Heads-up! Unsupervised Constituency Parsing via Self-Attention Heads

Bowen Li[†] **Taeuk Kim**[‡] **Reinald Kim Amplayo**[†] **Frank Keller**[†]
[†]ILCC, School of Informatics, University of Edinburgh, UK
[‡]Dept. of Computer Science and Engineering, Seoul National University, Korea
{bowen.li,reinald.kim}@ed.ac.uk
taeuk@europa.snu.ac.kr keller@inf.ed.ac.uk

Abstract

Transformer-based pre-trained language models (PLMs) have dramatically improved the state of the art in NLP across many tasks. This has led to substantial interest in analyzing the syntactic knowledge PLMs learn. Previous approaches to this question have been limited, mostly using test suites or probes. Here, we propose a novel fully unsupervised parsing approach that extracts constituency trees from PLM attention heads. We rank transformer attention heads based on their inherent properties, and create an ensemble of high-ranking heads to produce the final tree. Our method is adaptable to low-resource languages, as it does not rely on development sets, which can be expensive to annotate. Our experiments show that the proposed method often outperform existing approaches if there is no development set present. Our unsupervised parser can also be used as a tool to analyze the grammars PLMs learn implicitly. For this, we use the parse trees induced by our method to train a neural PCFG and compare it to a grammar derived from a human-annotated treebank.

1 Introduction

Pre-trained language models (PLMs), particularly BERT (Devlin et al., 2019) and others (Yang et al., 2019; Liu et al., 2019b; Radford et al., 2019) based on the transformer architecture (Vaswani et al., 2017), have dramatically improved the state of the art in NLP. Such models make it possible to train a large, generic language model on vast unannotated datasets, and then fine-tune it for a specific task using a small amount of annotated data. The success of PLMs has led to a large literature investigating the linguistic knowledge that PLMs learn implicitly during pre-training (Liu et al., 2019a; Clark et al., 2019; Kovaleva et al., 2019; Pimentel et al., 2020), sometimes referred to as BERTology (Rogers et al., 2020).

BERTology has been particularly concerned with the question whether BERT-type models learn syntactic structure. Typical approaches include test suites of sentences that instantiate specific syntactic structures (Goldberg, 2019), general probes (also known as diagnostic classifiers, Belinkov and Glass 2019) or structural probes (Hewitt and Manning, 2019). All of these approaches are limited: the first one requires the laborious compilation of language- and construction-specific suites of sentences; the second one sometimes fails to adequately reflect differences in representations (Zhang and Bowman, 2018; Hewitt and Liang, 2019; Voita and Titov, 2020); the third one involves designing a novel extraction model that is not applicable to tasks other than probing (Maudslay et al., 2020).

It is therefore natural to use a parsing task to test whether the representations learned by PLMs contain usable syntactic information. This enables us to test syntactic structure in general, rather than specific constructions, and doesn't require a specialized probe. In this paper, we will therefore use PLM attention heads to construct an *unsupervised constituency parser*. Previously, related approaches have been proposed under the heading of *zero-shot* constituency parsing (Kim et al., 2020a,b).[1] However, this prior work crucially relies on an annotated development set in order to identify transformer heads that are sensitive to syntactic structure. Existing approaches therefore are not truly unsupervised. For most low resource languages, no such annotated data is available, and often not even an annotation scheme exists. Thus, assuming a development set is not a realistic experimental setup (Kann et al., 2019). If a suitable development set is available, Shi et al. (2020) shows that an existing supervised parser trained on a few-shot setting can outperform all the unsupervised parsing

[1]Like Kim et al. (2020b), we use *zero-shot* to refer to the transfer from language modeling to constituency parsing.

Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing, pages 409–424
December 4 - 7, 2020. ©2020 Association for Computational Linguistics

methods by a significant margin. It strongly challenges tuning on an annotated development set for unsupervised parsing.

In this paper, we propose a novel approach to build a PLM-based unsupervised parser that does not require a development set: we rank transformer heads based on their inherent properties, such as how likely tokens are to be grouped in a hierarchical structure. We then ensemble the top-K heads to produce constituency trees.

We evaluate our approach and previous zero-shot approaches on the English Penn Treebank (PTB) and eight other languages on the SPMRL dataset. On the one hand, if the development set is absent, our approach largely outperforms previous zero-shot approaches on the English PTB. On the other hand, if previous zero-shot approaches are equipped with the development set, our approach can still match the parsing performance of these approaches using the single best head or layer-wise ensembling. For the multilingual experiment, we take advantage of the top-K heads selected in English and directly parse other languages using our approach. Surprisingly, on five out of nine languages, this *crosslingual* unsupervised parser matches previous approaches that rely on a development set in each target language with the single best head or layer-wise ensembling. However, our fully unsupervised method lags behind the previous state-of-the-art zero-shot parser if a top-K ensemble is used.

Furthermore, our approach can be use as a tool to analyze the capability of PLMs in learning syntactic knowledge. As no human annotation is required, our approach has the potential to reveal the grammar PLMs learn implicitly. Here, we use the tree structures generated by our parser to train a neural PCFG. We evaluate the learned grammar against the English PTB on internal tags and production rules both qualitatively and quantitatively.

2 Related Work

Recently, neural models have renewed interest in grammar induction. Earlier work (Choi et al., 2018; Williams et al., 2018) attempted to induce grammar by optimizing a sentence classification objective, while follow-up work (Htut et al., 2018; Shen et al., 2018a, 2019) showed that a language modeling objective performs better. Latest work employed autoencoders or probabilistic grammars (Drozdov et al., 2019; Kim et al., 2019a,b; Zhu et al., 2020).

A new line of work is zero-shot constituency parsing, whose goal is to automatically extract trees from PLMs in a parameter-free fashion. The top-down zero-shot parser (Kim et al., 2020a) utilizes the concept of *syntactic distance* (Shen et al., 2018b), where trees are induced by an algorithm that recursively splits a sequence of words in a top-down manner. However, this approach suffers from its greedy search mode, failing to take into account all possible subtrees. The chart-based zero-shot parser (Kim et al., 2020b) applies chart parsing to address this problem. Wu et al. (2020) introduced a parameter-free probing technique to analyze PLMs via perturbed masking.

There is also prior work on extracting constituency trees from self-attention mechanisms of transformers. Mareček and Rosa (2018) proposed heuristic approaches to convert attention weights to trees. Mareček and Rosa (2019) introduced a chart-based tree extraction method in transformer-based neural machine translation encoders and provide a quantitative study.

3 Zero-shot Constituency Parsing via PLMs

In this section, we briefly review the chart-based zero-shot parser and then introduce our ranking-based zero-shot parser.

3.1 Chart-based Zero-shot Parsing

In chart-based zero-shot parsing, a real-valued score $s_{tree}(\boldsymbol{t})$ is assigned for each tree candidate $\boldsymbol{t}$, which decomposes as:

$$s_{tree}(\boldsymbol{t}) = \sum_{(i,j)\in \boldsymbol{t}} s_{span}(i,j),$$

where $s_{span}(i,j)$ is the score (or cost) for a constituent that is located between positions i and j ($1 \leq i \leq j \leq n$, where n is the length of the sentence). Specifically, for a span of length 1, $s_{span}(i,j)$ is defined as 0 when $i = j$. For a span longer than 1, the following recursion applies:

$$s_{span}(i,j) = s_{comp}(i,j) + \min_{i\leq k<j} s_{split}(i,k,j) \quad (1)$$

$$s_{split}(i,k,j) = s_{span}(i,k) + s_{span}(k+1,j), \quad (2)$$

where $s_{comp}(\cdot,\cdot)$ measures the validity or compositionality of the span (i,j) itself, while $s_{split}(i,k,j)$ indicates how plausible it is to split the span (i,j) at position k. Two alternatives have been developed in Kim et al. (2020b) for $s_{comp}(\cdot,\cdot)$: the pair score function $s_p(\cdot,\cdot)$ and the characteristic score function $s_c(\cdot,\cdot)$.

The pair score function $s_p(\cdot,\cdot)$ computes the average pair-wise distance in a given span:

$$s_p(i,j) = \frac{1}{\binom{j-i+1}{2}} \sum_{(w_x,w_y)\in \mathtt{pair}(i,j)} f(g(w_x), g(w_y)), \quad (3)$$

where $\mathtt{pair}(i,j)$ returns a set consisting of all combinations of two words (e.g., w_x, w_y) inside the span (i,j).

