Revisiting the Practical Effectiveness of Constituency Parse Extraction from Pre-trained Language Models

Taeuk Kim
Dept. of Computer Science & Dept. of Artificial Intelligence
Hanyang University, Seoul, South Korea
kimtaeuk@hanyang.ac.kr

Abstract

Constituency Parse Extraction from Pre-trained Language Models (CPE-PLM) is a recent paradigm that attempts to induce constituency parse trees relying only on the internal knowledge of pre-trained language models. While attractive in the perspective that similar to in-context learning, it does not require task-specific fine-tuning, the practical effectiveness of such an approach still remains unclear, except that it can function as a probe for investigating language models' inner workings. In this work, we mathematically reformulate CPE-PLM and propose two advanced ensemble methods tailored for it, demonstrating that the new parsing paradigm can be competitive with common unsupervised parsers by introducing a set of heterogeneous PLMs combined using our techniques. Furthermore, we explore some scenarios where the trees generated by CPE-PLM are practically useful. Specifically, we show that CPE-PLM is more effective than typical supervised parsers in few-shot settings.

1 Introduction

With the increasing interest in the inner workings of pre-trained language models (PLMs; Devlin et al. (2019); Liu et al. (2019); Radford et al. (2019); Conneau et al. (2020)),[1] much work that attempts to explore the inherent knowledge embedded in the models has been recently proposed. One of the main topics in this direction is to reveal whether PLMs understand syntactic knowledge of human language, usually represented as parse trees. While a line of work (Hewitt and Manning (2019); Chi et al. (2020)) has investigated the existence of syntax in PLMs via structural probes with supervision from gold-standard parse trees, some studies (Kim et al., 2020, 2021; Wu et al., 2020) have found that one can extract reasonable parse structures directly from the patterns presented in PLMs' hidden representations or attention distributions *even without extra fine-tuning*. In other words, the studies have shown that PLMs implicitly store their understanding of syntactic knowledge in their parameters, and that such information can be easily reformulated into syntactic trees with almost no additional cost.

Although the aforementioned approach, dubbed **C**onstituency **P**arse **E**xtraction from **P**re-trained **L**anguage **M**odels (**CPE-PLM**; Kim et al. (2021)), is undoubtedly a useful tool with many analytic uses (Rogers et al., 2020), it still remains a research question whether this algorithm can also work for practical purposes. For instance, as CPE-PLM is free from fine-tuning of PLMs, it may be appealing in few-shot settings, akin to in-context learning for natural language understanding (Brown et al., 2020). Moreover, there exists a potential that the new parsing paradigm can substitute the role of supervised or unsupervised parsers for the case where an NLP model requires a parse tree as input.

In this work, we focus on revealing the practical effectiveness of CPE-PLM. Specifically, we first rewrite the procedure of CPE-PLM in a more rigorous form to clarify its working mechanism. We then introduce two new ensemble algorithms tailored for it, i.e., `Greedy` and `Beam`, making its parsing performance competitive with that of unsupervised parsers (Kim et al., 2019b; Zhu et al., 2020). We show that it is crucial to combine syntactic clues from heterogeneous PLMs for achieving comparable performance, and that this trend holds in not only English but also multilingual cases.

Equipped with the improved variants of CPE-PLM, as the next step, we investigate some scenarios in which their outputs (i.e., generated trees) can be practically utilized. We show that (1) it is viable to introduce the trees from CPE-PLM as auxiliary data for improving Recurrent Neural Network Grammars (RNNG; Dyer et al. (2016); Kim

[1] We use the term *pre-trained language models (PLMs)* to refer to the models that are based on Transformer (Vaswani et al., 2017) and pre-trained in a self-supervised manner, e.g., BERT (Devlin et al., 2019) and its variants.

Proceedings of the 29th International Conference on Computational Linguistics, pages 5398–5408
October 12–17, 2022.

et al. (2019c)), that (2) on classification with Tree LSTMs (Tai et al., 2015), the induced trees can replace gold-standard parses with a minimal loss, and that (3) CPE-PLM can be even more data-efficient than supervised parsers in few-shot settings.

2 Background and Related Work

2.1 Constituency Parse Extraction from Pre-trained Language Models

The term "**C**onstituency **P**arse **E**xtraction from **P**re-trained **L**anguage **M**odels (**CPE-PLM**)" coined by Kim et al. (2021) represents a range of parsing methods (Mareček and Rosa, 2019; Rosa and Mareček, 2019; Kim et al., 2020, 2021; Li et al., 2020) that aim to infer the parse tree of an input sentence by only exploiting the features obtained from PLMs. In detail, the approaches belonging to this paradigm attempt to directly (i.e., without training) apply simple heuristics or existing parsing algorithms, such as top-down (Shen et al., 2018a) and chart-based (Kitaev and Klein, 2018) ones, on the hidden representations or attention maps retrieved from PLMs. In the following, we illustrate the exact formulation of some representative methods.

As CPE-PLM does not demand more than *frozen* PLMs as its ingredient, which means *training-free*, it can be particularly useful when there are no resources available for training supervised parsers in terms of either (1) computing resources or (2) training data consisting of gold-standard annotations. However, its use has been limited to analytic purposes in the literature, utilized as a tool for probing the inner workings of PLMs. Our goal in this paper is therefore to investigate the utility of CPE-PLM in practical scenarios. Among several options, we select the chart-based variant (Kim et al., 2021) as our baseline, which generally outperforms others.

Chart-based CPE-PLM. Kim et al. (2021) propose a method that combines PLMs with the chart parsing algorithm without extra training. Formally, each tree candidate T for an input sentence, $w_1, w_2, \ldots, w_z$, is assigned a score $s_{tree}(T)$ that decomposes as $s_{tree}(T) = \sum_{(i,j)\in T} s_{span}(i,j)$, where $s_{span}(i,j)$ is a score for a constituent that is located between positions i and j in the sentence. $s_{span}(i,j)$ is defined as follows:

$$s_{span}(i,j) = \begin{cases} s_{comp}(i,j) + \min_{i\le k<j} s_{split}(i,k,j) & \text{if } i<j \\ 0 & \text{if } i=j \end{cases},$$

where $s_{split}(i,k,j) = s_{span}(i,k) + s_{span}(k+1,j)$. In other words, $s_{comp}(i,j)$ measures the compositionality of the span (i,j) itself while $s_{split}(i,k,j)$ indicates how plausible it is to divide the span (i,j) into two subspans (i,k) and $(k+1,j)$. Note that every $s_{span}(i,j)$ can be easily calculated in a bottom-up fashion with the aid of the CKY algorithm (Cocke, 1969; Kasami, 1966; Younger, 1967), once $s_{comp}(i,j)$ is properly defined.

