Unsupervised Full Constituency Parsing with Neighboring Distribution Divergence

Anonymous submission

Abstract

Unsupervised constituency parsing has been explored much but is still far from being solved as currently mainstream unsupervised constituency parser only captures the unlabeled structure of sentences. Properties in the substitution of constituents make it possible to detect constituents in a particular label. We propose an unsupervised and training-free labeling procedure by leveraging a newly introduced metric, Neighboring Distribution Divergence (NDD), which evaluates semantic changes caused by editions. We develop NDD into Dual POS-NDD (DP-NDD) and build templates called "molds" to extract labeled constituents from sentences. We show that DP-NDD labels constituents precisely and inducts more accurate unlabeled constituency trees than all previous unsupervised methods. Following two frameworks for labeled constituency trees inference, we set the new state-of-the-art for unlabeled F1 and labeled F1. Further studies show our approach can be scaled to other span labeling problems, i.e., named entity recognition.

1 Introduction

Constituency parsing is a basic but crucial parsing task in natural language processing. Constituency parsers are required to build parsing trees for sentences consisting of spans representing constituents such as noun phrases and verb phrases. Parsed constituency trees can be applied to many downstream systems (Lee et al., 2013; Chen et al., 2015; Zhong et al., 2020).

Since the introduction of deep learning into natural language processing, supervised neural networks have achieved remarkable success in constituency parsing (Kitaev and Klein, 2018; Liu et al., 2018; Nguyen et al., 2020; Zhang et al., 2020b). Unfortunately, the need for large annotated datasets limits the performance of supervised systems on languages of low resources. As the result, many unsupervised systems have been proposed

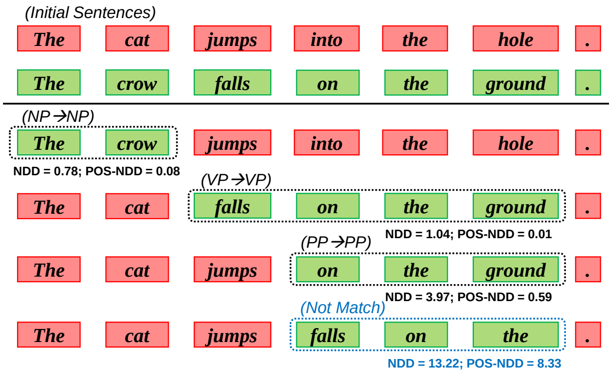

Figure 1: Examples for substitution property of constituents among sentences. Neighboring Distribution Divergence performs well on detecting plausible substitutions.

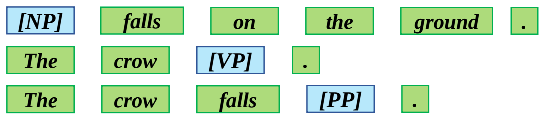

Figure 2: Molds constructed from examples in Figure 1.

for constituency parsing (Drozdov et al., 2019b; Kim et al., 2020; Shen et al., 2021; Sahay et al., 2021) by exploiting unlabeled corpus.

Current unsupervised constituency parsing systems are still far from the complete procedure, mainly because most of these systems only induct an unlabeled structure of the constituency tree. Rare attention has been paid to label constituents except for clustering (Drozdov et al., 2019a). Constituents have fine properties which mitigate the difficulty of unsupervised detection and labeling. Labels of constituents are very different from labels in other classification tasks since they represent syntactic roles. For a noun phrase in a sentence, it will be of high probability to play as a plausible noun phrase in another sentence, as shown in Figure 1. This phenomenon is also true for verb phrases and preposition phrases.

Towards better unsupervised full constituency

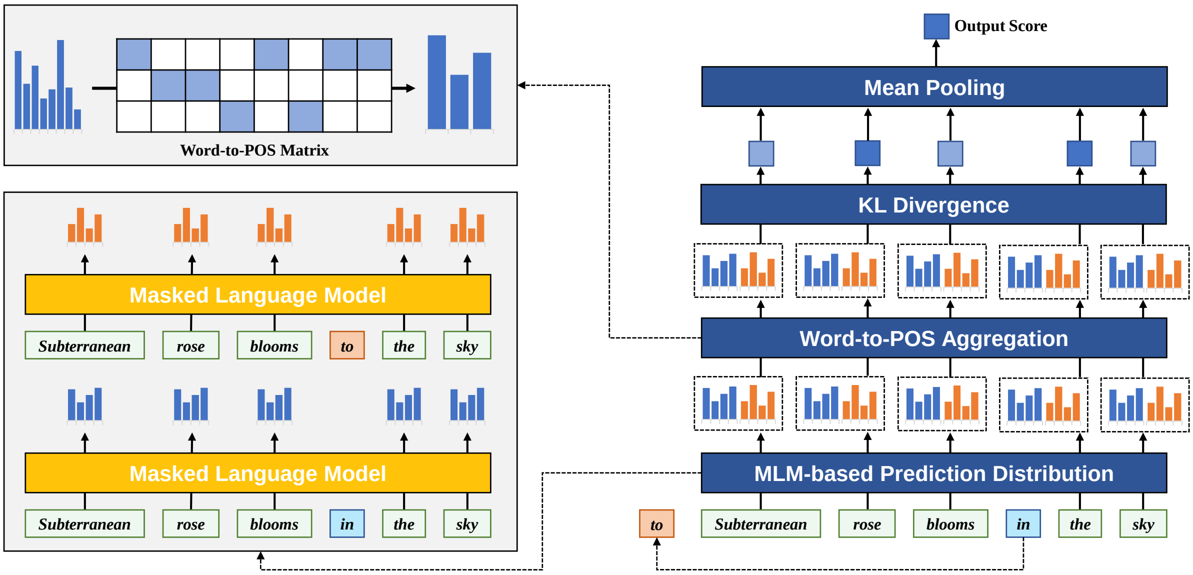

Figure 3: Calculating procedure for POS-NDD.

parsing, we leverage a recently proposed metric, Neighboring Distribution Divergence (NDD) (Peng et al., 2021), to automatically detect labeled constituents in sentences. NDD is a Pre-trained Language Model-based (PLM-based) (Devlin et al., 2019) metric and is initially proposed to detect semantic changes caused by editions.

In practice, we construct very few templates called "molds" as shown in Figure 2. We judge whether a span to be a constituent by fill it into the phrase mask (*[NP]*, *[VP]*, ...) and using NDD to detect the semantic similarity between the filled sentence and the initial sentence. If NDD is under a certain threshold, our method will predict the span to be a constituent.

To further boost the efficiency and performance of our method, we modify NDD into POS-NDD, which only considers the likeliness of part-of-speech (POS) sequences since the initial NDD is too sensitive to precise semantic differences. Also, we use a dual detecting method that evaluates both the **substitution to mold** and **substitution from mold** to better link constituents with the same label together. We named our final metric Dual POS-NDD (DP-NDD).

We experiment on Penn Treebanks to construct labeled constituency trees and label predicted treebanks from other unsupervised constituency parsers. Results from our experiments verify DP-NDD to be capable of inducting labeled constituency trees and labeling unlabeled constituents. Based on DP-NDD molds, we introduce two novel frameworks for unsupervised constituency parsing. Our algorithm parses by following simple rules but results in remarkable results which outperform all previous unsupervised parsers on the WSJ test dataset. Our algorithms set the first strong baseline in recent years for labeled F1 score. Our main contributions are concluded as follows:

- We propose an unsupervised method other than clustering for full constituency parsing, which involves constituent labeling.
- We introduce novel frameworks for unsupervised constituency parsing, which set a new state-of-the-art for unlabeled F1 and strong baselines for labeled F1.
- We introduce variants of NDD, POS-NDD, and DP-NDD, which are less sensitive to semantic differences between sentences and perform well for constituent detecting.
- We first model parsing in an editing form, which is different from the conventional practice, which aids editions with parsing results.

