Do Transformers Parse while Predicting the Masked Word?

Anonymous ACL submission

Abstract

Pre-trained language models have been shown to encode linguistic structures like parse trees in their embeddings while being trained unsupervised. Some doubts have been raised whether the models are doing parsing or only some computation weakly correlated with it. Concretely: (a) Is it possible to explicitly describe transformers with realistic embedding dimensions, number of heads, etc. that are *capable* of doing parsing —or even approximate parsing? (b) Why do pre-trained models capture parsing structure? This paper takes a step toward answering these questions in the context of generative modeling with PCFGs. We show that masked language models like BERT or RoBERTa of moderate sizes can approximately execute the Inside-Outside algorithm for the English PCFG (Marcus et al., 1993). We also show that the Inside-Outside algorithm is optimal for masked language modeling loss on the PCFG-generated data. We conduct probing experiments on models pre-trained on PCFG-generated data to show that this not only allows recovery of approximate parse tree, but also recovers marginal span probabilities computed by the Inside-Outside algorithm, which suggests an implicit bias of masked language modeling towards this algorithm.

1 Introduction

One of the surprising discoveries about transformer-based language models like BERT (Devlin et al., 2019) and RoBERTa (Liu et al., 2019) was that contextual word embeddings encode information about parsing, which can be extracted using a simple "linear probing" to yield approximately correct dependency parse trees for the text (Hewitt and Manning, 2019; Manning et al., 2020). Subsequently, Vilares et al. (2020); Wu et al. (2020); Arps et al. (2022) employed linear probing also to recover information about constituency parse trees. Investigating the parsing capability of transformers is of significant interest, as incorporating (the awareness of) syntax in large language models has been shown to enhance the final performance on various downstream tasks (Xu et al., 2021; Bai et al., 2021). Additionally, it can contribute to the ongoing exploration of the "mechanistic interpretability" for reverse engineering the inner workings of pre-trained large language models (Elhage et al., 2021; Olsson et al., 2022; Nanda et al., 2023).

The current paper focuses on the ability of BERT-style transformers to do constituency parsing, specifically for PCFGs. Prior studies (Bhattamishra et al., 2020b; Pérez et al., 2021) established that transformers are Turing complete, suggesting their potential for parsing. But do they actually parse while trying to do masked-word prediction? One reason to be cautiously skeptical is that naive translation of constituency parsing algorithms into a transformer results in transformers with number of heads that scales with the size of the grammar (Section 3.1), whereas BERT-like models have around a dozen heads. This leads to the following question.

(Qs 1): Are BERT-like models capable of parsing with realistic number of heads?

This is not an idle question as Maudslay and Cotterell (2021) suggested that linear probing relies on semantic cues for parsing. They created syntactically correct but semantically meaningless sentences and found a significant drop in parsing performance compared to previous studies.

(Qs 2): Do BERT-like models trained for masked language modeling (MLM) encode syntax, and if so, how and why?

1.1 This paper

To address Qs 1, we construct a transformer that executes the Inside-outside algorithm for PCFG (Section 3.1). If the PCFG has N non-terminals and the length of the sentence is L, our constructed transformer has $2L$ layers in total, N attention heads, and $2NL$ embedding dimensions in each layer. However, this is massive compared to BERT. For PCFG learned on Penn Treebank (PTB) (Marcus et al., 1993), $N = 1600$, average $L \approx 25$, which leads to a transformer with 80k embedding dimension, depth 50, and 1.6k attention heads per layer. By contrast, BERT has 768 embedding dimensions, 12 layers, and 12 attention heads per layer!

One potential explanation could be that BERT does not do exact parsing but merely computes *some* information related to parsing. After all, linear probing didn't recover complete parse trees. It recovered trees with modest F1 score, such as 78.2% for BERT (Vilares et al., 2020) and 82.6% for RoBERTa (Arps et al., 2022). To the best of our knowledge, no study has investigated parsing methods that strategically discard information to do more efficient approximate parsing. Toward this goal, we design an approximate version of the Inside-Outside algorithm (Section 3.3), executable by a transformer with $2L$ layers, 15 attention heads, and $40L$ embedding dimensions, while still achieving $> 70\%$ F1 score for constituency parsing on PTB dataset (Marcus et al., 1993).

Although realistic models can capture a fair amount of parsing information, it is unclear whether they need to do so for masked language modeling (MLM). After all, Maudslay and Cotterell (2021) suggested that linear probing picks up on semantic information that happens to correlate with parse trees. To further explore this, we trained a (masked) language model on the synthetic text generated from a PCFG tailored to English text, separating syntax from semantics in a more rigorous manner than Maudslay and Cotterell (2021). Section 3.2 notes that given such synthetic text, the Inside-Outside algorithm will minimize MLM loss. Note that parsing algorithms like CYK (Kasami, 1966) could be used instead of Inside-Outside, but they do not have an explicit connection to MLM (Section 3.2). Experiments with pre-trained models on synthetic PCFG data (Section 4.1) reveal the existence of syntactic information inside the models: simple probing methods recover reasonable parse tree structure (Section 4.2). Additionally, probes of contextualized embeddings reveal correlations with the information computed by the Inside-Outside algorithm (Section 4.3). This suggests transformers implicitly engage in a form of approximate parsing, in particular a process related to the Inside-Outside algorithm, to achieve low MLM loss.