Although the authors suggest two derivations for $s_{span}(i,j)$, in this work, the pair score function $s_p(\cdot,\cdot)$ is chosen as the main target of our interest, which is defined as follows:

$$s_p(i,j) := \binom{j-i+1}{2}^{-1} \sum_{(w_x,w_y)\in pair(i,j)} f(g(w_x), g(w_y)),$$

where $pair(i,j)$ returns a set consisting of all combinations of two words from a span (i,j), e.g., $pair(1,3) = \{(w_1,w_2),(w_1,w_3),(w_2,w_3)\}$, while $f(\cdot,\cdot)$ and $g(\cdot)$ are a distance measure function and representation extractor function respectively. To realize $f(\cdot,\cdot)$ and $g(\cdot)$, the authors consider two sets of functions, F and G. Given l as the number of layers in a PLM and a as the number of attention heads per layer, G refers to the set of functions $\{g_{(m,n)} | m = 1,\ldots,l, n = 1,\ldots,a\}$, each of which outputs the attention distribution of an input word computed by the n^{th} attention head on the m^{th} layer of the PLM. F is specified as $\{\texttt{JSD}, \texttt{HEL}\}$, where JSD and HEL correspond to the Jensen-Shannon and Hellinger distance. Here, HEL is only considered for simplicity.

Intuitively, a series of the operations described so far can be understood as (1) splitting an attention map by rows (i.e., into attention distributions) for representing each word and (2) comparing similarities between the rows to gauge the syntactic proximity of the corresponding words. Finally, chart-based CPE-PLM outputs $\hat{T}$, the tree that requires the lowest cost to build, as a prediction for the parse tree of the input sentence: $\hat{T} = \arg\min_T s_{tree}(T)$.

2.2 Ensemble Methods for CPE-PLM

In Section 2.1, we merely mentioned the case where only an element of G is adopted for computing the scores used in CPE-PLM. However, it is also feasible to employ more than just one, such that the method is provided with diverse information from different attention maps (i.e., $g_{(m,n)}$) to derive more reasonable parse trees. In fact, the previous work on CPE-PLM (Kim et al., 2020, 2021) already exploits such *ensemble* strategies in an *ambiguous* and *implicit* manner to boost its performance. As the introduction of those techniques can

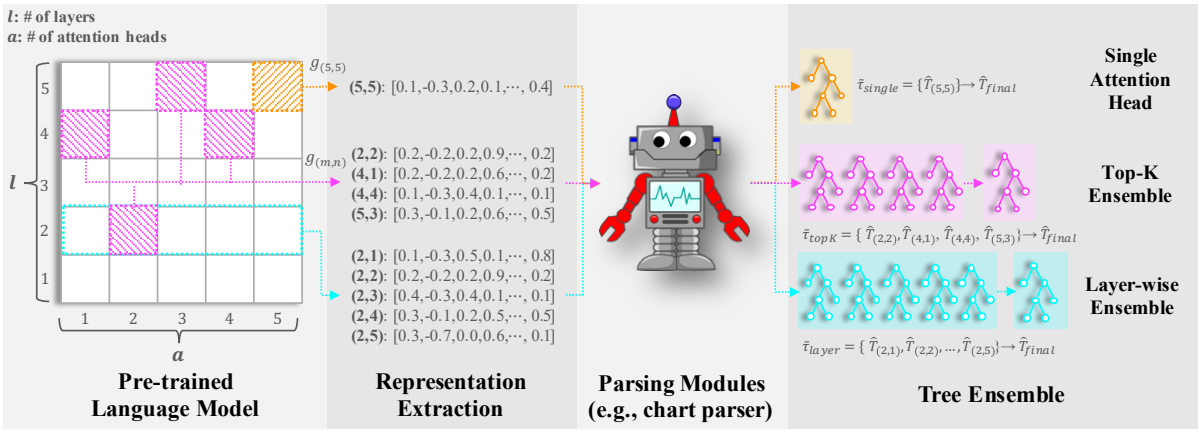

Figure 1: Concept diagram explaining the procedure of CPE-PLM with various ensemble methods. Given a PLM that has $l = 5$ Transformer layers, each of whose self-attention module consists of $a = 5$ individual attention heads, an input sentence is inserted into the PLM to compute the model's attention maps. Then, we construct parse candidates using the information from each $g_{(m,m)}$ and the CKY algorithm. Here, the role of ensemble methods is to determine which trees to be engaged in the final prediction of the resulting parse tree.

lead to a considerable gap in the final result, we claim that it is essential to clarify which ensemble method is employed and to develop more advanced ones. Accordingly, we here *explicitly* formulate the previous ensemble methods, which is one of our contributions in this work.

Figure 1 explains the process of applying ensemble methods to CPE-PLM. Formally, let $\tau := \{\hat{T}_{(m,n)} | m = 1, \ldots, l, n = 1, \ldots, a\}$ denote a pool of all the possible tree predictions computed with CPE-PLM using every $g_{(m,n)}$. The objective of the ensemble methods is to derive the best parse prediction from a subset of τ, denoted $\widetilde{\tau}$, so that it closely resembles the corresponding gold-standard tree. Once $\widetilde{\tau}$ is decided, every $\hat{T}_{(m,n)}$ from $\widetilde{\tau}$ is converted into the form of *syntactic distance* (Shen et al., 2018a) $\mathbf{d}_{(m,n)} \in \mathbb{R}^{z-1}$. [2] Then, the resulting vectors are averaged to derive $\mathbf{d}_{final}$, which is finally restored to the tree form $\hat{T}_{final}$. [3] In the following, we illustrate the characteristics of each ensemble technique shown when determining $\widetilde{\tau}$.

