Table 5 shows the hyperparameters for URNNG.

Parameter	Value
max_epoch	75
batch_size	32
hidden_dim	400
lr	1×10^{-4}
k_neg	100
freq_dist_power	0.75
margin	1.0

Table 3: Hyperparameters for DIORA. The parameter names are based on the author's implementation: https://github.com/iesl/diora

Parameter	Value
epochs	75
batch_size	64
emsize	200
nhid	400
nlayers	2
nslosts	15
nlookback	5
lr	1×10^{-3}
weight_decay	1×10^{-6}
clip	1.0
dropout	0.2
idropout	0.2
rdropout	0.0
tied	True
hard	True
res	0
resolution	0.1

Table 4: Hyperparameters for PRPN. The parameter names are based on the author's implementation: https://github.com/yikangshen/PRPN

To reduce learning time and amount of computation, training was terminated when the training loss converges, i.e., when the absolute difference of the training losses between the current and previous epoch is within 1×10^{-4}. In addition, we apply early stopping when the validation loss is not improved for 5 epochs.

E Results on Short Sentences

Figure 5 shows the mean branching scores calculated for the test data with a maximum length of

[23] https://catalog.ldc.upenn.edu/LDC99T42
[24] http://www.compling.jp/keyaki/index.html

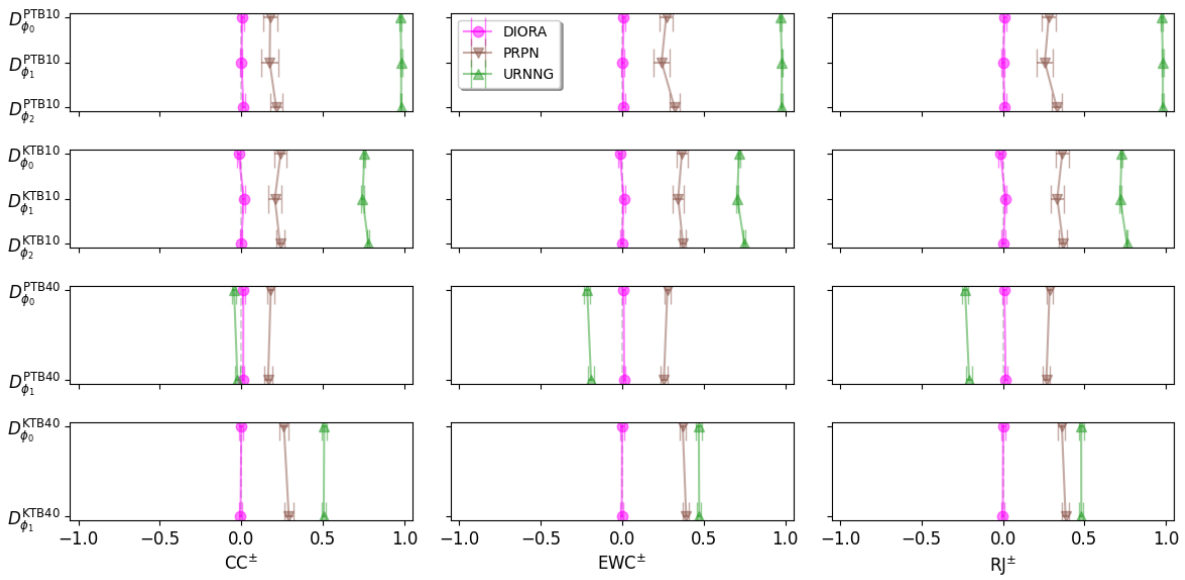


Figure 5: Average branching scores of DIORA, PRPN, and URNNG trained on each D_ϕ^Z. The scores are calculated on the parser outputs on test splits with a maximum length of 10. The top row is for the datasets created based on PTB10 and KTB10. The bottom row is for those based on PTB40 and KTB40. Note that each ϕ_i in D_ϕ^Z is a morphism generated randomly with a seed i for Z. Error bars show standard errors.

10.[25] For DIORA and PRPN, the overall trend is mostly the same as when there is no restriction on the maximum length (Figure 4). However, for URNNG, when the maximum length is set to 10, the branching scores, especially $\mathrm{CC}^{\pm}$, for D_*^{PTB40} are closer to 0 compared to when there is no limit. Nevertheless, for $\mathrm{EWC}^{\pm}$ and $\mathrm{RJ}^{\pm}$, URNNG still shows a left-branching bias. We conjecture that these observations might align with the results reported by Li et al. (2020b): URNNGs trained on PTB40 show higher F1 scores for test sentences with a maximum length of 10 compared to the other models, such as DIORA and PRPN.