Functions $f(\cdot,\cdot)$ and $g(\cdot)$ are the distance measure function and the representation extractor function, respectively. For g, given l as the number of layers in a PLM, g is actually a set of functions $g = \{g^d_{(u,v)} | u = 1,\ldots,l, v = 1,\ldots,a\}$, each of which outputs the attention distribution of the v^{th} attention head on the u^{th} layer of the PLM.[2] In case of the function f, there are also two options, Jensen-Shannon (JSD) and Hellinger (HEL) distance. Thus, $f = \{\text{JSD}, \text{HEL}\}$.

The characteristic score function $s_c(\cdot,\cdot)$ measures the distance between each word in the constituent and a predefined characteristic value $\boldsymbol{c}$ (e.g., the center of the constituent):

$$s_c(i,j) = \frac{1}{j-i+1} \sum_{i\le x\le j} f(g(w_x), \boldsymbol{c}), \quad (4)$$

where $\boldsymbol{c} = \frac{1}{j-i+1}\sum_{i\le y\le j} g(w_y)$.

Since $s_{comp}(\cdot,\cdot)$ is well defined, it is straightforward to compute every possible case of $s_{span}(i,j)$ using the CKY algorithm (Cocke, 1969; Kasami, 1966; Younger, 1967). Finally, the parser outputs $\hat{\boldsymbol{t}}$, the tree that requires the lowest score (cost) to build, as a prediction for the parse tree of the input sentence: $\hat{\boldsymbol{t}} = \arg\min_{\boldsymbol{t}} s_{tree}(\boldsymbol{t})$.

For attention heads ensembling, both a layer-wise ensemble and a top-K ensemble are considered. The first one averages all attention heads from a specific layer, while the second one averages the top-K heads from across different layers. At test time, separate trees produced by different heads are merged to one final tree via *syntactic distance*.[3] The chart-based zero-shot parser achieves the state of the art in zero-shot constituency parsing.

[2]The hidden representations of the given words can also serve as an alternative for g. But Kim et al. (2020a) show that the attention distributions provide more syntactic clues under the zero-shot setting.

[3]Details can be found in Kim et al. (2020b). For the ensemble parsing, marrying chart-based parser and top-down parser yields better results than averaging the attention distributions.

3.2 Ranking-based Zero-shot Parsing

The chart-based zero-shot parser relies on the existing development set of a treebank (e.g., the English PTB) to select the best configuration, i.e., the combination of $\{g \mid g^d_{(u,v)}, u = 1,\ldots,l, v = 1,\ldots,a\}$, $\{f \mid \text{JSD}, \text{HEL}\}$, $\{s_{comp} \mid s_p, s_c\}$, and heads ensemble that achieves the best parsing accuracy. Such a development set always contains hundreds of sentences, hence considerable annotation effort is still required. From the perspective of unsupervised parsing, such results arguably are not fully unsupervised.[4] Another argument against using a development set is that the linguistic assumptions inherent in the expert annotation required to create the development set potentially restrict our exploration of how PLMs model the constituency structures. It could be that the PLM learns valid constituency structures, which however do not match the annotation guidelines that were used to create the development set.

Here, we take a radical departure from the previous work in order to extract constituency trees from PLMs in a fully unsupervised manner. We propose a two-step procedure for unsupervised parsing: (1) identify syntax-related attention heads directly from PLMs without relying on a development set of a treebank; (2) ensemble the selected top-K heads to produce the constituency trees.

For identification of the syntax-related attention heads, we rank all heads by scoring them with a chart-based ranker. We borrow the idea of the chart-based zero-shot parser to build our ranker. Given an input sentence and a specific choice of f and s_{comp}, each attention head $g^d_{(u,v)}$ in the PLM yields one unique attention distribution. Using the chart-based zero-shot parser in Section 3.1, we can obtain the score of the best constituency tree as:[5]

$$s_{parsing}(u,v) = s_{tree}(\hat{\boldsymbol{t}}) = \textstyle\sum_{(i,j)\in\hat{\boldsymbol{t}}} s_{span}(i,j), \quad (5)$$

where $\hat{\boldsymbol{t}} = \arg\min_{\boldsymbol{t}} s_{tree}(\boldsymbol{t})$. It is obvious

[4]Some previous work (Shen et al., 2018a, 2019; Drozdov et al., 2019; Kim et al., 2019a) also use a development set to tune hyperparameters or early-stop training.

[5]Our ranking method works approximately as a maximum a posteriori probability (MAP) estimate, since we only consider the best tree the attention head generates. In unsupervised parsing, marginalization is a standard method for model development. We have tried to apply marginalization to our ranking algorithm where all possible trees are considered and the sum score is calculated (using the `logsumexp` trick) for ranking. But marginalization does not work well for attention distributions, where an "attending broadly" head with higher entropy is more favorable than a syntax-related head with lower entropy. So we only consider the score of the best tree.

that all combinations of $\{f \mid \text{JSD}, \text{HEL}\}$ and $\{s_{comp} \mid s_p, s_c\}$ will produce multiple scores for a given head. Here we average the scores of all such combinations to get one single score. Then we rank all attention heads and select the syntax-related heads for parsing. However, directly applying the chart-based zero-shot parser in Section 3.1 for ranking delivers a trivial, ill-posed solution. The recursion in Eq. (2) only encourages the intra-similarity inside the span. Intuitively, one attention head that produces the *same* attention distribution for each token (e.g., a uniform attention distribution or one that forces every token to attend to one specific token) will get the lowest score (cost) and the highest ranking.[6]

To address this issue, we first introduce inter-similarity into the recursion in Eq. (2) and get the following:

$$s_{split}(i,k,j) = \\ s_{span}(i,k) + s_{span}(k+1,j) - s_{cross}(i,k,j), \tag{6}$$

where the cross score $s_{cross}(i,k,j)$ is the similarity between two subspans (i,k) and $(k+1,j)$. However, this formulation forces the algorithm to go to the other extreme: one attention head that produces a totally *different* distribution for each token (e.g., force each token to attend to itself or the previous/next token) will get the highest ranking. To balance the inter- and intra-similarity and avoid having to introduce a tunable coefficient, we simply add a length-based weighting term to Eq. (1) and get:

$$s_{span}(i,j) = \\ \frac{j-i+1}{n}(s_{comp}(i,j) + \min_{i \le k < j} s_{split}(i,k,j)), \tag{7}$$

where $j-i+1$ is the length of the span (i,j). The length ratio functions as a regulator to assign larger weights to longer spans. This is motivated by the fact that longer constituents should contribute more to the scoring of the parse tree, since the inter-similarity always has strong effects on shorter spans. In this way, the inter- and intra-similarity can be balanced.

With respect to the choice for $s_{cross}(i,k,j)$, we follow the idea of s_p and s_c in Eq. (3) and (4) and propose the pair score function s_{px} and the characteristic score function s_{cx}[7] for cross score computation. s_{px} is defined as:

$$s_{\text{px}}(i,j) = \frac{1}{(k-i+1)(j-k)} \\ \sum_{(w_x,w_y)\in \texttt{prod}(i,k,j)} f(g(w_x), g(w_y)),$$

where $\texttt{prod}(i,k,j)$ returns a set of the product of words from the two subspans (i,k) and $(k+1,j)$. And s_{cx} is defined as:

$$s_{\text{cx}}(i,j) = f(\boldsymbol{c}_{i,k}, \boldsymbol{c}_{k+1,j}),$$

where $\boldsymbol{c}_{i,k} = \frac{1}{k-i+1}\sum_{i \le x \le k} g(w_x)$, $\boldsymbol{c}_{k+1,j} = \frac{1}{j-k}\sum_{k+1 \le y \le j} g(w_y)$.

We average all the combinations of $\{f \mid \text{JSD}, \text{HEL}\}$, $\{s_{comp} \mid s_p, s_c\}$ and $\{s_{cross} \mid s_{\text{px}}, s_{\text{cx}}\}$ to rank all the attention heads and select the top-K heads. After the ranking step, we perform constituency parsing by ensembling the selected heads. We simply employ the ensemble method in Section 3.1 and average all the combinations of $\{f \mid \text{JSD}, \text{HEL}\}$ and $\{s_{comp} \mid s_p, s_c\}$ to get a single predicted parse tree for a given sentence.