Naïve Baseline: Single Attention Head. The simplest way of implementing CPE-PLM is to utilize just a single attention head as a representative. To be specific, this baseline constructs $\widetilde{\tau}_{single} := \{\hat{T}_{(m^*,n^*)} | \forall m \forall n, val(g_{(m,n)}) \leq val(g_{(m^*,n^*)})\}$, where $val(g_{(m,n)})$ indicates the performance of CPE-PLM on the *validation* set, given $g_{(m,n)}$. In other words, it directly outputs the parse $\hat{T}_{(m^*,n^*)}$ generated by the best function $g_{(m^*,n^*)}$.

Layer-wise Ensemble. Kim et al. (2020) suggest to merge a group of trees that originated from the attention heads located in the same layer of a PLM. Specifically, this ensemble method defines $\widetilde{\tau}^m_{layer} := \{\hat{T}_{(m,n)} | n = 1, \ldots, a\}$ to consider layer-specific information. The best layer of the PLM (i.e., m^*) is also determined by its performance on the validation set. The intuition behind this heuristic is that attention heads of the same layer might be complementary cooperative in grasping a linguistic concept, considering that particular layers of a PLM seem specialized to capture a specific aspect of linguistic knowledge (Tenney et al., 2019; Jawahar et al., 2019; Jo and Myaeng, 2020).

Top-K Ensemble. Kim et al. (2021) propose utilizing the top-K $g(\cdot)$ functions instead of only using the best, $g_{(m^*,n^*)}$. First, a sorted set $\tau_{sorted} := \{\hat{T}_{(m^i,n^i)} | i, j \in \{1, \ldots, l \times a\} \cap val(g_{(m^i,n^i)}) \geq val(g_{(m^j,n^j)})$ whenever $i \leq j\}$ is specified as a variant of the set τ. That is, τ_{sorted} is identical to τ except that the elements of τ_{sorted} are arranged in descending order according to the validation performance of their corresponding functions $\{g_{(m^i,n^i)}\}$. Then, the top-K ensemble method defines $\widetilde{\tau}_{topK}$ as the set consisting of the first K elements of τ_{sorted}.

3 Proposed Methods

In this section, we additionally introduce two novel ensemble methods for CPE-PLM, which are more effective than the previous counterparts in improv-

[2] Note that z is the number of words in the input sentence.

[3] Refer to Kim et al. (2021) for the exact procedure of converting a tree into a syntactic distance and vice versa.

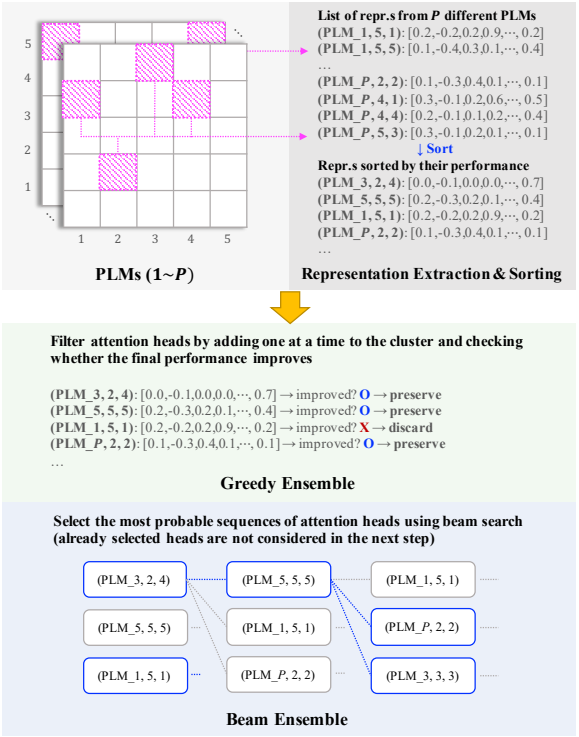

Figure 2: Concept diagram explaining the operation of `Greedy` and `Beam` in multi-PLM environments.

ing parsing performance. Furthermore, we propose to allow several PLMs to collaborate with each other to provide CPE-PLM with more diverse syntactic information (Figure 2).

Greedy Ensemble. We first consider a method that collects every helpful attention head in a greedy fashion. Unlike the previous ensemble methods which allow only a fixed number of attention heads to be engaged in the ensemble process, this sets no limit on the number of participants (i.e., attention heads), diversifying the source of syntactic clues.

Sticking to the notations defined in Section 2.2, we specify the `Greedy` ensemble algorithm in Algorithm 1. Its core logic is to append one attention head at a time to the cluster and test whether each augmentation is beneficial for making progress in the final (validation) performance ($val(G_{greedy})$).

Beam Ensemble. Even though the greedy ensemble method is simple and effective, there still exists a need for exploring more diverse groups of attention heads that have the potential to show better performance than the group chosen by the greedy algorithm. To this end, inspired by the beam search algorithm widely adopted in the natural language generation (NLG) literature, we introduce the `Beam` ensemble method in Algorithm 2.

Algorithm 1 Greedy Ensemble Algorithm

$G_{sorted} := \{g_{(m^i,n^i)} | i,j \in \{1,\dots,l\times a\} \cap val(g_{(m^i,n^i)}) \geq val(g_{(m^j,n^j)})$ whenever $i \leq j\}$
$g_{(m^i,n^i)} :=$ The i^{th} element of G_{sorted}
$\hat{T}_{(m^i,n^i)} :=$ A tree prediction generated using $g_{(m^i,n^i)}$
function GREEDY(G_{sorted})
  $G_{greedy}, \widetilde{\tau}_{greedy}, \mu \leftarrow \{\}, \{\}, 0$
  for $i = 1,\dots,l \times a$ **do**
    $G_{greedy} \leftarrow G_{greedy} \cup \{g_{(m^i,n^i)}\}$
    $\psi \leftarrow val(G_{greedy})$
    if $\psi > \mu$ **then**
      $\mu \leftarrow \psi$
      $\widetilde{\tau}_{greedy} \leftarrow \widetilde{\tau}_{greedy} \cup \{\hat{T}_{(m^i,n^i)}\}$
    else
      $G_{greedy} \leftarrow G_{greedy} \backslash \{g_{(m^i,n^i)}\}$
    end if
  end for
  return $\widetilde{\tau}_{greedy}$
end function