2 Neighboring Distribution Divergence

2.1 Background

We briefly describe the NDD metric in this section as the background for further discussion. More details like motivation and explanation can be referred to (Peng et al., 2021).

Given a W sentence with n-word $W = [w_1, w_2, \cdots, w_n]$, we use an edition E to convert

W to an edited sentence $W' = E(W)$. As we only use substitution for unsupervised constituency parsing, we limit E to a substituting operation which substitutes i-th to j-th word in W with a span $V = [v_1, v_2, \cdots, v_m]$.

$$\begin{aligned} & W' = E(W) \\ =& [w_1, \cdots, w_{i-1}, v_1, \cdots, v_m, w_{j+1}, \cdots, w_n] \end{aligned}$$

Then we evaluate the semantic disturbance on the overlapped part $[w_1, \cdots, w_{i-1}, w_{j+1}, \cdots, w_n]$ between the initial and edited sentences. For estimation, we use a masked language model to get the distribution of predicted words for each masked position before and after the edition.

$$\begin{aligned} W_i^{mask} &= [w_1, \cdots, w_{i-1}, \text{[MASK]}, w_{i+1}, \cdots, w_n]; \\ R &= PLM(W_i^{mask}); d_i = \text{softmax}(R_i) \in \mathbb{R}^c \end{aligned}$$

We first mask the i-th word in W and use the PLM to predict the distribution d_i on the masked position. Here, d_i is a $\mathbb{R}^c$ tensor, which refers to the existence probability of the words in a c-word dictionary of the PLM. We do this for the overlapped part mentioned above, both in W and W'.

After we get the predicted distributions for W and W', we use KL divergence to calculate the difference between the two distributions.

$$div_i = D_{KL}(d'_i || d_i) = \sum_{j=1}^{c} d'_{ij} \log(\frac{d'_{ij}}{d_{ij}})$$

Finally, we integrate the divergence values via a mean pooling layer.

$$\text{NDD}(W, W') = \sum_{k \in [1, \cdots, i-1, j+1, \cdots, n]} \frac{div_k}{n - (j - i + 1)}$$

According to the cases in (Peng et al., 2021), NDD is capable of capturing precise semantics changes. We will show in Section 2.3 how to use modified NDD to construct molds for unsupervised constituency parsing.

2.2 POS-NDD

NDD performs well on supervising editions, but it might be too sensitive to some precise semantic difference as in the explanation for Table 1 later. To adapt NDD to constituency parsing, we modify NDD's calculating procedure to concentrate on the structural rather than semantic difference.

Sentence	Sem.	Str.	POS-NDD	NDD
The spider built its nest in the cave.	-	-	0.00	0.00
The spider made its nest in the cave.	✗	✗	0.69	2.67
The spider caught the pests in the cave.	✓	✗	0.81	7.45
The spider a wasted bridge in the cave.	✓	✓	6.42	18.39

Table 1: Comparison between NDD and POS-NDD for semantic and structural change detection. The initial sentence is *"The spider built its nest in the cave."* **Sem.**: If there is a semantic change. **Str.**: If there is a structural change.

To do so, we gather the predicted words with the same POS together by summing up their existence probability. For implementation, we construct a word-to-POS matrix M as shown in Figure 3. M is a 2-dimension tensor of shape $\mathbb{R}^{p \times c}$ where p is the number of POS classes, and c is the scale of PLM's dictionary. M is constructed following the rule as follows:

$$M_{ij} = \begin{cases} 0, \text{if } j\text{-th word dictionary not in } i\text{-th POS class} \\ 1, \text{if } j\text{-th word dictionary in } i\text{-th POS class} \end{cases}$$

With M, we gather the existence probability of words in the same POS class together and calculate the KL divergence for POS-NDD. The weighted sum in POS-NDD calculation is the same as in NDD.

$$\begin{aligned} q_i &= M d_i, q'_i = M d'_i \\ div_i^{pos} &= D_{KL}(q'_i || q_i) = \sum_{j=1}^{p} q'_{ij} \log(\frac{q'_{ij}}{q_{ij}}) \end{aligned}$$

The comparison between the initial NDD and modified POS-NDD is presented in Table 1. In the first example, we edit the sentence while keeping both the semantics and structure unchanged—the edition results in rather low values for both NDD and POS-NDD. In the second example, our edition does not convert the sentence's structure but different semantics. Initial NDD is sensitive to this change as its value raises to nearly $\times 3$. In contrast, POS-NDD is less likely to be affected by semantics and concentrates more on sentence structure. The last example includes an edition that breaks the sentence's structure by substituting a verb phrase with a probable noun phrase. As the value of POS-NDD raises to almost $\times 8$, POS-NDD is verified to detect this anomaly.

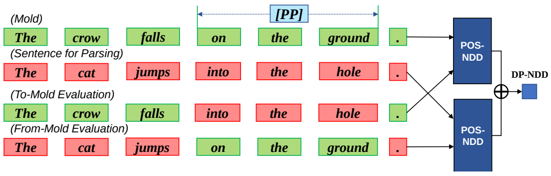

Figure 4: Dual mold for detecting constituents.

2.3 NDD-based Dual Mold

Based on POS-NDD, we build "molds" that can discern constituents in sentences. Our mold is defined as a quaternion (W, i, j, l) where W is an n-word sentence. i, j refers to the start and end position of the span for substitution. l refers to the constituent's label. Suppose we want to evaluate the probability of a span $V[s:t]$ (words from s-th to t-th position in an m-word sentence V) to a constituent with the label l. In that case, we will substitute $W[i:j]$ with $V[s:t]$ and calculate the POS-NDD between the sentence before and after the substitution.

$$\begin{aligned} S_{s,t}^{l,tm} &= \text{POS-NDD}(W, W') \\ W' &= [w_1, \cdots, w_{i-1}, v_s, \cdots, v_t, w_{j+1}, \cdots, w_n] \end{aligned}$$

We call this score **To-Mold** score as it is obtained by substituting spans in molds. Likewise, we also have a **From-Mold** score which is obtained by substituting spans in sentences for parsing with spans in molds.

$$\begin{aligned} S_{s,t}^{l,fm} &= \text{POS-NDD}(V, V') \\ V' &= [v_1, \cdots, v_{s-1}, w_i, \cdots, w_j, v_{t+1}, \cdots, v_m] \end{aligned}$$

We finally add To-Mold and From-Mold scores together for whole evaluation,

$$S_{s,t}^{l} = S_{s,t}^{l,tm} + S_{s,t}^{l,fm}$$

which forms a dual calculating procedure as shown in Figure 4. We thus name our method Dual POS-NDD. A lower DP-NDD score S refers to less disturbance in substituting to and by a constituent with label l and will thus reflect the likelihood of the span to be a constituent with the same label.

3 Constituency Tree Constructing

In this section, we introduce two frameworks that we use in experiments to generate labeled constituency trees.

3.1 Labeled Span Generating

Labeled Span Generating (LSG) is to directly generate labeled spans with DP-NDD molds and then integrate spans with different labels together to construct the full labeled tree. Our LSG algorithm is much simpler than previous rules-based systems as it only requires 4 steps for constituency parsing.

- **Candidate Selection** We first use simple linguistic rules to sample some candidates for a constituent label. For a span $V[s:t]$, we match the POS tags of $V[s-1]$, $V[s]$, $V[t]$, $V[t+1]$ to a POS list to roughly decide whether the span is a plausible candidate for constituent or not. For special labels, span length is also taken into consideration.

- **DP-NDD Scoring** We then use our Dual POS-NDD molds to score the sampled candidates as previously described. There are multiple molds for evaluation for some labels as difference exists among constituents with the same label. We choose the minimal value of DP-NDD scores from the molds.

- **Conflict Removing** After scoring, we remove spans which conflict with previously parsed span. Conflicting spans are those overlapping with previous spans by $(s < s', s' < t, t < t')$ or $(s' < s, s < t', t' < t)$.