3.3 How to select K

For ensemble parsing, Kim et al. (2020b) proposed three settings: the best head, layerwise ensemble, and top-K ensemble. To prevent introducing a tunable hyperparameter, we propose to select a value for K dynamically based on a property of the ranking score in Eq. (5).

Since we use a similarity-based distance, the lower the ranking score, the higher the ranking. Assuming that scores are computed for all attention heads, we can sort the scores in ascending order. Intuitively, given the order, we would like to choose the k for which ranking score increases the most, which means syntactic relatedness drops the most. Suppose $s_{parsing}(k)$ is the ranking score where k is the head index in the ascending order, then this is equivalent to finding the k with the greatest gradient on the curve of the score. We first estimate the gradient of $s_{parsing}(k)$ and then find the k with the greatest gradient. Finally, K is computed as:

$$K = \arg\max_k \sum_{\substack{k-\delta \le j \le k+\delta \\ j \ne k}} \frac{s_{parsing}(k+j) - s_{parsing}(k)}{j},$$

[6]Such cases do exist in PLMs. Clark et al. (2019) shows that BERT exhibits clear surface-level attention patterns. Some of these patterns will deliver ill-posed solutions in ranking: attend broadly, attend to a special tokens (e.g., [SEP]), attend to punctuation (e.g., period). One can also observe these patterns using the visualization tool provided by Vig (2019).

[7]Subscripts in the naming of functions in this paper: p – pair score, c – characteristic score, x – cross score.

where we smooth the gradient by considering δ steps. Here, we set $\delta = 3$.

In practice, we find that the greatest gradient always happens in the head or the tail of the curve. For the robustness, we select the K from the middle range of the score function curve, i.e., starting from 30 and ending with 75% of all heads.[8] We also provide a *lazy* option for K selection, which simply assume a fixed value of 30 for the top-K ensemble.

4 Grammar Learning

We are also interested in exploring to what extent the syntactic knowledge acquired by PLMs resembles human-annotated constituency grammars. For this exploration, we infer a constituency grammar, in the form of probabilistic production rules, from the trees induced from PLMs. This grammar can then be analyzed further, and compared to human-derived grammars. Thanks to the recent progress in neural parameterization, neural PCFGs have been successfully applied to unsupervised constituency parsing (Kim et al., 2019a). We harness this model[9] to learn probabilistic constituency grammars from PLMs by maximizing the joint likelihood of sentences and parse trees induced from PLMs. In the following, we first briefly review the neural PCFG and then introduce our training algorithm.

4.1 Neural PCFGs

A probabilistic context-free grammar (PCFG) consists of a 5-tuple grammar $\mathcal{G} = (S, \mathcal{N}, \mathcal{P}, \Sigma, \mathcal{R})$ and rule probabilities $\pi = \{\pi_r\}_{r\in\mathcal{R}}$, where S is the start symbol, $\mathcal{N}$ is a finite set of nonterminals, $\mathcal{P}$ is a finite set of preterminals, Σ is a finite set of terminal symbols, and $\mathcal{R}$ is a finite set of rules associated with probabilities π. The rules are of the form:

$$\begin{aligned}
&S \to A, && A \in \mathcal{N} \\
&A \to BC, && A \in \mathcal{N}, \quad B, C \in \mathcal{N} \cup \mathcal{P} \\
&T \to w, && T \in \mathcal{P}, w \in \Sigma.
\end{aligned}$$

[8]Although our ranking algorithm can filter out *noisy* heads, by observing the attention heatmaps, we find that noisy heads sometimes still rank high. We do not do any post-processing to further filter out the noisy heads, so we empirically search k starting at 30.

[9]A more advanced version of the neural PCFG, the compound PCFG, has also been developed in Kim et al. (2019a). In this model variant, a compound probability distribution is built upon the parameters of a neural PCFG. In preliminary experiments, we found the compound PCFG learns similar grammars as the neural PCFG. So we only use the more lightweight neural PCFG in this work.

Assuming $\mathcal{T}_\mathcal{G}$ is the set of all possible parse trees of $\mathcal{G}$, the probability of a parse tree $\boldsymbol{t} \in \mathcal{T}_\mathcal{G}$ is defined as $p(\boldsymbol{t}) = \prod_{r\in t_\mathcal{R}} \pi_r$, where $t_\mathcal{R}$ is the set of rules used in the derivation of $\boldsymbol{t}$. A PCFG also defines the probability of a given sentence $\boldsymbol{x}$ (string of terminals $\boldsymbol{x} \in \Sigma^*$) via $p(\boldsymbol{x}) = \sum_{\boldsymbol{t}\in\mathcal{T}_\mathcal{G}(\boldsymbol{x})} p(\boldsymbol{t})$, where $\mathcal{T}_\mathcal{G}(\boldsymbol{x}) = \{\boldsymbol{t} | \texttt{yield}(\boldsymbol{t}) = \boldsymbol{x}\}$, i.e., the set of trees $\boldsymbol{t}$ such that $\boldsymbol{t}$'s leaves are $\boldsymbol{x}$.

The traditional way to parameterize a PCFG is to assign a scalar to each rule π_r under the constraint that valid probability distributions must be formed. For unsupervised parsing, however, this parameterization has been shown to be unable to learn meaningful grammars from natural language data (Carroll and Charniak, 1992). Distributed representations, the core concept of the modern deep learning, have been introduced to address this issue (Kim et al., 2019a). Specifically, embeddings are associated with symbols and rules are modeled based on such distributed and shared representations.

In the neural PCFG, the log marginal likelihood:

$$\log p_\theta(\boldsymbol{x}) = \log \sum_{\boldsymbol{t}\in\mathcal{T}_\mathcal{G}(\boldsymbol{x})} p_\theta(\boldsymbol{t})$$

can be computed by summing out the latent parse trees using the inside algorithm (Baker, 1979), which is differentiable and amenable to gradient based optimization. We refer readers to the original paper of Kim et al. (2019a) for details on the model architecture and training scheme.

4.2 Learning Grammars from Induced Trees

Given the trees induced from PLMs (described in Section 3.2), we use neural PCFGs to learn constituency grammars. In contrast to unsupervised parsing, where neural PCFGs are trained solely on raw natural language data, we train them on the sentences and the corresponding tree structures induced from PLMs. Note that this differs from a fully supervised parsing setting, where both tree structures and internal constituency tags (nonterminals and preterminals) are provided in the treebank. In our case, the trees induced from PLMs have no internal annotations.

For the neural PCFG training, the joint likelihood is given by:

$$\log p(\boldsymbol{x}, \hat{\boldsymbol{t}}) = \sum_{r\in\hat{t}_\mathcal{R}} \log \pi_r,$$

where $\hat{\boldsymbol{t}}$ is the induced tree and $\hat{t}_\mathcal{R}$ is the set of rules applied in the derivation of $\hat{\boldsymbol{t}}$. Although tree structures are given during training, marginalization is still involved: all internal tags will be marginalized to compute the joint likelihood. Therefore, the grammars learned by our method are anonymized: nonterminals and preterminals will be annotated as NT-id and T-id, respectively, where id is an arbitrary ID number.

5 Experiments

We conduct experiments to evaluate the unsupervised parsing performance of our ranking-based zero-shot parser on English and eight other languages (Basque, French, German, Hebrew, Hungarian, Korean, Polish, Swedish). For the grammars learned from the induced parse trees, we perform qualitative and quantitative analysis on how the learned grammars resemble the human-crafted grammar of the English PTB.

5.1 General Setup

We prepare the PTB (Marcus et al., 1993) for English and the SPMRL dataset (Seddah et al., 2013) for eight other languages. We adopt the standard split of each dataset to divide it into development and test sets. For preprocessing, we follow the setting in Kim et al. (2019a,b).

We run our ranking algorithm on the development set to select the syntax-related heads and the ensemble parsing algorithm on the test set. We only use the raw sentences in the development set, without any syntactic annotations. We average all configurations both for ranking (f, s_{comp} and s_{cross}) and parsing (f and s_{comp}); hence we do not tune any hyperparameters for our algorithm. For K selection, we experiment with fixed top-K (i.e., top-30) and dynamically searching the best K described in Section 3.3, dubbed dynamic K. We report the unlabeled sentence-level F_1 score to evaluate the extent to which the induced trees resemble the corresponding gold standard trees.