Algorithm 2 Beam Ensemble Algorithm

$b :=$ beam size
$G_{sorted} := \{g_{(m^i,n^i)} | i,j \in \{1,\dots,l\times a\} \cap val(g_{(m^i,n^i)}) \geq val(g_{(m^j,n^j)})$ whenever $i \leq j\}$
$g_{(m^i,n^i)} :=$ The i^{th} element of G_{sorted}
function BEAM(G_{sorted}, b)
  $G_{beam}, e \leftarrow \{\}, 0$
  for $i = 1,\dots,b$ **do**
    $G_{beam} \leftarrow G_{beam} \cup \{\{g_{(m^i,n^i)}\}\}$
  end for
  while $e < b$ **do**
    $\hat{G}, \psi, e \leftarrow \{\}, \{\}, 0$
    for each $H \in G_{beam}$ **do**
      $\mu \leftarrow$ The largest index j given $\forall g_{(m^j,n^j)} \in H$.
      for $i = 1,\dots,b$ **do**
        if $\mu + i > l \times a$ **then**
          $e \leftarrow e + 1$
          break
        else
          $H_i \leftarrow H \cup \{g_{(m^{\mu+i},n^{\mu+i})}\}$
          $\hat{G} \leftarrow \hat{G} \cup \{H_i\}$
          $\psi \leftarrow \psi \cup \{val(H_i)\}$
        end if
      end for
    end for
    $\psi_{topB} \leftarrow$ The set consisting of the top b elements of ψ.
    $G_{beam} \leftarrow \{H | H \in \hat{G} \cap val(H) \in \psi_{topB}\}$
  end while
  $G^*_{beam} \leftarrow \{H^* | \forall H \in G_{beam}, val(H^*) \geq val(H)\}$
  $\widetilde{\tau}_{beam} \leftarrow \{\hat{T}_{(m^i,n^i)}) | \hat{T}_{(m^i,n^i)}$ is predicted using $g_{(m^i,n^i)} \in H^*$ (H^* is the sole element of G^*_{beam})$\}$
  return $\widetilde{\tau}_{beam}$
end function

Our beam ensemble algorithm is similar to one for NLG (Graves, 2012; Sutskever et al., 2014) except that the beam search procedure is not stochastic, but determined by the order of the elements of G_{sorted}. Furthermore, the already selected attention heads are not considered in the next search step, unlike NLG which allows the same word to be generated twice or more (see Figure 2 for example).

The merit of Beam is that it can explore a wider range of potential paths that might not be covered by the greedy algorithm.

Extension to Multi-PLM Settings. Until now, we have assumed that CPE-PLM is only applicable for only a single PLM. However, we propose for the first time extending its usage to the scenario in which multiple PLMs are available together. In other words, we expand τ to $\tau_{multi} := \{\hat{T}_{(p,m,n)} | p \in \{1,\ldots,P\}, m \in \{1,\ldots,l\}, n \in \{1,\ldots,a\}\}$, where P is the number of the PLMs involved. By doing so, it is expected that CPE-PLM can infer more accurate parse trees with the aid of diverse perspectives from heterogeneous PLMs. We show in Section 4.2 that this simple and intuitive extension leads to a significant improvement in the final performance. It also has value in that it is one of the initial attempts in the literature to leverage different PLMs simultaneously.

4 Experiments on Parsing

In this chapter, we aim to validate the effectiveness of the proposed ensemble algorithms, i.e., Greedy and Beam, which enable CLE-PLM to have a higher potential of being properly adopted for downstream tasks.

4.1 General Configurations

Datasets. To evaluate the parsing performance of CPE-PLM and other related models, we utilize the Penn Treebank dataset (PTB, Marcus et al. (1993)) for English and the SPMRL (Seddah et al., 2013) dataset for eight other languages, following Kim et al. (2021).[4] We also adhere to the standard of the previous work for preprocessing the datasets.

Evaluation Metrics. We utilize the *unlabeled* sentence-level $F1$ score as a main metric to evaluate the extent to which induced trees resemble corresponding gold-standard trees. It was originally introduced by Shen et al. (2018b, 2019), becoming the de-facto standard in unsupervised parsing.

Model Selection & Hyperparameters. As mentioned in Section 2.1, we build our approach upon chart-based CPE-PLM (Kim et al., 2021). Moreover, we employ twelve English PLMs and four multilingual PLMs to provide syntactic information.[5] To handle various PLMs in an integrated manner, we use the Transformers library developed by HuggingFace (Wolf et al., 2019). We determine hyperparameters for the Top-K and Beam ensemble methods using grid search. In consequence, we use K=20 and b=5 for single PLM cases and K=30 and b=30 in multi-PLM settings.

PLMs / Methods	Previous work		Chart CPE-PLM w/ ensemble methods				
	Top-down†	Chart‡	Single	Layer	Top-K	Greedy	Beam
Encoder-based							
BERT-base	32.4	42.7	34.1	35.3	42.5	43.0	42.8
BERT-large	34.2	44.2	38.7	40.6	44.4	45.0	44.5
RoBERTa-base	33.8	44.9	40.9	39.2	44.2	45.4	45.4
RoBERTa-large	34.1	41.9	39.5	38.9	44.9	47.2	43.7
ELECTRA-base	-	-	40.2	41.2	43.3	46.9	43.2
ELECTRA-large	-	-	**44.3**	41.3	46.6	47.9	47.2
Decoder-based							
GPT2	37.1	37.2	34.5	26.4	36.9	37.2	37.1
GPT2-medium	39.4	38.4	38.0	28.2	38.2	38.0	40.8
CTRL	-	-	35.7	28.7	44.4	45.8	44.9
Hybrid							
BART-large	-	-	37.5	32.6	39.8	37.5	38.5
XLNet-base	**40.1**	46.4	36.7	39.7	46.0	47.0	46.7
XLNet-large	38.1	46.4	39.5	38.9	45.7	47.2	46.8
Multilingual							
MBERT	-	45.0	39.0	40.3	44.6	47.1	45.7
XLM	-	**47.7**	41.9	42.1	47.1	47.5	47.1
XLM-R	-	46.7	41.6	**44.2**	46.5	48.5	47.4
XLM-R-large	-	44.6	40.7	36.7	44.3	46.8	46.9
Multiple PLMs							
Only multilingual	-	-	-	-	49.6	51.9	49.8
All models	-	-	-	-	**50.4**	**55.3**	**55.7**