- **Filtering and Overlapping Removing** Finally, we filter the spans by only keeping the spans with DP-NDD scores under a certain threshold. Then, we remove spans overlapped with other spans of the same label. If $(s < s', s' < t, t < t')$ or $(s' < s, s < t', t' < t)$, we only keep the span with higher DP-NDD. But if $(s < s', t' < t)$ or $(s' < s, t < t')$, we add a tolerance factor to the algorithm to keep both spans if the difference between the two scores is lower than the tolerance.

We execute the 4 steps above for each label and finally integrate spans parsed from each iteration to construct the whole labeled constituency tree.

3.2 Unlabeled Tree Labeling

Unlabeled Tree Labeling (ULT) uses a parsing algorithm to induct unlabeled treebanks from sentences and then uses DP-NDD molds to label the spans in the tree. Our UTL only annotates the edges in the tree with no changes in the tree structure. For each label, we use a mold to calculate the DP-NDD score. The span is labeled as the label of the mold to minimize the DP-NDD. In practice, we maximize the exponential of negative DP-NDD.

$$l_{s,t} = \underset{l}{\text{argmax}}(e^{-S^l_{s,t}})$$

We further refine the prediction by incorporating POS tags. We use the posterior probability collected before for approximation to induct the label of a span using the POS of start and end words.

$$l_{s,t} = \underset{l}{\text{argmax}}(\alpha e^{-S^l_{s,t}})$$
$$\alpha = p(l|\text{POS}(V[s]))p(l|\text{POS}(V[t]))$$

where we add α as a modifier to incorporate POS-based probability into prediction.

4 Experiment

4.1 Data and Configuration

We experiment with our parsing algorithm on Penn Treebank for Constituency Parsing. As our method is training-free, we only use the first 50 sentences in the development dataset to construct molds and handcraft some simple ones. We do not use the training dataset and test our algorithm on the test dataset. We use molds of a number fewer than 25. We apply BERT-base-cased (Devlin et al., 2019) as the PLM for calculating DP-NDD. We also have two configurations for thresholds and tolerances in LSG. A strict configuration will produce fewer predicted spans and will thus result in higher labeled F1 scores, while a loose configuration will, on the opposite, result in higher unlabeled F1 scores. For ULT, we use DIORA+PP (Post-processing) (Drozdov et al., 2019b) which is a strong baseline for unsupervised constituency parsing to induct the unlabeled treebanks. For probability approximation in POS-based refinement for UTL, we only use POS tags in the development dataset. Specific molds, POS-based rules, thresholds, and tolerances can be referred to Appendix A.

4.2 Main Result

Our main results and the comparison with previously reported results are shown in Table 2. We evaluate the models by unlabeled F1 for comparison with previous parsing methods. The performances of UTL and LSG are both reported to set baselines for those two frameworks.

Method	UF1	LF1*
LB	13.1	-
RB	16.5	-
RL-SPINN (Choi et al., 2018)	13.2	-
ST-Gumbel - GRU (Yogatama et al., 2017)	22.8	-
PRPN (Shen et al., 2018a)	38.3	-
BERT-base (Kim et al., 2020)	42.3	-
ON-LSTM (Shen et al., 2019)	47.7	-
XLNet-base (Kim et al., 2020)	48.3	-
DIORA (Drozdov et al., 2019b)	48.9	-
Tree-T (Wang et al., 2019)	49.5	-
StrctFormer (Shen et al., 2021)	54.0	-
PRPN+PP (Drozdov et al., 2019b)	45.2	-
DIORA+PP (Drozdov et al., 2019b)	55.7	-
DIORA+PP+Aug (Sahay et al., 2021)	58.3	-
DIORA+PP+Clustering (Drozdov et al., 2019a)	59.7	50.2
Neural PCFG (Kim et al., 2019)	50.8	-
Compound PCFG (Kim et al., 2019)	55.2	-
300D SPINN (Williams et al., 2018)	59.6	-
(LSG) w/o NDD	32.5	25.7
(UTL) DIORA+PP	54.7	36.8
(UTL) DIORA+PP+POS	54.7	47.2
(LSG) Tight DP-NDD	59.3	**55.4**
(LSG) Loose DP-NDD	**61.8**	51.5

Table 2: Comparison on unlabeled and labeled F1 scores among methods for unsupervised constituency parsing on WSJ test dataset. **PP:** Post-processing heuristics. **Aug**: Rule-based Augmentation. *: Multiple edges are kept as constituents can have multiple labels.

From Table 2[1], our DP-NDD-based LSG algorithm, DP-NDD with a loose configuration, outperforms all previous unsupervised methods for constituency parsing and remarkably boosts the state-of-the-art unlabeled F1 score to upper than 60.0. Compared with previous state-of-the-art methods consisting of complex systems like post-processing with numerous linguistic rules, our algorithms are much simpler and have better scalability. We attribute this advance to the power of a pre-trained language model that cast constituents with high structural differences into near spaces in the latent space.

For labeled F1 scores, our algorithms also reach significant performance. DP-NDD with a tight configuration achieves a strong performance of 54.5, which is even higher than most unlabeled F1 results from previous systems. Thus, we claim to have successfully implemented the first unsupervised full constituency parsing in recent years. Moreover, our method involves a much simpler PLM, BERT, than the highest baseline, xlnet, in (Kim et al., 2020), but reaches a much higher performance (13.5 unlabeled F1 score). Compared with the result of

[1]Codes for the unlabeled parser in (Drozdov et al., 2019a) are not released, UTLs are implemented on a weaker baseline (Drozdov et al., 2019b).

Label	UR	LP	LR	LF1	Prop.
NP	66.20	68.49	64.34	66.35	42.08%
VP	38.84	53.54	36.10	43.12	19.75%
ADJP	51.20	14.97	28.89	19.72	2.02%
ADVP	79.84	47.37	70.40	56.63	2.74%
PP	63.84	66.37	51.53	58.02	12.40%

Table 3: Performance of DP-NDD-based LSG algorithm on different labels. Unlabeled results (Unlabeled Recall) are from loose DP-NPP, and labeled (Labeled Precision, Labeled Recall, Labeled F1) results are from tight DP-NPP. **Prop.**: Proportion of labels in test treebanks.

Label	P	R	F1	P†	R†	F1†
NP	88.02	86.70	87.36	91.34	98.86	94.95
VP	99.17	50.70	67.10	98.52	90.26	94.21
ADJP	27.86	75.88	40.76	91.33	37.23	52.90
ADVP	63.19	55.74	59.23	93.93	85.15	89.32
PP	40.32	82.57	54.18	84.30	97.69	90.50

Table 4: Labeling performance of DP-NDD-based UTL algorithm on unlabeled golden edges in WSJ-10 treebanks. †: Refined by POS.

BERT-base in (Kim et al., 2020), the unlabeled F1 score from DP-NDD is 19.5 higher, which shows the high efficiency of the DP-NDD-based method.

Compared with UTL, LSG achieves higher F1 scores in both unlabeled and labeled treebanks. We conclude from this phenomenon that using label-specific method (Our molds are for a certain label) can extract constituents better than parsing spans of different labels with a unified algorithm like in other PLM-based methods (Kim et al., 2020; Shen et al., 2021). For UTL, incorporating POS benefits labeling in this framework much as this lifts the unlabeled F1 score to 8.5 higher.

We also launch an ablation study by removing DP-NDD scores from the LSG framework. LSG without DP-NDD returns all spans that satisfy the POS constraints. Without the guide of DP-NDD, the performance of the LSG algorithm drops dramatically, even to half of the initial implementation. We conclude from this phenomenon that our DP-NDD metric is essential for unsupervised full constituency parsing.

NDD distributions caused by substitution of phrases are presented label-wise in Appendix B, which supports and explains the effectiveness of our NDD-based approach.