2 Preliminaries

2.1 Attention

We'll focus on encoder-only transformers like BERT and RoBERTa (Devlin et al., 2019; Liu et al., 2019), which stack identical layers with an attention module followed by a feed-forward module. Each attention module has multiple heads, represented by three matrices $\boldsymbol{Q}_h, \boldsymbol{K}_h, \boldsymbol{V}_h \in \mathbb{R}^{d\times d}$. For an input sequence of length L, we use $\boldsymbol{E}^{(\ell)} \in \mathbb{R}^{L\times d}$ to denote contextual embeddings after layer ℓ's computations, where $\boldsymbol{e}_i^{(\ell)}$ is the embedding of the i^{th} token. The output of the attention head h at layer ℓ is $\boldsymbol{v}_{i,h}^{(\ell)} = \sum_{j\in[L]} a_{i,j}^h \boldsymbol{V}_h \boldsymbol{e}^{(\ell)}$, where $a_{i,j}^h$ is the attention score between $\boldsymbol{e}_i$ and $\boldsymbol{e}_j$ for head h:

$$a_{i,j}^h = f_{\text{attn}}(\boldsymbol{E}^{(\ell)}\boldsymbol{K}_h^\top, \boldsymbol{Q}_h \boldsymbol{e}_i^{(\ell)})_j. \quad (1)$$

f_{attn} is a non-linear function and is generally used as softmax on $\boldsymbol{E}^{(\ell)}\boldsymbol{K}_h^\top \boldsymbol{Q}_h \boldsymbol{e}_i^{(\ell)}$. Finally, the output of the attention module is given by $\sum_h \boldsymbol{v}_{i,h}^{(\ell)}$. This is a general definition of the attention module and captures the split and merge of the embeddings across the attention heads used in practice.

2.2 PCFG and parsing

PCFG model A probabilistic context-free grammar (PCFG) is a language generative model. It is defined as a 5-tuple $\mathcal{G} = (\mathcal{N}, \mathcal{I}, \mathcal{P}, n, p)$, where

- $\mathcal{N}$ is the set of non-terminal. $\mathcal{I}, \mathcal{P} \subset \mathcal{N}$ are sets of *in-terminals* and *pre-terminals* respectively. $\mathcal{N} = \mathcal{I} \cup \mathcal{P}$, and $\mathcal{I} \cap \mathcal{P} = \phi$.
- $[n]$ is the set of all possible words.
- $\forall A \in \mathcal{I}, B, C \in \mathcal{N}$, there is a rule $A \to BC$.
- For rule $A \to BC$ where $A \in \mathcal{I}, B, C \in \mathcal{N}$, there is a probability $\Pr[A \to BC]$ satisfying for all A, $\sum_{B,C} \Pr[A \to BC] = 1$.
- For all $A \in \mathcal{P}, w \in [n]$, a rule $A \to w$.
- For each rule $A \to w$ where $A \in \mathcal{P}, w \in [n]$, a probability $\Pr[A \to w]$, which satisfies for all A, $\sum_w \Pr[A \to w] = 1$.
- A non-terminal Root $\in \mathcal{I}$.

Data generation from PCFG Strings are generated from the PCFG $\mathcal{G} = (\mathcal{N}, \mathcal{I}, \mathcal{P}, n, p)$ as follows: we maintain a string $s_t \in ([n] \cup \mathcal{N})^*$ at step t with $s_1 =$ ROOT. At step t, if all characters in s_t belong to $[n]$, the generation process ends, and s_t is the resulting string. Otherwise, we pick a character

$A \in s_t$ such that $A \in \mathcal{N}$. If $A \in \mathcal{P}$, we replace the character A to w with probability $\Pr[A \to w]$. If $A \in \mathcal{I}$, we replace the character A to two characters B, C with probability $\Pr[A \to BC]$.

Parse trees and parsing For a sentence $s = w_1 \ldots w_L$ with length L, a labeled parse tree represents the likely derivations of a sentence under PCFG $\mathcal{G}$. It is defined as a list of spans with non-terminals $\{(A, i, j)\}$ that forms a tree. An unlabelled parse tree is a list of spans that forms a tree.

To find the unlabelled parse tree for a sentence s under the PCFG model, the Labelled-Recall algorithm (Goodman, 1996) is commonly used. This algorithm searches for the tree $T = \{(i, j)\}$ that maximizes $\sum_{(i,j) \in T} \operatorname{score}(i, j)$, where $\operatorname{score}(i, j) = \max_{A \in \mathcal{N}} \Pr[A \Rightarrow w_i w_{i+1} \cdots w_j, \text{Root} \Rightarrow s | \mathcal{G}] := \max_{A \in \mathcal{N}} \mu(A, i, j)$ is the marginal probability of span $w_i w_{i+1} \cdots w_j$ under non-terminal A.