Table 1: $F1$ scores of CPE-PLM on the PTB test set conditioned on the different combinations of PLMs and ensemble methods. We show that Greedy and Beam are more effective than baselines, and that we attain much more competitive scores in multiple PLM settings. The best score for each column is in **bold**. We also report numbers from two previous studies for reference. †: From Kim et al. (2020). ‡: From Kim et al. (2021).

4.2 Verification of CPE-PLM's Performance

We first conduct experiments on the English PTB dataset using CPE-PLM, with the objective of comparing the effects of different PLMs and ensemble methods on the paradigm. We also test the settings in which multiple PLMs are employed at the same time. From Table 1, we confirm that our Greedy and Beam algorithms are more effective than other techniques in most cases and that their impact is amplified when combined with multiple PLMs. As a result, we succeed in achieving the state-of-the-

[4] We use national codes to represent languages, i.e., en: English, eu: Basque, fr: French, de: German, he: Hebrew, hu: Hungarian, ko: Korean, pl: Polish, and sv: Swedish.

[5] The list of PLMs we use is (1) English PLMs: BERT-base/large (Devlin et al., 2019), RoBERTa-base/large (Liu et al., 2019), ELECTRA-base/large (Clark et al., 2020), GPT2(-medium) (Radford et al., 2019), CTRL (Keskar et al., 2019), BART (Lewis et al., 2020), XLNet (Yang et al., 2019), (2) multilingual PLMs: MBERT, XLM (Conneau and Lample, 2019), XLM-R(-large) (Conneau et al., 2020).

Models	F_1	SBAR	NP	VP	PP	ADJP	ADVP
Unsupervised parsers							
PRPN†	47.3	50	59	46	57	44	32
ON-LSTM †	48.1	51	64	41	54	38	31
Neural PCFG†	50.8	52	71	33	58	32	45
Compound PCFG †	55.2	**56**	74	41	68	40	52
Neural L-PCFG‡	55.3	53	67	**48**	65	49	58
CPE-PLM (Ours)							
XLM-R + Greedy	48.5	46	69	29	62	48	73
All PLMs + Greedy	55.3	54	**75**	36	**76**	**50**	**76**
All PLMs + Beam	**55.7**	53	74	42	75	46	72

Table 2: Comparison of the best CPE-PLM variants with unsupervised parsers. We show that with the aid of `Greedy` and `Beam`, CPE-PLM becomes competitive with unsupervised PCFGs. We also report recall scores on six phrasal categories in addition to $F1$ scores. The best result for each column is in **bold**. †: From Kim et al. (2019b). ‡: From Zhu et al. (2020).

art $F1$ score (55.7) on PTB in the CPE-PLM literature, improving by up to eight points compared against the previous best (47.7). We also observe that Transformer encoder-based and multilingual PLMs are more attractive options for CPE-PLM.

Second, we take the best instances of CPE-PLM from Table 1 and compare them with a set of unsupervised parsers on PTB. In particular, we consider PRPN (Shen et al., 2018b), ON (Shen et al., 2019), Neural PCFG, Compound PCFG (Kim et al., 2019b), and Neural L-PCFG (Zhu et al., 2020) as baselines. Note that all the models including CPE-PLM are evaluated on the same condition where we assume we have access to the validation set (for tuning hyperparameters), following the prior work (Kim et al., 2019b). From Table 2, we show that with the introduction of `Greedy` and `Beam`, CPE-PLM becomes comparable to off-the-shelf unsupervised parsers in terms of $F1$. Specifically, our CPE-PLM instance with `Beam` succeeds in achieving the best $F1$ score among all the candidates, and the variant with `Greedy` accomplishes the best recall scores on four phrasal categories (NP, PP, ADJP, and ADVP). Based on these quantitative results, we claim that CPE-PLM is proper to be an alternative for unsupervised parsers in some cases.

Finally, we extend the language domain of our experiments from English to eight other languages. We exploit four multilingual PLMs that are capable of processing all the languages we consider, and each ensemble method is optimized for respective languages. In Table 3, we demonstrate that our ensemble methods are universally more effective than the top-K algorithm across different languages. Furthermore, we confirm that CPE-PLM with `Greedy` attains the best performance in every case when operated on the combination of all the four PLMs and, showing 47.5 $F1$ score on average.

Models / Language	en	eu	fr	de	he	hu	ko	pl	sv	Avg.
Single PLM										
MBERT										
Top-K ensemble	44.6	39.3	35.9	35.9	37.8	33.2	47.5	51.1	32.6	39.8
Greedy ensemble	47.1	40.2	36.9	37.5	38.6	30.2	49.1	52.4	31.9	40.4
Beam ensemble	45.7	41.2	36.1	37.6	38.0	33.8	49.1	51.4	32.6	40.6
XLM										
Top-K ensemble	47.1	34.6	36.4	43.8	41.0	36.3	33.6	58.5	36.0	40.8
Greedy ensemble	47.5	38.4	37.0	45.4	41.5	36.4	35.1	58.0	36.4	41.7
Beam ensemble	47.1	38.7	36.8	43.6	41.9	36.3	35.2	56.8	36.2	41.4
XLM-R										
Top-K ensemble	46.5	39.5	35.8	37.5	40.1	36.6	49.8	52.7	32.8	41.3
Greedy ensemble	48.5	39.4	36.1	39.0	40.3	36.5	50.8	53.5	33.1	41.9
Beam ensemble	47.4	39.0	35.3	37.9	39.7	37.0	50.2	53.9	32.7	41.5
XLM-R-large										
Top-K ensemble	44.3	37.2	29.7	36.3	35.8	31.0	45.5	44.7	27.6	36.9
Greedy ensemble	46.8	39.5	32.9	40.1	37.0	34.0	46.4	47.1	31.0	39.4
Beam ensemble	46.9	39.2	33.0	39.2	36.1	33.4	45.8	50.7	29.0	39.3
Multiple PLMs										
All models										
Top-K ensemble	49.6	40.9	38.8	44.3	44.5	38.5	51.1	58.7	37.2	44.8
Greedy ensemble	**51.9**	**44.0**	**41.9**	**47.3**	**48.1**	**40.1**	**53.7**	**61.4**	**39.0**	**47.5**
Beam ensemble	49.8	42.7	40.4	47.0	45.9	39.4	53.4	60.8	38.2	46.4