5 Analysis and Discussion

5.1 Label-specific Evaluation

We analyze the ability of our LSG algorithm for parsing edges of different labels in this section. We report the unlabeled and labeled performance of the LSG algorithm on different labels. Precision, recall, and F1 score are all considered for labeled treebanks, and only recall is evaluated.

As presented in Table 3, LSG performs well on extracting noun, adverb, and preposition phrases. For these phrases, LSG leads to high results in unlabeled recalls and labeled F1 scores. We mainly attribute the success of LSG to the high performance in discerning noun phrases, which take 42.08% proportion of the constituents. LSG performs relatively weaker for verb, and adjective phrases as patterns of these phrases are more variable. Thereby, LSG will be more likely to confuse them with other phrases when trying to discern. We will elaborate this point in Section 5.3.

In contrast, phrases with regular patterns like adverb phrases and preposition phrases are more likely to be discerned successfully. This phenomenon can be attributed to the matching nature of our algorithm, as The substitution will cause less disturbance if another span in a similar pattern substitutes a span. Take instances in Figure 1 for explanation, substituting *into the hole* with most preposition phrases will only result in subtle disturbance, i.e., *in a warm autumn day*, *before the crashing*. But verb phrases contain a variety of patterns like *is so smart* and *to enjoy their lunch*. Their substitution to the verb phrase *jumps into the hole* will cause much more disturbance. Thus, the selection of molds for verb phrases should be more careful to cover the patterns of verb phrases. But this remains another problem that these patterns may be confused with other phrases like labeling *to enjoy their lunch* to be a preposition phrase. The current structure-oriented LSG algorithm may not offer a proper solution to this confusion, so we plan to leverage precise semantics for a try in the future.

5.2 Labeling Performance

We analyze the labeling performance of our UTL algorithm in this section. To avoid parsing bias caused by parser chosen for constructing unlabeled constituency trees, we follow (Drozdov et al., 2019b), we construct a WSJ-10 dataset by sampling sentences with length under 10 from train, development, and test datasets. Then, constituents including noun, verb, adjective, adverb, and preposition phrases are filtered from these sentences. WSJ-10

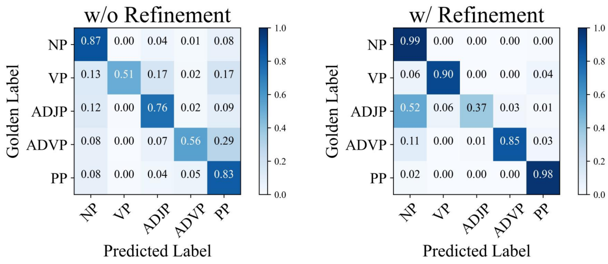

Figure 5: Confusion matrix in labeling WSJ-10 dataset.

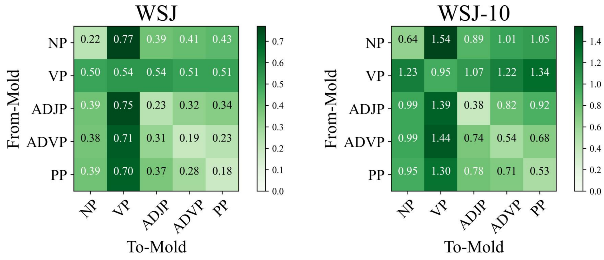

Figure 6: Average semantic disturbance (POS-NDD) caused by constituent substitution.

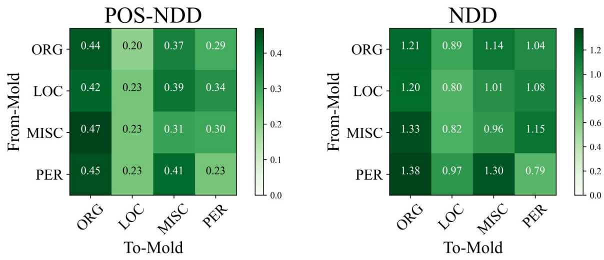

Figure 7: Average semantic disturbance (NDD & POS-NDD) caused by entity substitution for named entity extraction.

contains 17935 golden constituents, and we use the UTL algorithm to label constituents.

We report the experiment results of UTL in Table 4. UTL results in high precision and recall for labeling noun phrases without the refinement of POS information, which verifies its capacity for discerning noun phrase patterns. The adjective phrase remains the most difficult constituent for parsing, and other phrases are of medium parsing difficulty. POS-based refinement works for all phrases by significantly improving the F1 score of noun, verb, adverb, and preposition phrases to around 90.0 while still leaving the adjective phrase as a hard problem due to the difficulty in keeping recall and precision score for adjective phrase balanced.

5.3 Confusion in Constituent Discerning

Following the discussion of labeling performance, we further analyze factors that affect the constituent discerning procedure. We depict the confusion matrix in Figure 5. When POS is not used to help to parse, the most confusing labels are verb and adjective phrases. But the adjective phrase becomes prominently confusing when POS is considered, indicating that some adjective phrases have common POS patterns with noun phrases.

To go deeper into the factors behind the confusion in labeling, we construct disturbance matrices by sampling constituent pairs from WSJ and WSJ-10 datasets. We sample 2000 for each label pair and record the average POS-NDD caused by the substitution. The disturbance matrix is shown to be the direct reflection of pattern differences among constituents. Generally, self disturbance (disturbance between constituents of the same labels) is lower than mutual disturbance (disturbance between constituents of different labels). Moreover, Phrases with more patterns like verb phrases have a higher self disturbance. Referred to the confusion matrix without refinement, confusion appears when the self disturbance is not enough lower than the mutual disturbance, i.e., VP-ADJP, VP-PP, ADVP-PP. The discerning difficulty leads to the drop in recall scores for verb and preposition phrases. For adjective phrases, its precision is affected to drop as some parts of noun phrases, which take a large proportion in constituents, are mislabeled as adjective phrases.

5.4 How about other tasks?

We conduct experiments on named entity recognition (NER) to verify the generality of applying NDD for capturing labeled spans. As all entities are noun phrases, the capability of discerning entities will verify the potential of NDD for more semantically precise tasks. We choose Conll-03 (Sang and Meulder, 2003) as the NER dataset. Conll-03 consists of named entities labeled in 4 types: *[ORG]*, *[PER]*, *[MISC]* and *[PER]*. We sample 2000 pairs of spans in the same way we do in 5.3. We evaluate the average disturbance caused by substituting one span with another using POS-NDD and the original NDD.

Figure 7 shows the disturbance matrix for NER. Compared with constituents, substitution using named entity on average results in much lower POS-NDD since named entities are all noun phrases, as mentioned before. Generally, the self disturbance is lower than mutual disturbance, making it plausible to label named entities with NDD.

Compared with POS-NDD, NDD captures preciser semantics changes as described before. NER experiment results also support that NDD generally performs better in discerning named entities, which share structural similarity with each other, especially for the disturbance caused by substitution to *[LOC]*.

Among labels, *[PER]* is the easiest for discerning as it differs the most from other labels. In contrast, *[ORG]* and *[LOC]* are likely to be confused with each other as they play similar roles in semantics. For instance, we may say *a meeting took place in **UN*** or *a meeting took place in **Paris***, but *a meeting took place in **Jack*** is not semantically plausible. We conclude from the disturbance matrix the difficulty in entity labeling should be ranked as *[ORG]>[MISC]>[LOC]>[PER]*.

6 Related Work

6.1 Unsupervised Constituency Parsing

Since the introduction of language models pre-trained on large corpus like BERT (Devlin et al., 2019), extracting constituents from those models raises as a new way for unsupervised constituency parsing (Kim et al., 2020; Shen et al., 2021). These methods try to extract constituents by calculating the syntactic distance (Shen et al., 2018b) which is supposed to reflect the information association among constituents according to (Shen et al., 2018a; Wang et al., 2019). The extraction of latent trees from PLMs has been studied on a variety of language models in (Kim et al., 2020), which provides rich posterior knowledge for completing unsupervised constituency parsing.