For neural PCFG training, we modify some details but keep most of the model configurations of Kim et al. (2019a); we refer readers to the original paper for more information. We train the models on longer sentences for more epochs. Specifically, we train on sentences of length up to 30 in the first epoch, and increase this length limit by five until the length reaches 80. We train for 30 epochs and use a learning rate scheduler.

Model	Top-down	Chart-based			Our ranking-based		
Configuration	Single /Layer†	Single /Layer†	Top -K	Top -K‡	Top -K	Dynamic K	Full heads
	w/ dev trees			w/o dev trees			
BERT-base-cased	32.6	37.5	**42.7**	29.3	34.8	**37.1**	35.8
BERT-large-cased	36.7	41.5	**44.6**	21.5	36.1	**38.7**	33.2
XLNet-base-cased	39.0	40.5	**46.4**	38.4	41.2	**42.7**	42.4
XLNet-large-cased	37.3	39.7	**46.4**	34.1	40.6	41.1	**41.2**
RoBERTa-base	38.0	41.0	**45.0**	35.9	41.7	**42.1**	39.6
RoBERTa-large	33.8	38.6	**42.8**	30.2	33.1	**37.5**	35.7
GPT2	35.4	34.5	**38.5**	21.9	26.1	**27.2**	26.1
GPT2-medium	37.8	38.5	**39.8**	19.4	**29.1**	**29.1**	27.2
AVG	36.3	39.0	**43.3**	28.8	35.3	**36.9**	35.1
AVG w/o GPT2 *	36.2	39.8	**44.7**	31.6	37.9	**39.8**	38.0

Table 1: Unlabeled sentence-level parsing F_1 scores on the English PTB test set. †: the best results of the top single head and layer-wise ensemble. ‡: directly applying the chart-based parser for ranking (no development set trees) and ensembling the top-K heads for parsing. *: average F_1 scores without GPT2 and GPT2-medium. Bold figures highlight the best scores for the two different groups: with and without development trees.

Model	F_1	SBAR	NP	VP	PP	ADJP	ADVP
Balanced	18.5	7	27	8	18	27	25
Left branching	8.7	5	11	0	5	2	8
Right branching	39.4	**68**	24	**71**	42	27	38
BERT-base-cased	37.1	36	49	30	42	40	69
BERT-large-cased	38.7	38	50	30	46	42	**72**
XLNet-base-cased	**42.7**	45	**58**	31	46	46	**72**
XLNet-large-cased	41.1	44	54	30	42	**48**	64
RoBERTa-base	42.1	38	**58**	31	**47**	42	71
RoBERTa-large	37.5	35	53	29	33	36	54

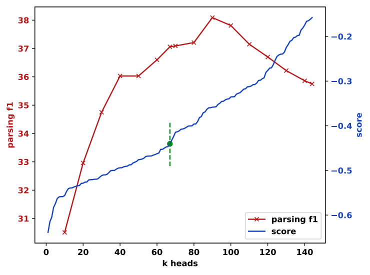
Table 2: Unlabeled parsing scores and recall scores on six constituency tags of trivial baseline parse trees as well as ones achieved by our parser using dynamic K on different PLMs.

5.2 Results on the English PTB

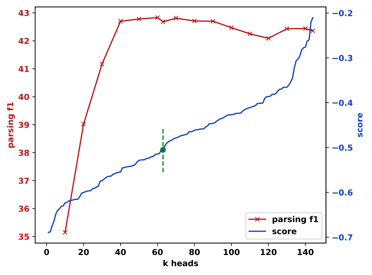
We first evaluate our ranking-based zero-shot parser on the English PTB dataset. We apply our methods to four different PLMs for English: BERT (Devlin et al., 2019), XLNet (Yang et al., 2019), RoBERTa (Liu et al., 2019b), and GPT2 (Radford et al., 2019).[10]

Table 1 shows the unlabeled F_1 scores for our ranking-based zero-shot parser as well as for previous zero-shot parsers in two settings, with and without an annotated development set. We employ the chart-based parser in a setting without development trees, where Eqs. (1) and (2) are used for

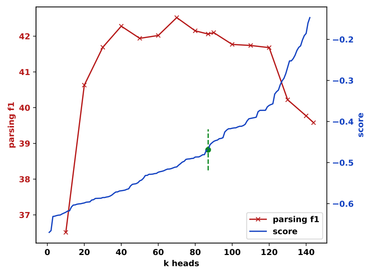
[10]We follow previous work (Kim et al., 2020a,b) in using two variants for each PLM, where the X-base variants consist of 12 layers, 12 attention heads, and 768 hidden dimensions, while the X-large ones have 24 layers, 16 heads, and 1024 dimensions. With regard to GPT2, the GPT2 model corresponds to X-base while GPT2-medium to X-large.



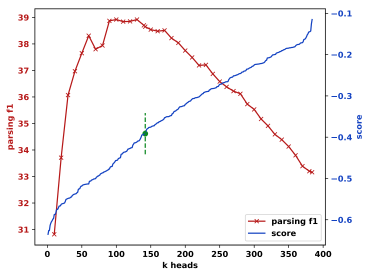
(a) BERT-base-cased



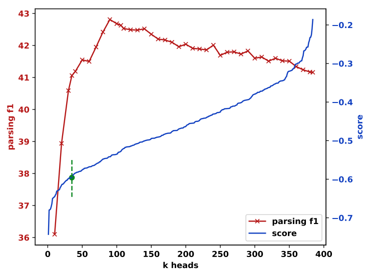
(b) XLNet-base-cased



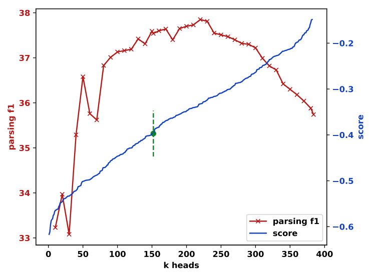
(c) RoBERTa-base



(d) BERT-large-cased



(e) XLNet-large-cased



(f) RoBERTa-large

Figure 1: Relation between K for top-K and parsing performance on different PLMs. The blue curve shows the ranking score of heads where heads are sorted in an ascending order. The red curve shows the parsing performance that is evaluated on the PTB test set given every 10 heads. The green dashed line indicates the dynamic K.

ranking and ensembling the top-K (i.e., top-30) heads. Compared to our method under the same configuration, its poor performance confirms the effectiveness of our ranking algorithm.

With respect to the K selection, our dynamic K method beats both fixed top-30 and full heads. Surprisingly, using all attention heads for ensemble parsing yields nearly the same performance as using top-30 heads. This suggests that although our ranking algorithm filters out some noisy heads, it is still not perfect. On the other hand, the ensemble parsing method is robust to noisy heads when full attention heads are used. Figure 1 shows how the ensemble parsing performance changes given different K selection. We can identify a roughly concave shape of the parsing performance curve, which indicates why our ranking algorithm works. Interestingly, the parsing performance does not drop too much when K reaches the maximum for XLNet. We conjecture that syntactic knowledge is more broadly distributed across heads in XLNet.

Our ranking-based parser performs badly on GPT2 and GPT2-medium, which is not unexpected. Unlike other PLMs, models in the GPT2 category are auto-regressive language models, whose attention matrix is strictly lower triangular. It makes it hard for our ranking algorithm to work properly. But for top-down and chart-based zero-shot parsers, tuning against an annotated development set can alleviate this problem. We focus on BERT, XLNet and RoBERTa and only evaluate these three models in the rest of our experiments. Except for GPT2 variants, our parser with dynamic K outperforms the top-down parser in all cases. On average (without GPT2 variants), even though our parser only requires raw sentence data, it still matches the chart-based parser with the top single head or layer-wise ensemble. To explore the limit of the chart-based parser, we also present the results by selecting the top-K (i.e., top-20) heads using the annotated development set (Kim et al., 2020b). [11] Note that in this setting, the best configuration, i.e., the combination of g, f and s_{comp} as well as K are selected against the development set. This setting serves as an upper bound of the chart-based zero-shot parsing and largely outperforms our ranking-based method.