Table 3: CPE-PLM with three ensemble methods for nine languages. We observe that it is optimal for every language to leverage `Greedy` on top of the integration of all the four multilingual PLMs considered. The best result for each column is in **bold**.

5 Experiments on Downstream Tasks

In the previous section, we demonstrated that CPE-PLM's performance can be significantly improved by introducing the techniques proposed in this work. We now turn our attention towards its outputs (i.e., generated parse trees) and investigate the utility of such trees in two application scenarios where tree structures are taken as input.

5.1 Training (U)RNNG with Induced Trees

Recurrent Neural Network Grammar (RNNG) (Dyer et al., 2016) and its unsupervised variant (URNNG, Kim et al. (2019c)) are neural architectures which perform language modeling and parsing together. In Kim et al. (2019b), the authors showed that training (U)RNNG with the trees generated by other unsupervised parsers results in a parsing model that is even better than the parsers which provided the trees used in training. Following the previous work, we here examine whether the output trees from CPE-PLM can also function as meaningful signals for training (U)RNNG. For our experiments, we acquire the best two instances from Table 1 (which accomplished 55.3 and 55.7 parsing $F1$ scores respectively) and use them as our pseudo parsers. We employ Compound PCFG (Kim et al., 2019b) as an unsupervised parser baseline.

From Kim et al. (2019b) (The *best* over trials)	PPL (↓)	F_1 (↑)
LSTM LM	86.2	–
PRPN	87.1	47.9
Induced RNNG	95.3	47.8
Induced URNNG	90.1	51.6
ON	87.2	50.0
Induced RNNG	95.2	50.6
Induced URNNG	89.9	55.1
Neural PCFG	252.6	52.6
Induced RNNG	95.8	51.4
Induced URNNG	86.0	58.7
Compound PCFG (**best**)	196.3	60.1
Induced RNNG	89.8	58.1
Induced URNNG	**83.7**	**66.9**
Our results (*Averaged* over several trials)	PPL (↓)	F_1 (↑)
Compound PCFG (re-experimented, **average**)	–	54.0
Induced RNNG	91.5	54.7
Induced URNNG	85.4	57.8
CPE-PLM (All PLMs + `Greedy`)	–	55.3
Induced RNNG	86.3	55.0
Induced URNNG	**81.3**	57.2
CPE-PLM (All PLMs + `Beam`)	–	55.7
Induced RNNG	87.3	57.0
Induced URNNG	82.0	**60.7**

Table 4: Experiments on training (U)RNNG with the trees induced by unsupervised parsers and CPE-PLM. The upper section presents the results reported by Kim et al. (2019c) while the bottom shows the outcomes from our experiments. The best numbers for each column of the respective sections are in **bold**. We show that the (U)RNNGs trained with the trees induced by CPE-PLM attain better language modeling and parsing abilities compared to the cases of unsupervised parsers.

In Table 4, we present results from Kim et al. (2019b) and our experiments on PTB. Note that our results and ones from the previous study are not directly comparable, because we report the performance of each model *averaged* over 4 different runs while Kim et al. (2019b) utilize the *best* instance. From the experimental results, we confirm that the (U)RNNG models trained with CPE-PLM have better language modeling capability than those trained with other unsupervised parsers. In particular, we attain the perplexity of 81.3 when leveraging our greedy ensemble algorithm for CPE-PLM, outperforming the strong baseline (Compound PCFG: 85.4). Moreover, we succeed in obtaining a more powerful parsing model by training (U)RNNG with the aid of CPE-PLM (All PLMs + `Beam`). Using this, we achieve 60.7 in $F1$, 5 points higher than that of the original (55.7).

5.2 Text Classification using Tree LSTM

Recursive neural network (RvNN; Socher et al. (2013); Tai et al. (2015)) is a type of neural architecture, whose composition order is determined by an input tree structure. In spite of RvNNs' strong performance on several sentence-level tasks and robust linguistic motivation on which they were invented, the usage of RvNNs is generally restricted due to their reliance on gold/silver-standard trees.[6] We here attempt to mitigate this limitation by taking advantage of CPE-PLM. To this end, we conduct experiments on four text classification tasks with Tree LSTMs: the target tasks are SST2 (Socher et al., 2013), MR (Pang and Lee, 2005), SUBJ (Pang and Lee, 2004), and TREC (Li and Roth, 2002). We use a variant of Tree LSTM (Kim et al., 2019a) whose leaf nodes are processed by a separate LSTM in advance. We inject three distinct types of trees—right-branching trees, which are a strong heuristic-based approach in English, the trees induced by CPE-PLM (with `Beam`), and those generated by a supervised parser—into the model and evaluate their impact on the final performance.

Models / Tasks (Metric: Acc.)	SST2	MR	SUBJ	TREC
Tree LSTM				
+ Right-branching trees	85.72	83.37	94.80	94.50
+ **CPE-PLM (All PLMs + Beam ensemble)**	86.10	**83.62**	94.85	94.75
+ Supervised parser (Klein and Manning, 2003)	**86.70**	**83.62**	**95.12**	**95.05**

Table 5: Text classification with Tree LSTMs. We observe that CPE-PLM-oriented parses outperform right-branching trees but are inferior to silver-standard trees. All the results are averaged over four different runs.