Models trained on masked language models put forward another framework for unsupervised parsing procedures. These models, like DIORA and its variants (Drozdov et al., 2019b; Sahay et al., 2021), have been verified by experiment results to be efficient in discerning constituents from sentences. Unfortunately, these models fail to label the constituents after constructing an unlabeled treebank from sentences. Our method differs from previous work by using constituency molds to match constituents and thus induct their labels. Instead of figuring out direct relationships among words, we allow neighboring words to supervise the structural disturbance caused by substitution. As a result, our method enables labeling on the constituency tree, which implements the full unsupervised constituency parsing.

6.2 Neighboring Distribution Divergence

Neighboring distribution divergence (Peng et al., 2021) is initially proposed to detect semantic changes caused by editions like compression (Xu and Durrett, 2019) or rewriting (Liu et al., 2020). Their experiments on syntactic tree pruning and semantic predicate detection also show NDD to be aware of syntax and semantics. NDD is verified to have the capacity to detect predicates for semantic role labels by deleting or substituting words, which serves as our motivation to transfer this idea to unsupervised constituency parsing. We follow the idea in (Peng et al., 2021) and further adapt it to extract and label constituents.

In previous years, there have been other works that focus on leveraging pre-trained models to produce metrics reflecting syntactic or semantic information. To evaluate the quality of text generation, BERTScore (Zhang et al., 2020a) matches representations from the pre-trained language model of generated and golden sentences. Using pre-trained AMR parsers, (Opitz and Frank, 2021) offers an explainable metric, MF-Score, for AMR-to-sentence generation. MF-Score assigns scores by reconstructing the AMR graphs to compare them with the golden ones. Thus, it evaluates semantic similarity better than conventional sequence matching metrics like BLEU and ROUGE. Encouraged by our success in applying NDD for parsing, we plan to explore these pre-trained model-based automatic metrics for more tasks.

7 Conclusion

In this paper, we explore an unsupervised full constituency parsing procedure that includes constituent labeling. We develop the recently proposed NDD metric into POS-NDD and exploit it by using the dual mold to match constituents. Based on DP-NDD, we introduce two novel frameworks, labeled span generation and unlabeled tree labeling, which establish solid baselines for labeled constituency tree construction and set the new state-of-the-art for unlabeled F1 score. Further studies on constituents with NDD disclose the pattern variety of constituents with the same label and pattern similarity among constituents with different labels. Experiments on the NER dataset verify the generalization of our method to other tasks.

References

Changge Chen, Peilu Wang, and Hai Zhao. 2015. Shallow discourse parsing using constituent parsing tree. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning - Shared Task*, pages 37–41, Beijing, China. Association for Computational Linguistics.

Jihun Choi, Kang Min Yoo, and Sang-goo Lee. 2018. Learning to compose task-specific tree structures. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 5094–5101. AAAI Press.

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, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics.

Andrew Drozdov, Patrick Verga, Yi-Pei Chen, Mohit Iyyer, and Andrew McCallum. 2019a. Unsupervised labeled parsing with deep inside-outside recursive autoencoders. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 1507–1512. Association for Computational Linguistics.

Andrew Drozdov, Patrick Verga, Mohit Yadav, Mohit Iyyer, and Andrew McCallum. 2019b. 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, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 1129–1141. Association for Computational Linguistics.

Taeuk Kim, Jihun Choi, Daniel Edmiston, and Sang-goo Lee. 2020. Are pre-trained language models aware of phrases? simple but strong baselines for grammar induction. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Yoon Kim, Chris Dyer, and Alexander M. Rush. 2019. Compound probabilistic context-free grammars for grammar induction. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 2369–2385. Association for Computational Linguistics.

Nikita Kitaev and Dan Klein. 2018. Constituency parsing with a self-attentive encoder. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 2676–2686. Association for Computational Linguistics.

Heeyoung Lee, Angel X. Chang, Yves Peirsman, Nathanael Chambers, Mihai Surdeanu, and Dan Jurafsky. 2013. Deterministic coreference resolution based on entity-centric, precision-ranked rules. *Comput. Linguistics*, 39(4):885–916.

Lemao Liu, Muhua Zhu, and Shuming Shi. 2018. Improving sequence-to-sequence constituency parsing. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 4873–4880. AAAI Press.

Qian Liu, Bei Chen, Jian-Guang Lou, Bin Zhou, and Dongmei Zhang. 2020. Incomplete utterance rewriting as semantic segmentation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 2846–2857. Association for Computational Linguistics.

Thanh-Tung Nguyen, Xuan-Phi Nguyen, Shafiq R. Joty, and Xiaoli Li. 2020. Efficient constituency parsing by pointing. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 3284–3294. Association for Computational Linguistics.

Juri Opitz and Anette Frank. 2021. Towards a decomposable metric for explainable evaluation of text generation from AMR. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume, EACL 2021, Online, April 19 - 23, 2021*, pages 1504–1518. Association for Computational Linguistics.

Letian Peng, Zuchao Li, and Hai Zhao. 2021. A novel metric for evaluating semantics preservation. *CoRR*, abs/2110.01176.

Atul Sahay, Anshul Nasery, Ayush Maheshwari, Ganesh Ramakrishnan, and Rishabh K. Iyer. 2021. Rule augmented unsupervised constituency parsing. In *Findings of the Association for Computational Linguistics: ACL/IJCNLP 2021, Online Event, August 1-6, 2021*, volume ACL/IJCNLP 2021 of *Findings of ACL*, pages 4923–4932. Association for Computational Linguistics.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning, CoNLL 2003, Held in cooperation with*

HLT-NAACL 2003, Edmonton, Canada, May 31 - June 1, 2003, pages 142–147. ACL.

Yikang Shen, Zhouhan Lin, Chin-Wei Huang, and Aaron C. Courville. 2018a. Neural language modeling by jointly learning syntax and lexicon. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

Yikang Shen, Zhouhan Lin, Athul Paul Jacob, Alessandro Sordoni, Aaron C. Courville, and Yoshua Bengio. 2018b. Straight to the tree: Constituency parsing with neural syntactic distance. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 1171–1180. Association for Computational Linguistics.

Yikang Shen, Shawn Tan, Alessandro Sordoni, and Aaron C. Courville. 2019. Ordered neurons: Integrating tree structures into recurrent neural networks. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Yikang Shen, Yi Tay, Che Zheng, Dara Bahri, Donald Metzler, and Aaron C. 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, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pages 7196–7209. Association for Computational Linguistics.

Yau-Shian Wang, Hung-yi Lee, and Yun-Nung Chen. 2019. Tree transformer: Integrating tree structures into self-attention. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 1061–1070. Association for Computational Linguistics.

Adina Williams, Andrew Drozdov, and Samuel R. Bowman. 2018. Do latent tree learning models identify meaningful structure in sentences? *Trans. Assoc. Comput. Linguistics*, 6:253–267.

Jiacheng Xu and Greg Durrett. 2019. Neural extractive text summarization with syntactic compression. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 3290–3301. Association for Computational Linguistics.

Dani Yogatama, Phil Blunsom, Chris Dyer, Edward Grefenstette, and Wang Ling. 2017. Learning to compose words into sentences with reinforcement learning. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020a. Bertscore: Evaluating text generation with BERT. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Yu Zhang, Houquan Zhou, and Zhenghua Li. 2020b. Fast and accurate neural CRF constituency parsing. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*, pages 4046–4053. ijcai.org.

Xiaoshi Zhong, Erik Cambria, and Amir Hussain. 2020. Extracting time expressions and named entities with constituent-based tagging schemes. *Cogn. Comput.*, 12(4):844–862.

A Detailed Configuration

Before we release our codes, you can re-implement the results in our experiments with the configuration setting in this section.