Table 2 presents the parsing scores as well as recall scores on different constituents of trivial baselines and our parser. It indicates that trees induced from XLNet-base-cased, XLNet-large-cased and RoBERTa-base can outperform the right-branching baseline without resembling it. This confirms that PLMs can produce non-trivial parse trees. Large gains on NP, ADJP and ADVP compared to the

[11]Selecting heads against a development set ensures the quality of high ranking heads; top-20 heads are optimal in this setting (Kim et al., 2020b), unlike top-30 in our setting.

	Model	English	Basque	French	German	Hebrew	Hungarian	Korean	Polish	Swedish	AVG
	Trivial baselines										
	Balanced	18.5	24.4	12.9	15.2	18.1	14.0	20.4	26.1	13.3	18.1
	Left branching	8.7	14.8	5.4	14.1	7.7	10.6	16.5	28.7	7.6	12.7
	Right branching	39.4	22.4	1.3	3.0	0.0	0.0	21.1	0.7	1.7	10.0
w/ dev trees	Chart-based (Single/Layer) †										
	M-BERT	41.2	38.1	30.6	32.1	31.9	30.4	46.4	43.5	27.5	35.7
	XLM	43.0	35.3	35.6	41.6	39.9	34.5	35.7	51.7	33.7	39.0
	XLM-R	44.4	40.4	31.0	32.8	34.1	32.4	47.5	44.7	29.2	37.4
	XLM-R-large	40.8	36.5	26.4	30.2	32.1	26.8	45.6	47.9	25.8	34.7
	AVG	42.4	37.6	30.9	34.2	34.5	31.0	43.8	46.9	29.1	36.7
	Chart-based (top-K) †										
	M-BERT	45.0	41.2	35.9	35.9	37.8	33.2	47.6	51.1	32.6	40.0
	XLM	**47.7**	41.3	**36.7**	**43.8**	**41.0**	36.3	35.7	**58.5**	**36.5**	**41.9**
	XLM-R	47.0	**42.2**	35.8	37.7	40.1	**36.6**	**51.0**	52.7	32.9	41.8
	XLM-R-large	45.1	40.2	29.7	37.1	36.2	31.0	46.9	47.9	27.8	38.0
	AVG	46.2	41.2	34.5	38.6	38.8	34.3	45.3	52.6	32.5	40.4
w/o dev trees	Crosslingual ranking-based (Dynamic K) ‡										
	M-BERT	40.7	**38.2**	31.0	31.0	29.0	27.1	43.3	30.7	25.8	33.0
	XLM	44.9	26.6	**35.8**	**39.7**	**39.6**	32.9	28.0	**50.1**	**34.1**	36.9
	XLM-R	**45.5**	**38.2**	34.0	35.5	36.7	**33.5**	**45.2**	39.4	29.9	**37.6**
	XLM-R-large	41.0	37.9	28.0	28.0	31.3	24.6	44.4	32.2	24.9	32.5
	AVG	43.0	34.7	32.4	33.5	35.0	29.8	40.4	39.2	29.2	35.3

Table 3: Parsing results on nine languages with multilingual PLMs. †: attention heads are selected on the development trees in the target language. ‡: attention heads are selected on raw sentences in English. Bold figures highlight the best scores for the two different groups: with and without development trees.

right branching baseline show that PLMs can better identify such constituents.

5.3 Results for Languages other than English

Low-resource language parsing is one of the main motivations for the development of unsupervised parsing algorithms, which makes a multilingual setting ideal for evaluation. Multilingual PLMs are attractive in this setting because they are trained to process over one hundred languages in a language-agnostic manner. Kim et al. (2020b) has investigated the zero-shot parsing capability of multilingual PLMs assuming that a small annotated development set is available. Here, by taking advantage of our ranking-based parsing algorithm, we use a more radical crosslingual setting. We rank attention heads only on sentences in English and directly apply the parser to eight other languages. We follow Kim et al. (2020b) and use four multilingual PLMs: a multilingual version of the BERT-base model (M-BERT, Devlin et al. 2019), the XLM model (Conneau and Lample, 2019), the XLM-R and XLM-R-large models (Conneau et al., 2020). Each multilingual PLM differs in architecture and pre-training data, and we refer readers to the original papers for more details.

In Table 3, our crosslingual parser outperforms the trivial baselines in all cases by a large margin. Compared with the chart-based parser with the top head or layer-wise ensemble, our crosslingual parser can match the performance on five out of nine languages. Among four model variants, XLM-R and XLM-R-large have identical training settings and pre-training data, and so form a controlled experiment. By directly comparing XLM-R and XLM-R-large, we conjecture that, as the capacity of the PLM scales, the model has more of a chance to learn separate hidden spaces for different languages. This is consistent with a recent study on multilingual BERT (Dufter and Schütze, 2020) showing that underparameterization is one of the main factors that contribute to multilinguality. Again, our method lags behind the chart-base zero-shot parser with a top-K ensemble. More experimental results including using target language for head selection in our method can be found in Appendix A.1.

5.4 Grammar Analysis

By not relying on an annotated development set, we have an unbiased way of investigating the tree structures as well as the grammars that are inher-

Trees	Preterminal Acc†	Rule Acc‡	Parsing F_1
Gold*	66.1	**46.2**	-
BERT-base-cased	64.4	24.8	37.1
BERT-large-cased	64.0	22.3	38.7
XLNet-base-cased	**67.7**	26.1	42.7
XLNet-large-cased	65.8	27.3	41.1
RoBERTa-base	65.7	27.2	42.1
RoBERTa-large	62.4	25.1	37.5

Table 4: Preterminal (PoS tag) and production rule accuracies of PCFG_{PLM} and $\text{PCFG}_{\text{Gold}}$ on the entire PTB. †: PoS tagging accuracy using the many-to-one mapping (Johnson, 2007). ‡: production rule accuracy where anonymized nonterminals and preterminals are mapped to the gold tags using the many-to-one mapping. *: $\text{PCFG}_{\text{Gold}}$.

ent in PLMs. Specifically, we first parse the raw sentences using our ranking-based parser described in Section 3.2 and then train a neural PCFG given the induced trees using the method in Section 4.2. We conduct our experiments on the English PTB and evaluate how the learned grammar resembles PTB syntax in a quantitative way on preterminals (PoS tags) and production rules. We visualize the alignment of preterminals and nonterminals of the learned grammar and the gold labels in Appendix A.2 as a qualitative study. We also showcase parse trees of the learned grammar to get a glimpse of some distinctive characteristics of the learned grammar in Appendix A.3. For brevity, we refer to a neural PCFG learned from trees induced of a PLM as PCFG_{PLM} and to a neural PCFG learned from the gold parse trees as $\text{PCFG}_{\text{Gold}}$.

In Table 4, we report preterminal (unsupervised PoS tagging) accuracies and production rule accuracies of PCFG_{PLM} and $\text{PCFG}_{\text{Gold}}$ on the corpus level. For preterminal evaluation, we map the anonymized preterminals to gold PoS tags using many-to-one (M-1) mapping (Johnson, 2007), where each anonymized preterminal is matched onto the gold PoS tag with which it shares the most tokens. For production rule evaluation, we map both nonterminals and preterminals to gold tags using M-1 mapping to get the binary production rules.[12] We find that all PCFG_{PLM} grammars except for $\text{PCFG}_{\text{RoBERTa-large}}$ outperform a discrete HMM baseline (62.7, He et al. 2018) but are far from the state of the art for neural grammar induction (80.8, He et al. 2018). All PCFG_{PLM} produce similar accuracies on preterminals as $\text{PCFG}_{\text{Gold}}$. However, for the production rules, PCFG_{PLM} lags behind $\text{PCFG}_{\text{Gold}}$ by a large margin. This makes sense as presumably the tree structures heavily affect nonterminal learning. We also present the parsing F_1 scores of corresponding trees against the gold trees in Table 4 for comparison. We observe that for all PCFG_{PLM}, both preterminal accuracies and production rule accuracies correlate well with the parsing F_1 scores of the corresponding trees.

[12]For the gold annotations, we drop all unary rules. For n-ary rules ($n > 2$), we convert them to binary rules by right branching and propagating the parent tag. For example, a n-ary rule $A \rightarrow B\ C\ D$ yields $A \rightarrow B\ A$ and $A \rightarrow C\ D$.