F Branching Distributions of Model Outputs

Figure 6, Figure 7, and Figure 8 show the histograms of branching scores calculated for the outputs of DIORA, PRPN, and URNNG, respectively. Each parser is trained on the train split of D_ϕ^Z and evaluated on the train, dev, and test splits. Each dotted vertical line indicates the average branching score $\bar{B}_m$ over the dataset calculated for each parser m trained with different random seeds. Also, note that the results of randomly generated morphisms ϕ are plotted overlaid on the same row since we do not find significant differences between them.

[25] Note that the results of the same trained parsers are shown in Figure 4 and Figure 5.

Parameter	Value
num_epochs	18
min_epochs	8
batch_size	16
train_q_epochs	2
w_dim	650
h_dim	650
q_dim	256
num_layers	1
dropout	0.5
samples	8
lr	1.0
q_lr	1×10^{-4}
action_lr	0.1
decay	0.5
kl_warmup	2
max_grad_norm	5.0
q_max_grad_norm	1.0

Table 5: Hyperparameters for URNNG. The parameter names are based on the author's implementation: https://github.com/harvardnlp/urnng

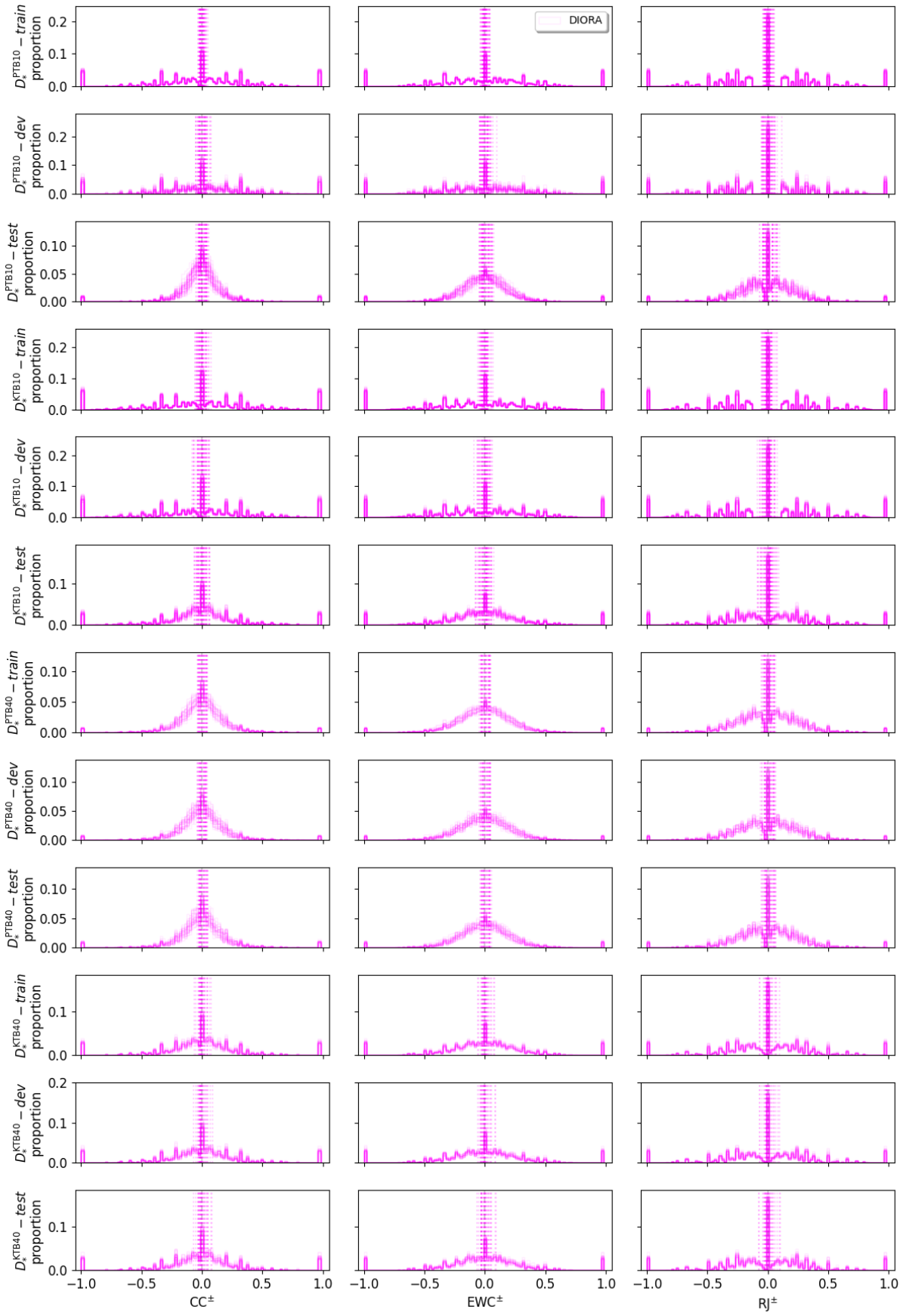


Figure 6: Histograms of branching scores calculated for the outputs of DIORA. Each parser is trained on the train split of D_ϕ^Z and evaluated on the train, dev, and test splits. Each dotted vertical line shows the mean for each parser.

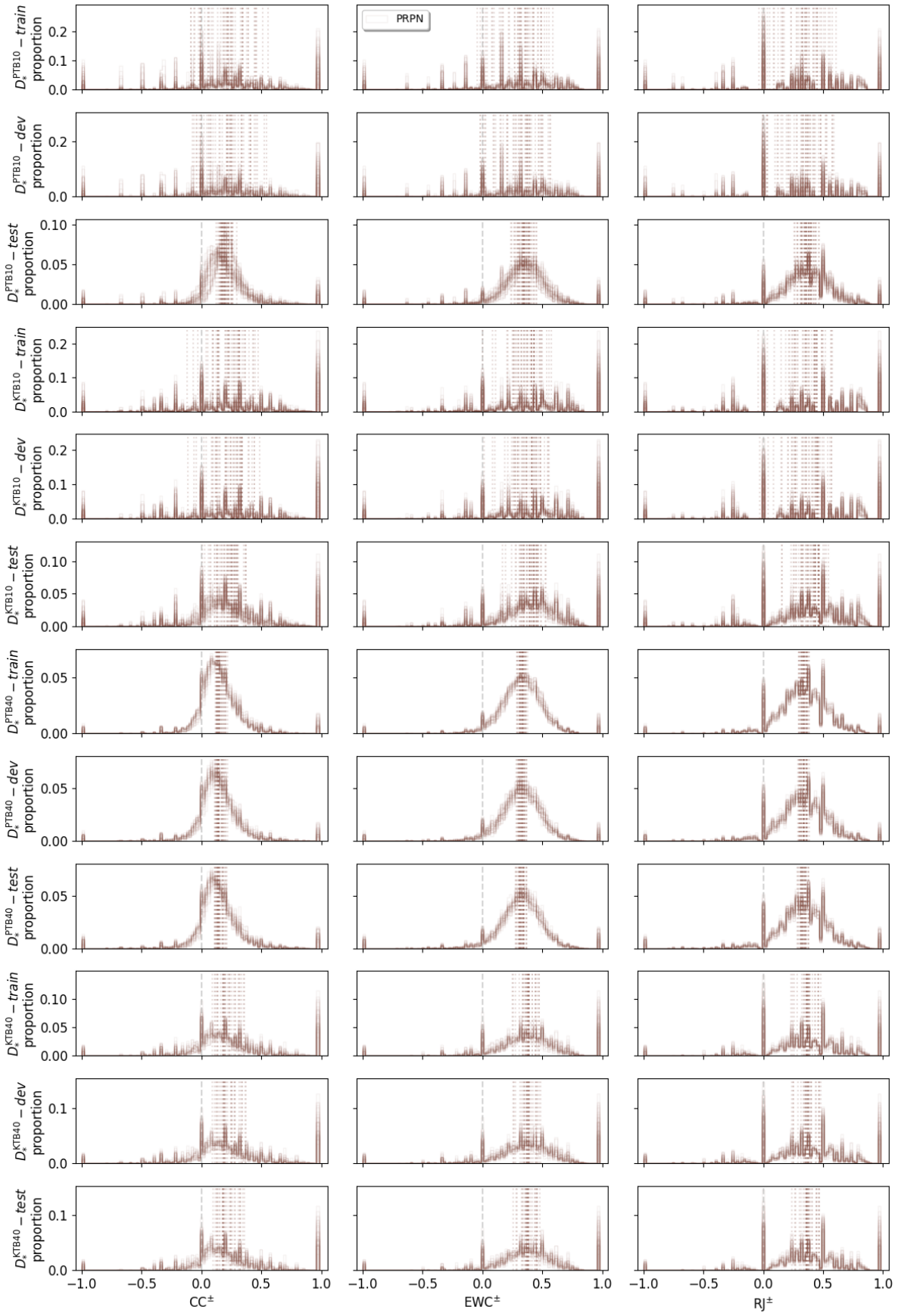


Figure 7: Histograms of branching scores calculated for the outputs of PRPN. Each parser is trained on the train split of D_ϕ^Z and evaluated on the train, dev, and test splits. Each dotted vertical line shows the mean for each parser.