A.1 Mold

W	i	j	l
Influential members of the House Ways and Means Committee introduced legislation that would restrict how the new savings-and-loan bailout agency can raise capital , creating another potential obstacle to the government 's sale of sick thrifts .	16	20	NP†
The complex financing plan in the S&L bailout law includes raising $ 30 billion from debt issued by the newly created RTC .	1	4	NP
Another $ 20 billion would be raised through Treasury bonds , which pay lower interest rates .	5	16	VP†
The bill intends to restrict the RTC to Treasury borrowings only , unless the agency receives specific congressional authorization .	3	19	VP
The complex financing plan in the S&L bailout law includes raising $ 30 billion from debt issued by the newly created RTC .	17	22	VP
But the RTC also requires " working " capital to maintain the bad assets of thrifts that are sold , until the assets can be sold separately .	10	27	VP
" Such agency ' self-help ' borrowing is unauthorized and expensive , far more expensive than direct Treasury borrowing , " said Rep. Fortney Stark -LRB- D. , Calif. -RRB- , the bill 's chief sponsor .	9	11	ADJP†
" Such agency ' self-help ' borrowing is unauthorized and expensive , far more expensive than direct Treasury borrowing , " said Rep. Fortney Stark -LRB- D. , Calif. -RRB- , the bill 's chief sponsor .	13	15	ADJP
" To maintain that dialogue is absolutely crucial .	7	8	ADJP
Many money managers and some traders had already left their offices early Friday afternoon on a warm autumn day – because the stock market was so quiet .	8	8	ADVP†
This country is fairly big .	4	4	ADVP
Therefore , we can exchange in the market .	1	1	ADVP
" To maintain that dialogue is absolutely crucial .	7	8	ADVP
Once again -LCB- the specialists -RCB- were not able to handle the imbalances on the floor of the New York Stock Exchange , " said Christopher Pedersen , senior vice president at Twenty-First Securities Corp .	14	22	PP†
Big investment banks refused to step up to the plate to support the beleaguered floor traders by buying big blocks of stock , traders say .	17	22	PP
Just days after the 1987 crash , major brokerage firms rushed out ads to calm investors .	1	6	PP

Table 5: Molds for result reproduction (from NP to PP). †: Used for UTL

W	i	j	l
That debt would be paid off as the assets are sold , leaving the total spending for the bailout at $ 50 billion , or $ 166 billion including interest over 10 years .	21	23	QP†
" We would have to wait until we have collected on those assets before we can move forward , " he said .	7	13	SBAR†
Instead , it settled on just urging the clients who are its lifeline to keep that money in the market .	10	13	SBAR
Influential members of the House Ways and Means Committee introduced legislation that would restrict how the new savings-and-loan bailout agency can raise capital , creating another potential obstacle to the government 's sale of sick thrifts .	16	23	S†
Another $ 20 billion would be raised through Treasury bonds , which pay lower interest rates .	12	12	WHNP†
But the RTC also requires " working " capital to maintain the bad assets of thrifts that are sold , until the assets can be sold separately .	16	17	WHNP
Prices in Brussels , where a computer breakdown disrupted trading , also tumbled .	5	5	WHADVP†
Dresdner Bank last month said it hoped to raise 1.2 billion marks -LRB- $ 642.2 million -RRB- by issuing four million shares at 300 marks each .	13	17	PRN†
Today 's Fidelity ad goes a step further , encouraging investors to stay in the market or even to plunge in with Fidelity .	14	14	PRT†

Table 6: Molds for result reproduction (the rest). †: Used for UTL

How do we choose the molds? Table 5 and 6 shows the molds we use for discerning constituents in LSG and labeling in UTL. The molds are handcrafted or selected from the first 20 sentence (contained such a labeled constituent) in the development dataset. To guarantee the quality of molds, we test them on UTL framework to label constituents and selected the molds perform well on classification evaluated by AUC (> 0.85). The most well-performed molds are preserved for constituent labeling in UTL.

A.2 POS Constraint

Label	POS($V[i]$)	POS($V[j]$)	POS($V[i-1]$)	POS($V[j+1]$)	Max Len
NP	DET PROPN NOUN ADJ PRON NUM SYM	NOUN PROPN PRON NUM PART	ADP VERB PUNCT SOS SCONJ CCONJ AUX NOUN ADV NUM PART DET ADJ PROPN PRON	PUNCT ADP VERB AUX CCONJ ADV NOUN DET PART ADJ SCONJ PROPN PRON NUM	-
VP	VERB AUX PART ADV	NOUN VERB PROPN NUM ADV ADJ	NOUN PRON PART AUX VERB PROPN PUNCT ADV DET CCONJ	PUNCT CCONJ PROPN AUX	-
ADJP	ADJ ADV NUM SYM	ADJ NOUN VERB NUM PROPN	AUX DET VERB NOUN PUNCT ADP PART CCONJ	PUNCT NOUN ADP SCONJ CCONJ PROPN	-
ADVP	ADV	ADV	-	-	-
PP	ADP	NOUN PROPN NUM	NOUN VERB PUNCT SOS ADJ PROPN NUM ADV	PUNCT ADP VERB AUX CCONJ SCONJ	-
QP	SYM ADV NUM	NUM	ADP VERB SOS PUNCT AUX DET	NOUN PUNCT ADP ADJ	5
SBAR	SCONJ DET PRON ADV PART ADP	NOUN VERB PROPN NUM	VERB NOUN PUNCT	PUNCT	-
S	PART DET PRON VERB PROPN	NOUN VERB PROPN	VERB SCONJ PUNCT NOUN DET SOS ADP ADV CCONJ PRON	PUNCT	-
WHNP	DET PRON	DET PRON	PUNCT NOUN	VERB AUX	-
WHADVP	ADV	ADV	PUNCT NOUN SOS VERB ADP AUX	DET NOUN PRON PROPN ADJ VERB	-
PRN	PUNCT	PUNCT	NOUN PROPN ADJ VERB PUNCT ADV	PUNCT VERB AUX SCONJ NOUN ADP DET CCONJ	-
PRT	ADP	ADP	VERB	DET PUNCT ADP NOUN ADV ADJ PART CCONJ NUM	1

Table 7: POS and length constraints for result reproduction. **SOS:** Start of the sentence. **EOS:** End of the sentence. -: No constraint.

Table 7 shows our constraints for POS and max length. These constraints are inducted by statistical and constituency property.

Why do we need the POS constraints? As the annotation of constituents are very different from the actual semantic roles of them, we need extra rules to filter some spans that satisfy the semantic property of constituent but are ignored by the annotation. For instance, *John* and *Smith* in *John Smith* all appear to be a noun phrase and they exactly can play the role as a noun phrase. However, only *John Smith* will be annotated as a noun phrase. The POSes are predicted by taggers and thus are **not golden**.

How do we design the POS constraints? We do this in a simple way: we count the proportion of POS in certain positions of spans (POS($V[i]$), POS($V[j]$), POS($V[i-1]$), POS($V[j+1]$)) and remove POS of which appearance frequency is under a certain threshold, i.e., 1%.

A.3 Threshold and Tolerance

Label	Threshold (t)	Tolerance (t)	Threshold (l)	Tolerance (l)
NP	2.0	0.15	1.4	0.10
VP	0.8	0.15	2.0	0.05
ADJP	0.2	0.04	0.6	0.10
ADVP	0.8	0.03	0.8	0.03
PP	0.2	0.10	0.4	0.12
QP	0.2	0.03	0.2	0.03
SBAR	0.2	0.01	2.2	0.10
S	0.2	0.10	2.0	0.15
WHNP	1.0	0.10	1.0	0.10
WHADVP	1.0	0.10	1.0	0.10
PRN	1.0	0.10	1.0	0.10
PRT	1.0	0.10	1.0	0.10

Table 8: Thresholds and tolerances for result reproduction. **t**: Tight configuration. **l**: Loose configuration.

How do we choose the hyperparameter setting? We search the best hyperparameter on the development dataset to optimize unlabeled F1 score (Loose) and labeled F1 score (Tight) and then apply them to the test dataset.