6 Conclusion

In this paper, we set out to analyze the syntactic knowledge learned by transformer-based pretrained language models. In contrast to previous work relying on test suites and probes, we proposed to use a zero-shot unsupervised parsing approach. This approach is able to parse sentences by ranking the attention heads of the PLM and ensembling them. Our approach is able to completely do away with a development set annotated with syntactic structures, which makes it ideal in a strictly unsupervised setting, e.g., for low resource languages. We evaluated our method against previous methods on nine languages. When development sets are available for previous methods, our method can match them or produce competitive results if they use the top single head or layer-wise ensembling of attention heads, but lags behind them if they ensemble the top-K heads. Furthermore, we present an analysis of the grammars learned by our approach: we use the induced trees to train a neural PCFG and evaluate the pre-terminal and non-terminal symbols of that grammar. In future work, we will develop further methods for analyzing the resulting grammar rules. Another avenue for follow-up research is to use our method to determine how the syntactic structures inherent in PLMs change when these models are fine-tuned on a specific task.

Acknowledgments

We thank the reviewers for their valuable suggestions regarding this work.

References

James K Baker. 1979. Trainable grammars for speech recognition. *The Journal of the Acoustical Society of America*.

Yonatan Belinkov and James Glass. 2019. Analysis methods in neural language processing: A survey. *TACL*, 7:49–72.

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

Jihun Choi, Kang Min Yoo, and Sang-goo Lee. 2018. Learning to compose task-specific tree structures. In *AAAI*.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. 2019. What does bert look at? an analysis of bert's attention. In *BlackBoxNLP@ACL*.

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

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*.

Andrew Drozdov, Patrick Verga, Mohit Yadav, Mohit Iyyer, and Andrew McCallum. 2019. Unsupervised latent tree induction with deep inside-outside recursive auto-encoders. In *NAACL*.

Philipp Dufter and Hinrich Schütze. 2020. Identifying necessary elements for bert's multilinguality. *arXiv preprint arXiv:2005.00396*.

Yoav Goldberg. 2019. Assessing BERT's syntactic abilities. *arXiv preprint arXiv:1901.05287*.

Junxian He, Graham Neubig, and Taylor Berg-Kirkpatrick. 2018. Unsupervised learning of syntactic structure with invertible neural projections. In *EMNLP*.

John Hewitt and Percy Liang. 2019. Designing and interpreting probes with control tasks. In *EMNLP-IJCNLP*.

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

Phu Mon Htut, Kyunghyun Cho, and Samuel Bowman. 2018. Grammar induction with neural language models: An unusual replication. In *BlackboxNLP@EMNLP*.

Mark Johnson. 2007. Why doesn't em find good hmm pos-taggers? In *EMNLP-CoNLL*.

Katharina Kann, Kyunghyun Cho, and Samuel R Bowman. 2019. Towards realistic practices in low-resource natural language processing: The development set. In *EMNLP-IJCNLP*.

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

Taeuk Kim, Jihun Choi, Daniel Edmiston, and Sang goo Lee. 2020a. 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. 2020b. Multilingual zero-shot constituency parsing. *arXiv preprint arXiv:2004.13805v2*.

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

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

Olga Kovaleva, Alexey Romanov, Anna Rogers, and Anna Rumshisky. 2019. Revealing the dark secrets of BERT. In *EMNLP-IJCNLP*.

Nelson F. Liu, Matt Gardner, Yonatan Belinkov, Matthew E. Peters, and Noah A. Smith. 2019a. Linguistic knowledge and transferability of contextual representations. In *NAACL*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019b. Roberta: A robustly optimized bert pretraining approach. In *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. 2018. Extracting syntactic trees from transformer encoder self-attentions. In *BlackboxNLP@EMNLP*.

David Mareček and Rudolf Rosa. 2019. From balustrades to pierre vinken: Looking for syntax in transformer self-attentions. In *BlackboxNLP@ACL*.

Rowan Hall Maudslay, Josef Valvoda, Tiago Pimentel, Adina Williams, and Ryan Cotterell. 2020. A tale of a probe and a parser. In *ACL*.

Tiago Pimentel, Josef Valvoda, Rowan Hall Maudslay, Ran Zmigrod, Adina Williams, and Ryan Cotterell. 2020. Information-theoretic probing for linguistic structure. In *ACL*.

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. *arXiv preprint arXiv:2002.12327*.

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 *4th Workshop on Statistical Parsing of Morphologically-Rich Languages*.

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

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

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*.

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*.

Jesse Vig. 2019. A multiscale visualization of attention in the transformer model. In *ACL*.

Elena Voita and Ivan Titov. 2020. Information-theoretic probing with minimum description length. *arXiv preprint arXiv:2003.12298*.

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

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*.

Kelly Zhang and Samuel Bowman. 2018. Language modeling teaches you more than translation does: Lessons learned through auxiliary syntactic task analysis. In *BlackboxNLP@EMNLP*.

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

A Appendix

A.1 More Results on Languages other than English

We present a comprehensive analysis of the chart-based parser and our ranking-based parser on the multilingual setting. In addition to Table 3, for our method, we conduct experiments using target language for head selection with both Top-K (i.e., top-30) ensemble and dynamic K ensemble.

In Table 5, we find that our ranking-based parser with Top-K ensemble performs slightly better than that using dynamic K. In contrast to the superiority of dynamic K on English PLMs in Table 1, multi-lingual PLMs produce similar parsing performance with a *lazy* top-30 ensemble. We conjecture that there could be no clear concave pattern (like Figure 1) in the relation of K and parsing performance in this crosslingual setting.

We also experimented with another setting for our ranking-based parser: selecting attention heads based on the sentences in the target language. Interestingly, we observe a considerable parsing performance drop on both top-K and dynamic K ensemble. We suspect that our chart-based ranking algorithm (e.g., the inherent context free grammar assumption) does not work equally well in all languages, at least for the annotation scheme provided by the SPMRL dataset. In this scenario, using English for head selection has a better chance to capture syntax-related attention heads. Again, as we discussed before, using annotated trees in the target language can always ensure the quality of selected top-K heads.

A.2 Visualization of the Alignment for Internal Tags

Since the recall scores in Table 2 have shown ability of PLMs to identify different nonterminals, here we visualize the alignment between PCFG internal tags and corresponding gold labels in Figures 2 and 3. For the nonterminal alignment, some of the learned nonterminals clearly align to gold standard labels, in particular for frequent ones like NP and VP. Compared to $\text{PCFG}_{\text{Gold}}$, PCFG_{PLM} learns a more uncertain grammar and resulting in overall lower precision.

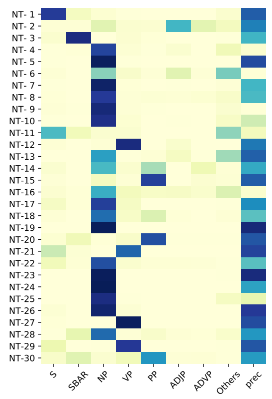
But for the preterminal (PoS tag) alignment, no clear difference can be identified between $\text{PCFG}_{\text{Gold}}$ and PCFG_{PLM}. This is consistent with the finding in Table 4 that all PCFG_{PLM} produce similar accuracies on preterminals as $\text{PCFG}_{\text{Gold}}$.

A.3 Parse tree samples

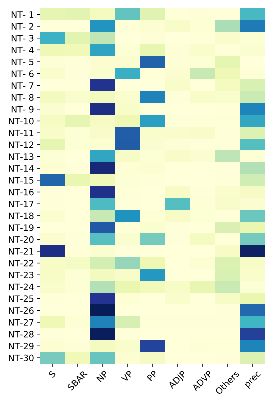
In Figure 4, we show parse trees obtained by $\text{PCFG}_{\text{Gold}}$, PCFG_{PLM} and the gold standard reference on a sample sentence. In this sample, $\text{PCFG}_{\text{Gold}}$ predicts the constituency tree structure accurately. On the development set, $\text{PCFG}_{\text{Gold}}$ reaches around 72 unlabeled F_1 score, as it is supervised by the PTB trees. Although this is a low F_1-score, it is not untypical for PCFG-based models, which are limited by their insufficiently flexible rules and their lack of lexicalization. Also note that the oracle trees only yield 84.3 F_1. PCFG_{PLM} perform worse than $\text{PCFG}_{\text{Gold}}$ when compared against the gold tree. They are able to identify short NPs, but don't work well for larger constituents. We also observe some frequent incorrect patterns which are also present in this example, e.g., grouping VBD with the preceding NP, or IN with the preceding VBD.