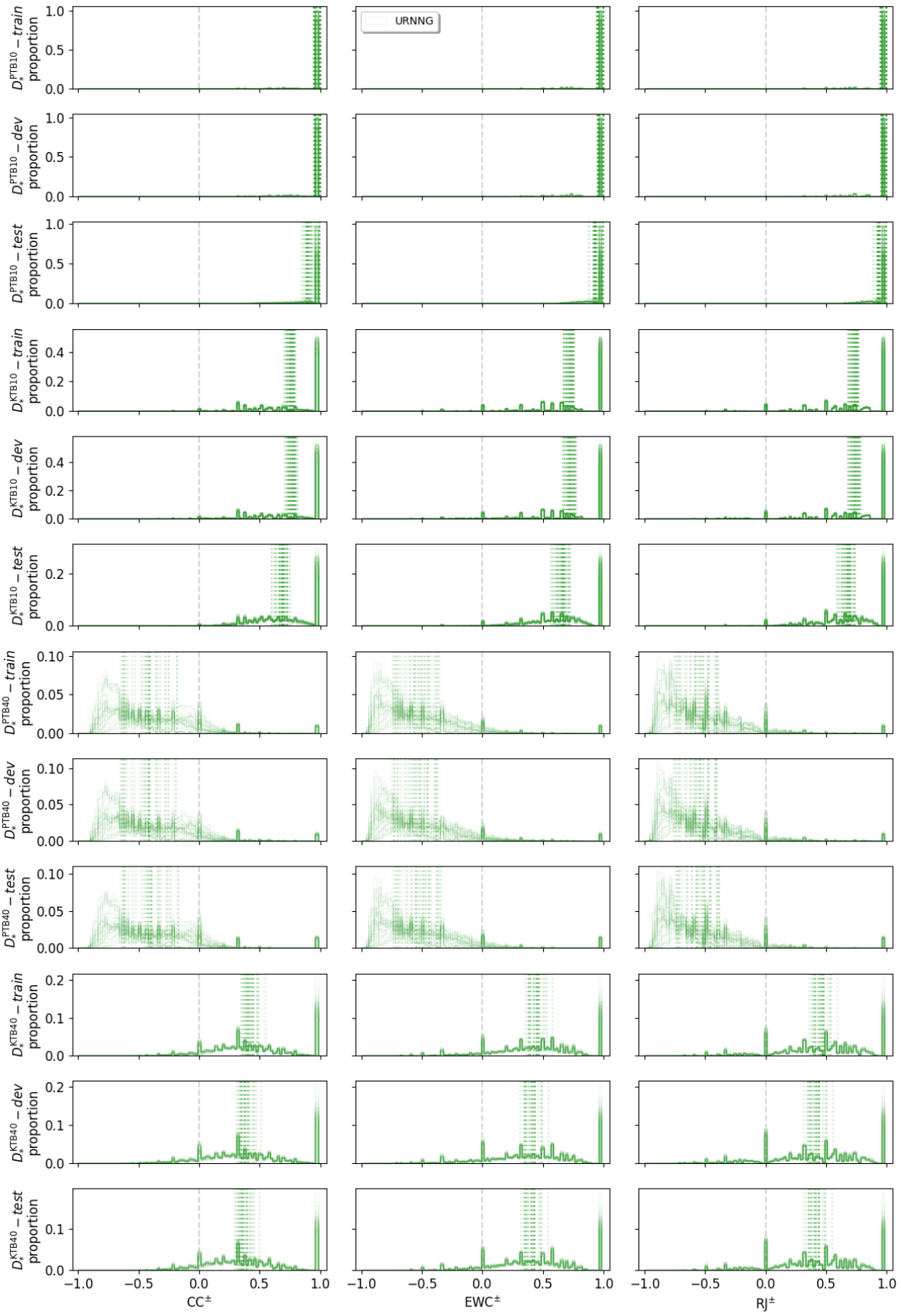


Figure 8: Histograms of branching scores calculated for the outputs of URNNG. Each parser is trained on the train split of D_ϕ^Z and evaluated on the train, dev, and test splits. Each dotted vertical line shows the mean for each parser.