B Distributions of NDD caused by Different Substitutions

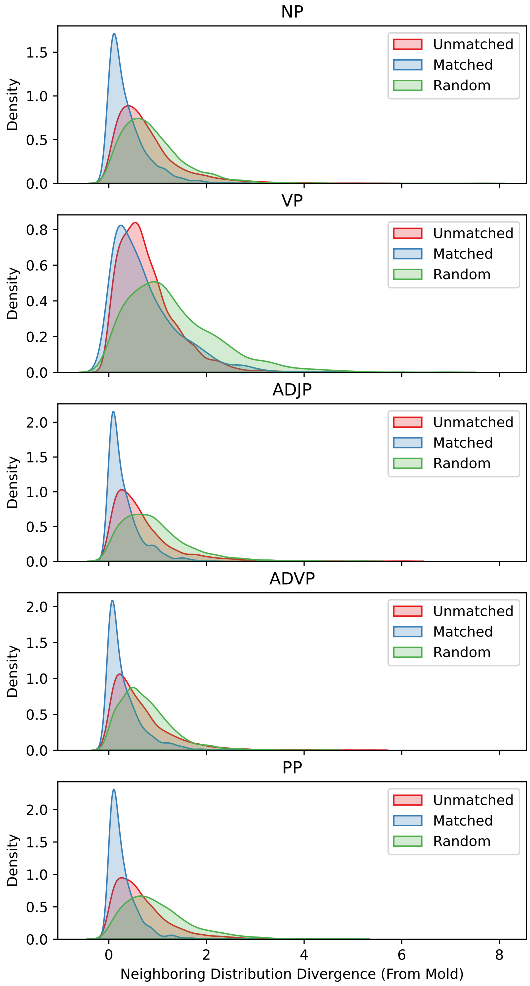

Figure 8: Distributions of from-mold POS-NDD.

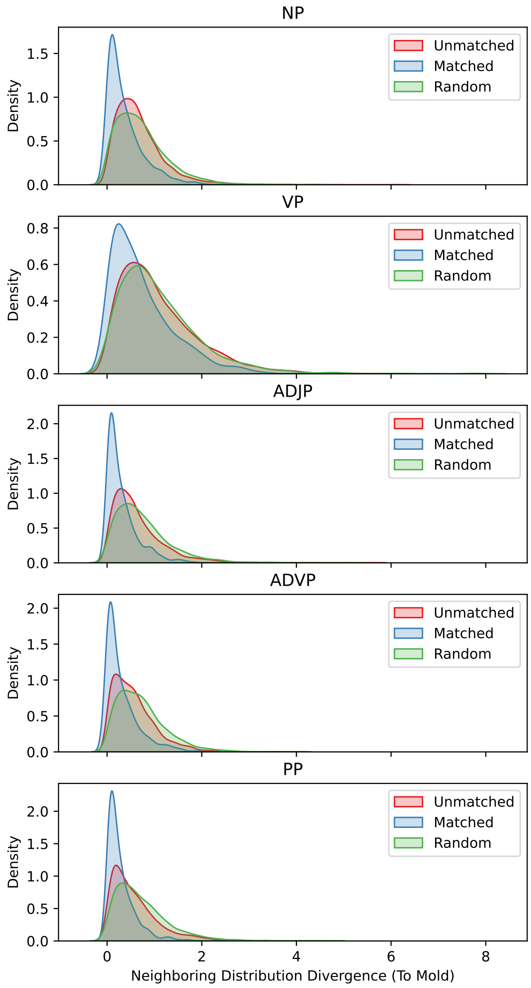

Figure 9: Distributions of to-mold POS-NDD.

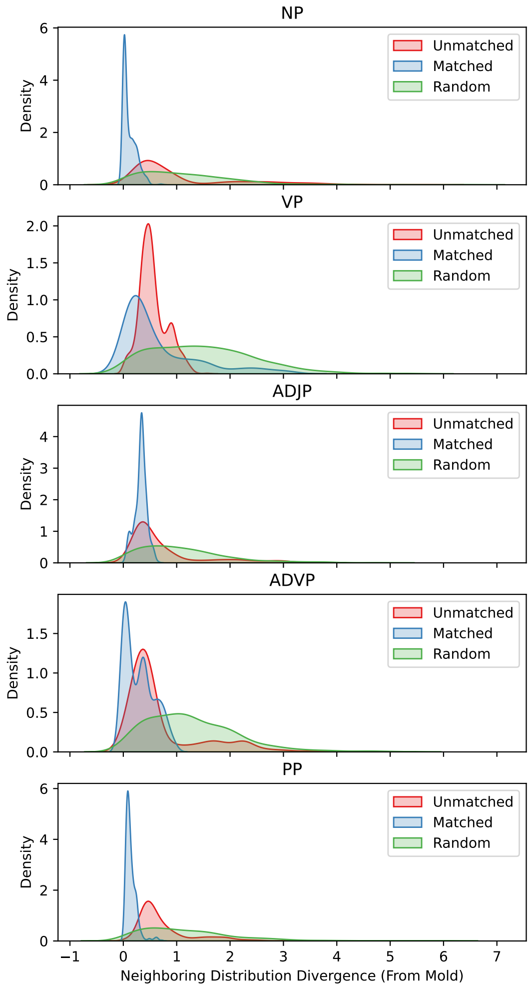

Figure 10: Distributions of from-mold POS-ND of selected molds for UTLD.

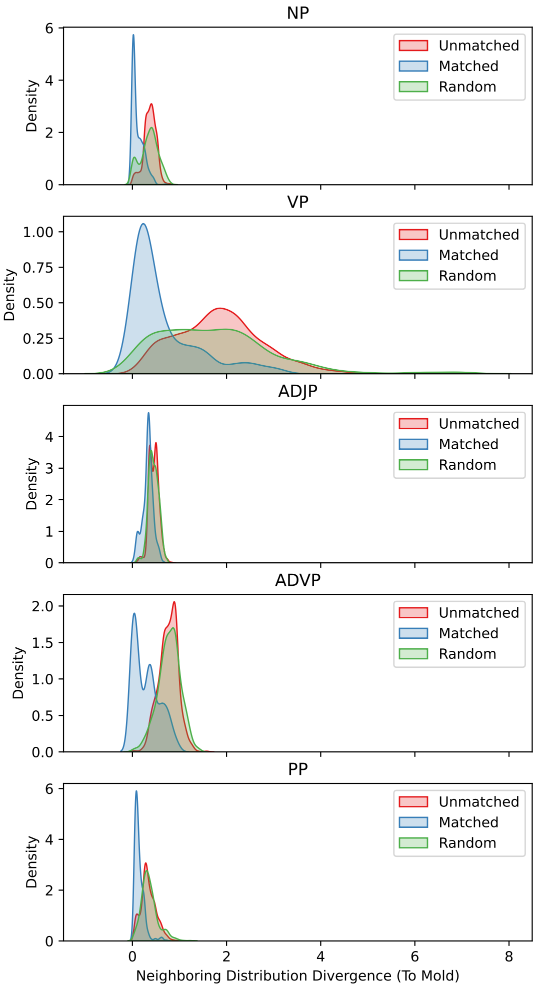

Figure 11: Distributions of to-mold POS-NDD of selected molds for UTL.

C Analysis for Disturbance Caused by Substitution

Figure 12, 13 and 14 show the three cases of disturbance on neighboring prediction distributions caused by substituting operations. These operations substitute the span *The cat* in *The cat jumps into the hole* by *The crow*, *My house* and *In London*. We use the five candidates with the highest existence probability in the initial sentence to show the changes on each word's prediction.

D Parsing Cases

Parsing cases are enumerated in this section.

Will decoding algorithms like CKY improve parsing performance? No, in our experiments, applying CKY actually results in a drop of > 10 in unlabeled F1 score.