	Language	English	Basque	French	German	Hebrew	Hungarian	Korean	Polish	Swedish	AVG
	Trivial baselines										
	Balanced	18.5	24.4	12.9	15.2	18.1	14.0	20.4	26.1	13.3	18.1
	Left branching	8.7	14.8	5.4	14.1	7.7	10.6	16.5	28.7	7.6	12.7
	Right branching	39.4	22.4	1.3	3.0	0.0	0.0	21.1	0.7	1.7	10.0
Target language for head selection	Chart-based (Single/Layer) †										
	M-BERT	41.2	38.1	30.6	32.1	31.9	30.4	46.4	43.5	27.5	35.7
	XLM	43.0	35.3	35.6	41.6	39.9	34.5	35.7	51.7	33.7	39.0
	XLM-R	44.4	40.4	31.0	32.8	34.1	32.4	47.5	44.7	29.2	37.4
	XLM-R-large	40.8	36.5	26.4	30.2	32.1	26.8	45.6	47.9	25.8	34.7
	AVG	42.4	37.6	30.9	34.2	34.5	31.0	43.8	46.9	29.1	36.7
	Chart-based (Top-K) †										
	M-BERT	45.0	41.2	35.9	35.9	37.8	33.2	47.6	51.1	32.6	40.0
	XLM	**47.7**	41.3	**36.7**	**43.8**	**41.0**	36.3	35.7	**58.5**	**36.5**	**41.9**
	XLM-R	47.0	**42.2**	35.8	37.7	40.1	**36.6**	**51.0**	52.7	32.9	41.8
	XLM-R-large	45.1	40.2	29.7	37.1	36.2	31.0	46.9	47.9	27.8	38.0
	AVG	46.2	41.2	34.5	38.6	38.8	34.3	45.3	52.6	32.5	40.4
	Ranking-based (Top-K) ‡										
	M-BERT	41.5	38.9	33.9	30.2	36.3	30.9	39.0	18.4	26.3	31.7
	XLM	44.6	21.0	29.8	39.2	30.5	25.2	23.8	55.2	30.3	31.9
	XLM-R	44.8	36.0	34.1	31.8	36.4	32.5	40.3	29.6	26.7	33.4
	XLM-R-large	41.1	36.8	30.3	26.8	33.4	24.9	37.4	17.5	26.3	29.2
	AVG	43.0	33.2	32.0	32.0	34.2	28.4	35.1	30.2	27.4	31.6
	Ranking-based (Dynamic K) ‡										
	M-BERT	40.7	39.1	28.4	25.5	26.9	31.2	41.3	22.2	21.3	29.5
	XLM	44.9	20.8	29.9	40.3	34.4	27.7	23.6	55.1	31.2	32.9
	XLM-R	45.5	37.3	30.7	31.5	31.8	34.1	40.8	36.0	27.4	33.7
	XLM-R-large	41.0	36.5	29.0	30.1	32.6	25.3	43.9	30.0	25.5	31.6
	AVG	43.0	33.4	29.5	31.9	31.4	29.6	37.4	35.8	26.4	31.9
English for head selection	Crosslingual ranking-based (Top-K) ‡										
	M-BERT	-	37.9	33.4	31.2	31.5	29.4	45.3	33.4	27.2	34.5
	XLM	-	25.9	34.4	39.2	39.5	31.9	27.5	**50.4**	**34.2**	36.4
	XLM-R	-	37.9	33.9	35.1	36.8	33.3	44.7	39.7	30.3	37.4
	XLM-R-large	-	35.7	28.5	28.5	34.7	25.5	44.5	36.9	27.1	33.6
	AVG	-	34.3	32.6	33.5	35.6	30.0	40.5	40.1	29.7	35.5
	Crosslingual ranking-based (Dynamic K) ‡										
	M-BERT	-	**38.2**	31.0	31.0	29.0	27.1	43.3	30.7	25.8	33.0
	XLM	-	26.6	**35.8**	**39.7**	**39.6**	32.9	28.0	50.1	34.1	36.9
	XLM-R	-	**38.2**	34.0	35.5	36.7	**33.5**	**45.2**	39.4	29.9	**37.6**
	XLM-R-large	-	37.9	28.0	28.0	31.3	24.6	44.4	32.2	24.9	32.5
	AVG	-	34.7	32.4	33.5	35.0	29.8	40.4	39.2	29.2	35.3

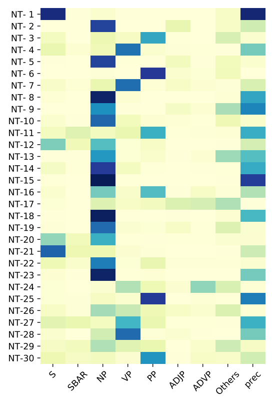
Table 5: Parsing results on nine languages with multilingual PLMs. Except for the trivial baselines, all experimental results are divided into two groups: using target language for head selection and using English for head selection (crosslingual). †: results of the best configurations of f, g, s_{comp} and K are decided on an annotated development set. ‡: results where only raw sentences are required. For top-K, 20 is used for chart-based and 30 is used for our ranking-based. Bold figures highlight the best scores for the two different groups: using target language and English for head selection.