In Table 5, we present the accuracy of diverse Tree LSTM instances on four tasks. Although the absolute difference in accuracy between the instances is marginal, we discover a clear pattern that silver-standard trees (ones from supervised parsers) are always the most helpful while the parses induced by CPE-PLM rank second, outperforming right-branching trees. This outcome supports our claim that CPE-PLM can be an attractive option when supervised parsers are not available.

On the other hand, we find that the performance of Tree LSTMs is not that sensitive to their tree inputs, which was similarly observed by Shi et al. (2018). However, we highlight that the trees closer to their gold-standard counterparts are more beneficial across all the tasks considered. We leave as future work the application of CPE-PLM to advanced tree models that are more input structure-sensitive.

6 Discussion

So far, we have focused on verifying the utility of CPE-PLM through the lens of (1) its improved parsing performance and (2) the effectiveness of its

[6] We use the term *silver-standard* trees to indicate parse trees predicted by sophisticated supervised parsers.

CPE-PLM configurations	Used proportion of validation set				
	1%	2%	5%	10%	100%
All PLMs					
+ `Greedy`	49.4	49.9	52.7	54.3	55.3
Relative loss (-)	5.9	5.3	2.5	0.9	-
+ `Beam`	51.3	49.8	51.8	52.9	55.7
Relative loss (-)	4.5	6.0	4.0	2.9	-

Table 6: Relative performance loss of CPE-PLM on PTB with regard to the proportion of the validation set used. We obtain reasonable performance only with 1% (17 examples) of the validation set.

Models	Number of used annotations								
	1	2	5	10	17 (1%)	2%	5%	10%	100%
CPE-PLM (All PLMs)									
+ `Greedy`	**46.2**	**48.4**	**49.9**	49.1	49.4	**49.9**	**52.7**	54.3	55.3
+ `Beam`	45.4	45.9	47.7	**49.6**	**51.3**	49.8	51.8	52.9	55.7
Supervised (Benepar)	-	11.6	12.5	14.0	17.0	31.1	50.2	**71.4**	**92.2**

Table 7: Comparison between CPE-PLM and a supervised parser (Benepar) in few-shot settings.

output trees for downstream tasks. In this section, we conduct in-depth analysis on the limitations of the current form of CPE-PLM and propose countermeasures to alleviate the problems.

Reliance on the validation set. CPE-PLM is training-free, but it exploits gold-standard trees from the validation set to decide the best combination of attention heads ($g_{(m,n)}$). Although we allow this configuration in this work to have a fair comparison with some previous work on unsupervised parsing (Kim et al., 2019b) that also made use of the validation set to optimize hyperparameters, it is always better to reduce such reliance as argued in the few-shot classification literature (Perez et al., 2021). Therefore, we here attempt to examine the robustness of CPE-PLM with respect to the number of data instances from the validation set. Specifically, we conduct a controlled experiment where CPE-PLM is provided with only a limited proportion of the validation set. In Table 6, we confirm that CPE-PLM only loses roughly five points in performance when just 17 (1%) gold standard trees are available, implying that they work quite well even with a limited number of validation trees.

Furthermore, to showcase the data-efficiency of CPE-PLM in few-shot settings, we compare the performance of CPE-PLM and an off-the-shelf supervised parser (Benepar; Kitaev and Klein (2018)) in few-shot settings.[7] From Table 7, we discover that CPE-PLM shows much better performance than the normal parser in extreme cases where few dozen trees are provided. When trees more than 10% of the validation set are available, the supervised parser starts to outperform CPE-PLM.

[7]Specification on training a supervised parser in few-shot settings can be found in Appendix A.

Models	$F1$ w/ ensemble (↑)		Inference time (↓)
	`Greedy`	`Beam`	
Unsupervised parsers/CPE-PLM			
Compound PCFG (Kim et al., 2019b)	55.2		31 min.
CPE-PLM (All PLMs)	55.3	55.7	27 min.
Parsers trained with induced trees			
Distance (Shen et al., 2018a)	53.8	55.0	36 sec.
Benepar (Kitaev and Klein, 2018)	**56.6**	**59.3**	**32 sec.**

Table 8: Training normal parsers with supervision from the trees induced by CPE-PLM. We show that it is viable to build a much faster parser while preserving (or even boosting) the performance of CPE-PLM by relying on existing techniques for supervised parsing.

Issues on the execution time. As identified in Table 8, where we estimate the accuracy and execution time of different approaches on PTB, CPE-PLM and Compound PCFG are still too slow to be readily utilized, compared to supervised counterparts which are generally highly optimized. To relieve this inefficiency, we propose to exploit normal parsers (Shen et al., 2018a; Kitaev and Klein, 2018) by training them with the trees generated by CPE-PLM if a suitable amount of gold annotations are not available for supervision. In Table 8, we demonstrate that it is possible to transfer syntactic knowledge from PLMs to supervised parsers without loss of accuracy, while significantly reducing the execution time at the same time. We even obtain performance gain in some cases, achieving nearly 60 in $F1$ score. We expect that this direction can be particularly useful for low-resource languages for which it is hard to collect gold annotations.

7 Conclusion

In this paper, we introduce two ensemble methods and multi-PLM configurations for Constituency Parse Extraction from Pre-trained Language Models (CPE-PLM). We demonstrate that the performance of CPE-PLM can be competitive with that of unsupervised parsers with the aid of the proposed approaches, and that the parses induced by CPE-PLM are practically useful in several applications where parse trees are required as input. We also propose solutions for mitigating some inherent limitations of CPE-PLM. We anticipate that its potential will be further greater in the near future with the introduction of more sophisticated PLMs.

Acknowledgements

We would like to thank anonymous reviewers for their fruitful feedback. This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) (No.2020-0-01373, Artificial Intelligence Graduate School Program (Hanyang University)). This work was supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT) (No.2022R1F1A1074674).

References

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners.

Ethan A. Chi, John Hewitt, and Christopher D. Manning. 2020. Finding universal grammatical relations in multilingual BERT. In *ACL*.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. Electra: Pre-training text encoders as discriminators rather than generators. In *ICLR*.

John Cocke. 1969. *Programming languages and their compilers: Preliminary notes*. New York University.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *ACL*.