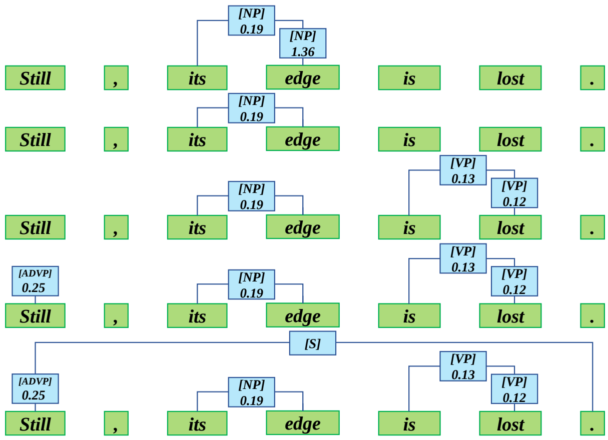

Figure 15: LSG Parsing Case (LP= 100.0, LR= 100.0, LF1= 100.0).

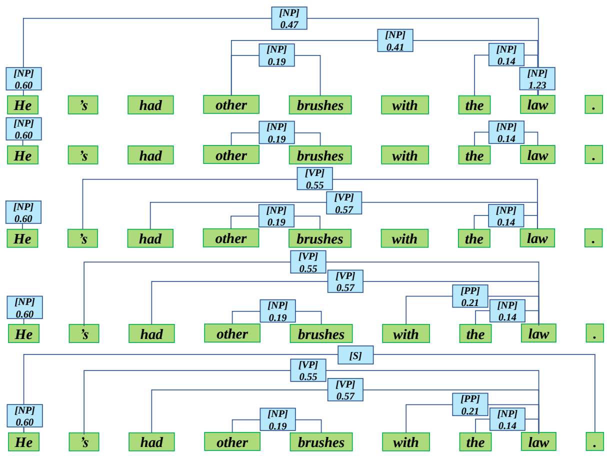

Figure 16: LSG Parsing Case (LP= 100.0, LR= 87.5, LF1= 93.3, *other brushes with the law* missed).

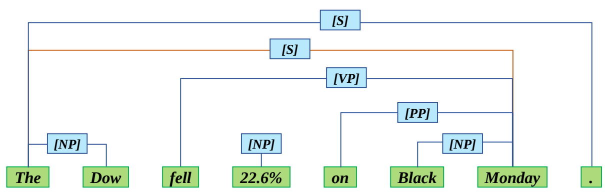

Figure 17: UTL Parsing Case (LP= 85.7, LR= 100.0, LF1= 92.3, Labeling Acc.= 100.0, The **red edge** refers to the fault in unlabeled tree from DIORA+PP).

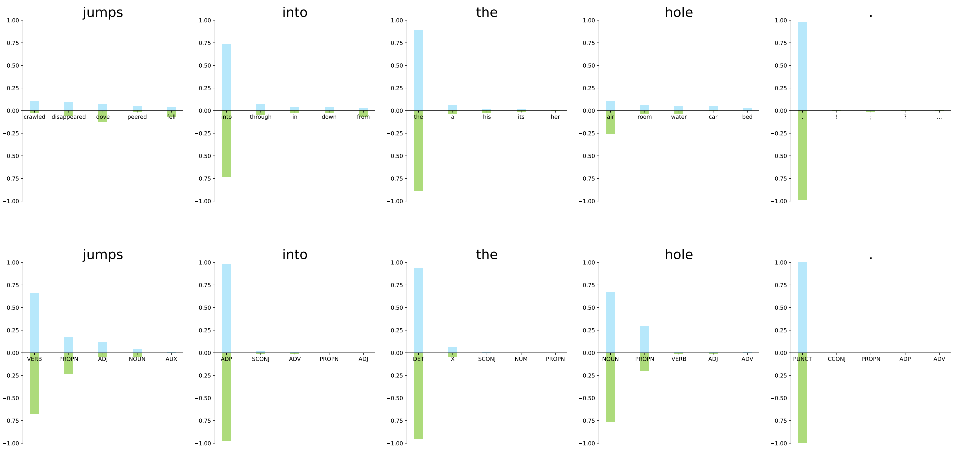

Figure 12: Prediction distribution disturbance (**Upper:** POS-NDD, **Lower:** NDD) (**Blue:** Before substitution, **Green:** After substitution) caused by constituent substitution (From ***The crow*** to *The cat* in sentence *The cat jumps into the hole.*).

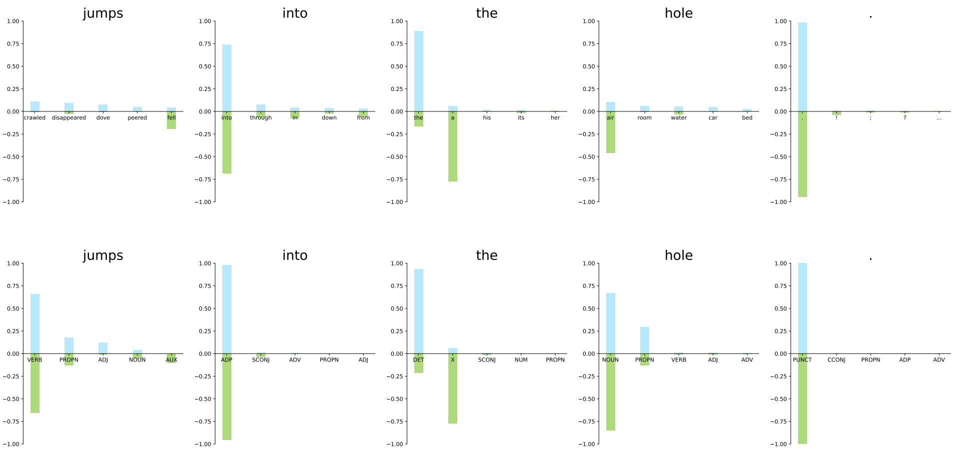

Figure 13: Prediction distribution disturbance (**Upper:** POS-NDD, **Lower:** NDD) (**Blue:** Before substitution, **Green:** After substitution) caused by constituent substitution (From ***My house*** to *The cat* in sentence *The cat jumps into the hole.*).

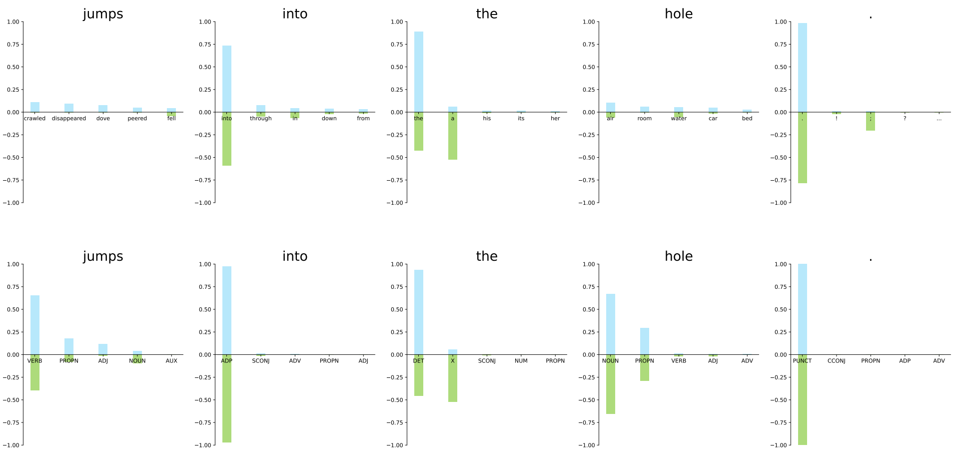

Figure 14: Prediction distribution disturbance (**Upper:** POS-NDD, **Lower:** NDD) (**Blue:** Before substitution, **Green:** After substitution) caused by constituent substitution (From ***In London*** to *The cat* in sentence *The cat jumps into the hole.*).