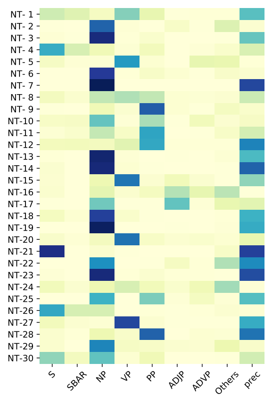
(a) PCFG$_{\text{Gold}}$



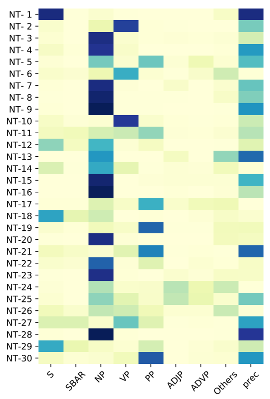
(b) PCFG$_{\text{BERT-base-cased}}$



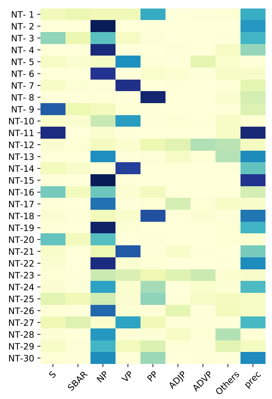
(c) PCFG$_{\text{BERT-large-cased}}$



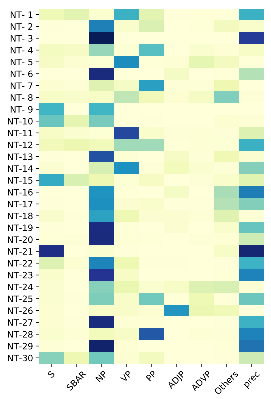
(d) PCFG$_{\text{XLNet-base-cased}}$



(e) PCFG$_{\text{XLNet-large-cased}}$

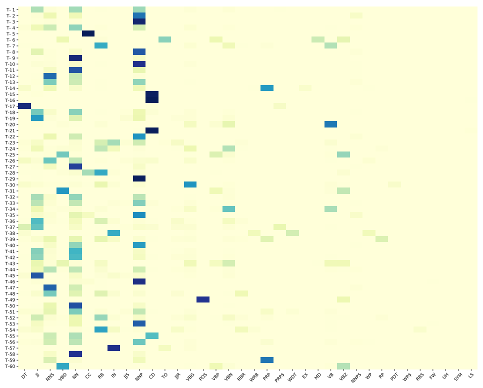


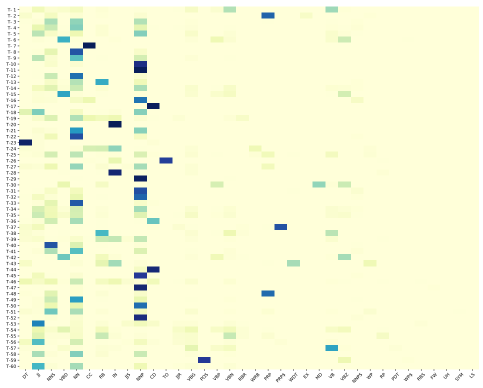
(f) PCFG$_{\text{RoBERTa-base}}$

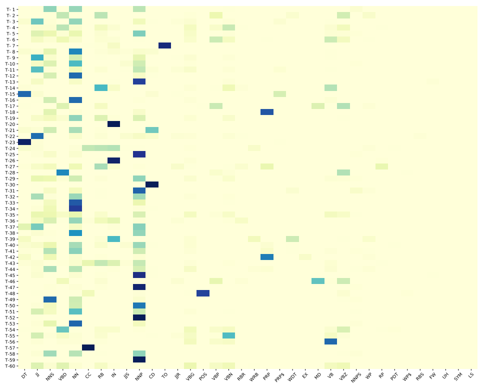


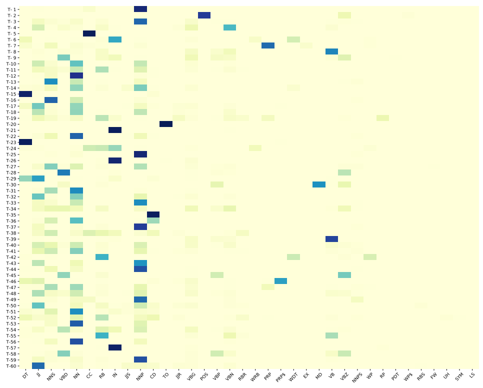
(g) PCFG$_{\text{RoBERTa-large}}$

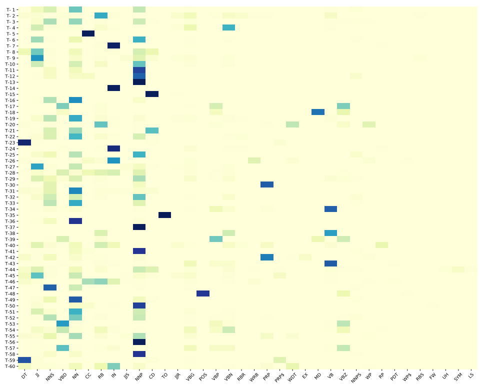
Figure 2: Alignment of induced nonterminals of PCFG$_{\text{PLM}}$ and PCFG$_{\text{Gold}}$ on the entire PTB. The last column `prec` shows the precision that a nonterminal predicts a particular gold constituent.

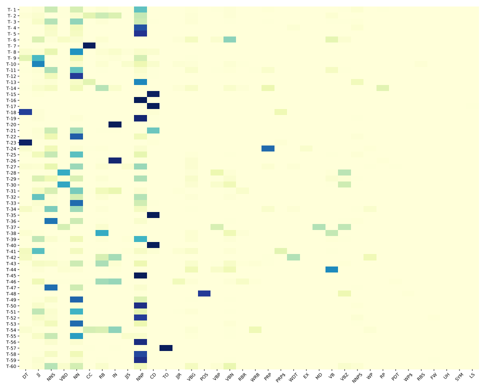

(a) $\text{PCFG}_{\text{Gold}}$


(b) $\text{PCFG}_{\text{BERT-base-cased}}$


(c) $\text{PCFG}_{\text{BERT-large-cased}}$


(d) $\text{PCFG}_{\text{XLNet-base-cased}}$


(e) $\text{PCFG}_{\text{XLNet-large-cased}}$


(f) $\text{PCFG}_{\text{RoBERTa-base}}$

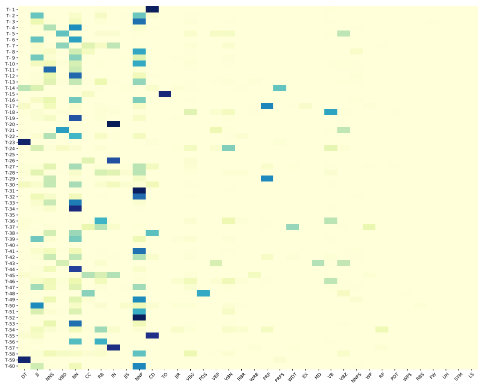

(g) $\text{PCFG}_{\text{RoBERTa-large}}$

Figure 3: Alignment of induced preterminals (PoS tags) of PCFG_{PLM} and $\text{PCFG}_{\text{Gold}}$ on the entire PTB.

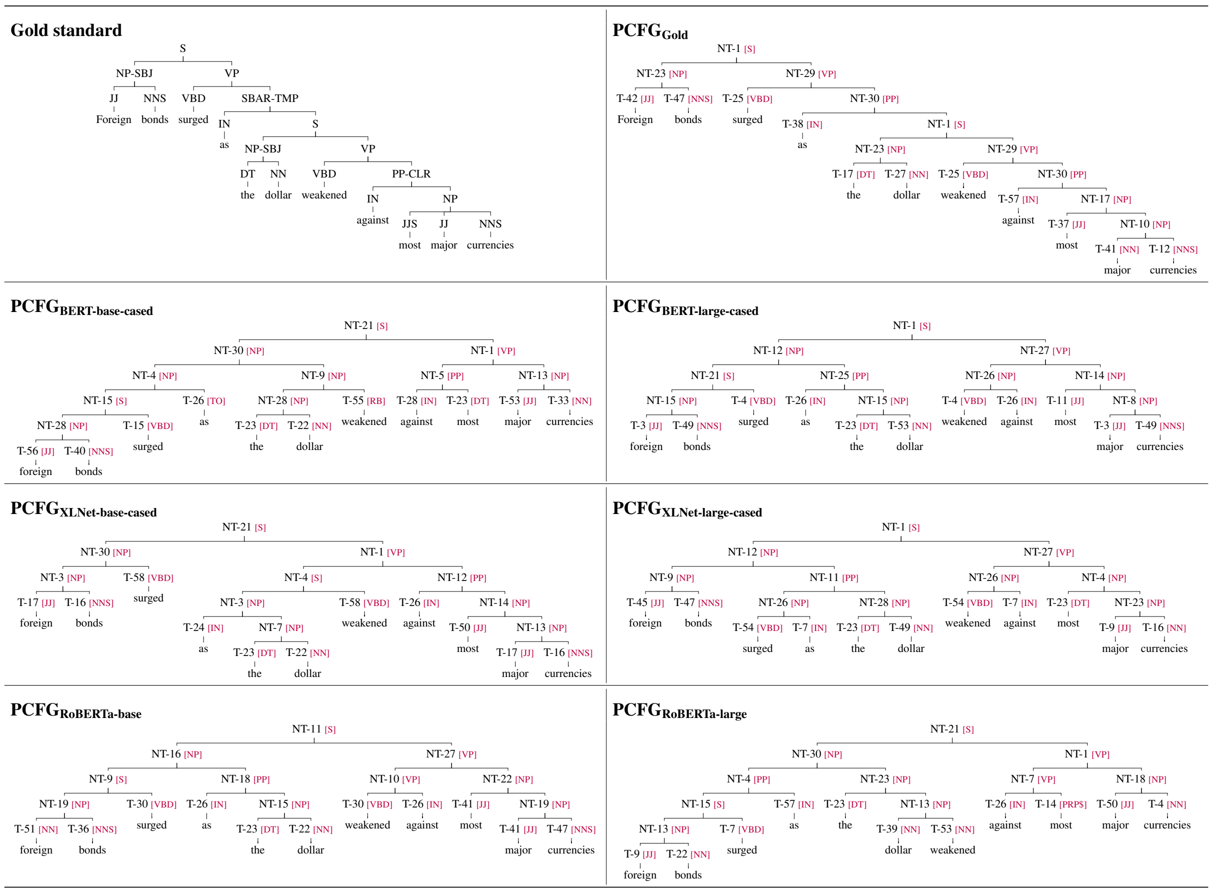


Figure 4: Parse tree samples of gold standard, $\text{PCFG}_{\text{Gold}}$, and PCFG_{PLM}. The mapped tag (marked in red) for each anonymized nonterminal and preterminal is obtained via many-to-one mapping.