Alexis Conneau and Guillaume Lample. 2019. Cross-lingual language model pretraining. In *NeurIPS*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*.

Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A. Smith. 2016. Recurrent neural network grammars. In *NAACL*.

Alex Graves. 2012. Sequence transduction with recurrent neural networks. *arXiv preprint arXiv:1211.3711*.

John Hewitt and Christopher D Manning. 2019. A structural probe for finding syntax in word representations. In *NAACL*.

Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. 2019. What does BERT learn about the structure of language? In *ACL*.

Jae-young Jo and Sung-Hyon Myaeng. 2020. Roles and utilization of attention heads in transformer-based neural language models. In *ACL*.

Tadao Kasami. 1966. An efficient recognition and syntax-analysis algorithm for context-free languages. *Coordinated Science Laboratory Report no. R-257*.

Nitish Shirish Keskar, Bryan McCann, Lav R Varshney, Caiming Xiong, and Richard Socher. 2019. Ctrl: A conditional transformer language model for controllable generation. *arXiv preprint arXiv:1909.05858*.

Taeuk Kim, Jihun Choi, Daniel Edmiston, Sanghwan Bae, and Sang-goo Lee. 2019a. Dynamic compositionality in recursive neural networks with structure-aware tag representations. In *AAAI*.

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 *ICLR*.

Taeuk Kim, Bowen Li, and Sang-goo Lee. 2021. Multilingual chart-based constituency parse extraction from pre-trained language models. In *Findings of the Association for Computational Linguistics: EMNLP 2021*.

Yoon Kim, Chris Dyer, and Alexander Rush. 2019b. Compound probabilistic context-free grammars for grammar induction. In *ACL*.

Yoon Kim, Alexander Rush, Lei Yu, Adhiguna Kuncoro, Chris Dyer, and Gábor Melis. 2019c. Unsupervised recurrent neural network grammars. In *NAACL*.

Nikita Kitaev and Dan Klein. 2018. Constituency parsing with a self-attentive encoder. In *ACL*.

Dan Klein and Christopher D Manning. 2003. Accurate unlexicalized parsing. In *ACL*.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *ACL*.

Bowen Li, Taeuk Kim, Reinald Kim Amplayo, and Frank Keller. 2020. Heads-up! unsupervised constituency parsing via self-attention heads. In *AACL-IJCNLP*.

Xin Li and Dan Roth. 2002. Learning question classifiers. In *COLING*.

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. *arXiv preprint arXiv:1907.11692*.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*.

David Mareček and Rudolf Rosa. 2019. From balustrades to pierre vinken: Looking for syntax in transformer self-attentions. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*.

Bo Pang and Lillian Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *ACL*.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *ACL*.

Ethan Perez, Douwe Kiela, and Kyunghyun Cho. 2021. True few-shot learning with language models.

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

Anna Rogers, Olga Kovaleva, and Anna Rumshisky. 2020. A primer in BERTology: What we know about how BERT works. *TACL*, 8:842–866.

Rudolf Rosa and David Mareček. 2019. Inducing syntactic trees from bert representations. *arXiv preprint arXiv:1906.11511*.

Djamé Seddah, Reut Tsarfaty, Sandra Kübler, Marie Candito, Jinho D. Choi, Richárd Farkas, Jennifer Foster, Iakes Goenaga, Koldo Gojenola Galletebeitia, Yoav Goldberg, Spence Green, Nizar Habash, Marco Kuhlmann, Wolfgang Maier, Joakim Nivre, Adam Przepiórkowski, Ryan Roth, Wolfgang Seeker, Yannick Versley, Veronika Vincze, Marcin Woliński, Alina Wróblewska, and Eric Villemonte de la Clergerie. 2013. Overview of the SPMRL 2013 shared task: A cross-framework evaluation of parsing morphologically rich languages. In *Proceedings of the Fourth Workshop on Statistical Parsing of Morphologically-Rich Languages*.

Yikang Shen, Zhouhan Lin, Athul Paul Jacob, Alessandro Sordoni, Aaron Courville, and Yoshua Bengio. 2018a. Straight to the tree: Constituency parsing with neural syntactic distance. In *ACL*.

Yikang Shen, Zhouhan Lin, Chin wei Huang, and Aaron Courville. 2018b. Neural language modeling by jointly learning syntax and lexicon. In *ICLR*.

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

Haoyue Shi, Karen Livescu, and Kevin Gimpel. 2020. On the role of supervision in unsupervised constituency parsing. In *EMNLP*.

Haoyue Shi, Hao Zhou, Jiaze Chen, and Lei Li. 2018. On tree-based neural sentence modeling. In *EMNLP*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *EMNLP*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *NeurIPS*.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *ACL-IJCNLP*.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019. BERT rediscovers the classical NLP pipeline. In *ACL*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NeurIPS*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.

Zhiyong Wu, Yun Chen, Ben Kao, and Qun Liu. 2020. Perturbed masking: Parameter-free probing for analyzing and interpreting BERT. In *ACL*.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *NeurIPS*.

Daniel H Younger. 1967. Recognition and parsing of context-free languages in time n3. *Information and control*.

Hao Zhu, Yonatan Bisk, and Graham Neubig. 2020. The return of lexical dependencies: Neural lexicalized pcfgs. In *TACL*.

A Appendix: Details on Training Few-shot Parsers

Following Shi et al. (2020), we train a supervised parser (**Benepar**; Kitaev and Klein (2018)) in few-show learning settings to provide a robust baseline for our experiments. To be specific, we leverage the official code and hyperparameters of the parser obtained from `https://github.com/nikitakit/self-attentive-parser`. Given a designated number of parses from the PTB validation set, we utilize 90% of them as the training set while the remaining 10% are used as the real validation set. We train the parser for 100 epochs, similar to Shi et al. (2020). Compared against the experimental results reported from Shi et al. (2020), our few-shot parsers show relatively weaker performance. We conjecture this gap comes from the methods Shi et al. (2020) exploited to boost their performance. For instance, (1) they further pre-trained their word embeddings on sentences from PTB and (2) utilized data augmentation and self-training techniques, all of which are not applied in our case.