Marginal probabilities are computed by Inside-Outside algorithm (Baker, 1979), with the inside probabilities $\alpha(A, i, j)$ and the outside probabilities $\beta(A, i, j)$ computed by the following recursion

$$\begin{aligned} &\alpha(A, i, j) \\ &= \sum_{B,C} \sum_{k=i}^{j-1} \Pr[A \to BC] \alpha(B, i, k) \alpha(C, k+1, j), \quad (2) \\ &\beta(A, i, j) \\ &= \sum_{B,C} \sum_{k=1}^{i-1} \Pr[B \to CA] \alpha(C, k, i-1) \beta(B, k, j) \quad (3) \\ &\quad + \sum_{B,C} \sum_{k=j+1}^{L} \Pr[B \to AC] \alpha(C, j+1, k) \beta(B, i, k) \end{aligned}$$

with the base cases $\alpha(A, i, i) = \Pr[A \to w_i]$ for all A, i and $\beta(\text{Root}, 1, L) = 1$ for all A. The marginal probabilities are then computed as

$$\mu(A, i, j) = \alpha(A, i, j) \times \beta(A, i, j). \quad (4)$$

Parsing performance is evaluated by two types of unlabelled F1 scores, which depend on the average method: Sentence F1 (average of F1 scores for each sentence) and Corpus F1 (considers total true positives, false positives, and false negatives).

2.3 Probing

A probe $f(\cdot)$ is a supervised model that predicts a target $\operatorname{tar}(\boldsymbol{x})$ for a given input $\boldsymbol{x}$ (Alain and Bengio, 2017; Hupkes et al., 2018; Conneau et al., 2018). As an example, Hewitt and Manning (2019) used a probe $f(\cdot)$ to predict the tree distance $\operatorname{tar}(i, j) = d_{\mathcal{T}}(i, j)$ between words in a dependency parse tree $\mathcal{T}$. Although mathematically equivalent, probes and supervised models have different goals. The latter aims for high prediction scores, while the former seeks to identify certain intrinsic information in embeddings (Maudslay et al., 2020; Chen et al., 2021). Probes should be limited to only detect the desired information, with low performance on uncontextualized embeddings and high performance on contextualized ones.

3 Parsing using Transformers

We design transformers with moderate layers and heads for parsing and masked language modeling. In Section 3.1, we prove that transformers can execute the Inside-Outside algorithm for bounded-length sentences with any PCFG. In Section 3.2, we connect our construction with masked language modeling and demonstrate the optimality of the Inside-Outside algorithm for MLM on PCFG-generated data. Finally, in Section 3.3, we demonstrate the ability to reduce the size of these constructions while retaining their parsing performance.

3.1 Transformers can execute Inside-Outside algorithm

We first give a construction (Theorem 3.1) that relies on *hard attention*, where only one of the attended positions will have positive attention score. For this construction, we define $f_{\text{attn}} : \mathbb{R}^{L \times d} \times \mathbb{R}^d$ such that the attention scores in eq. 1 are given by

$$a_{i,j}^h = \operatorname{ReLU}((\boldsymbol{K}_h \boldsymbol{e}_j^{(\ell)})^\top \boldsymbol{Q}_h \boldsymbol{e}_i^{(\ell)}). \quad (5)$$

This is similar to softmax attention used in practice, with softmax replaced by ReLU activation.

Theorem 3.1 (Hard attention)**.** *There exists a model with hard attention modules (5), $(4|\mathcal{N}|+1)L$ embeddings, $2L-1$ layers, and $4|\mathcal{N}|$ attention heads in each layer that simulates the Inside-Outside algorithm on all sentences with length at most L generated by PCFG $\mathcal{G} = (\mathcal{N}, \mathcal{I}, \mathcal{P}, n, p)$ and embed all inside and outside probabilities.*

Proof sketch. We give the proof sketch and defer details to Appendix B.1. The core idea is to use the first L layers to compute the inside probabilities with the recursive eq. 2. Each layer $\ell \leq L$ computes $\alpha(A, i, j)$ for all position pairs (i, j) with $j - i = \ell$ and all non-terminals A. The next L layers compute the outside probabilities with the recursive eq. 3. Each layer $L + \ell > L$ computes $\beta(A, i, j)$ for all position pairs (i, j) with $j - i = L - \ell$ and all non-terminals A.

At any position i in a layer $\ell \le L$, the input token embeds inside probabilities of all spans with a maximum length of ℓ, starting and ending at i: $\alpha(A, i, j)$ and $\alpha(A, k, i)$ for all non-terminals A and position tuples (i, j, k) where $j - i < \ell$, $i - k < \ell$. To compute $\alpha(A, i, i + \ell)$ at each position i for each non-terminal A, we use an attention head that calculates an inner product between the embeddings at positions i and $i+\ell$, weighted by the matrix containing $\Pr[A \to BC]_{B,C \in \mathcal{N}}$. The token at position i attends only to the token at $i+\ell$ thanks to the position embeddings and hard attention. We use another attention head to compute $\alpha(A, i-\ell, i)$, and store the new inside probability terms along with the previous ones in the embeddings. We use a similar technique to compute the outside probabilities in the next L layers. In layer $L + \ell$, we use two attention heads to compute $\beta(A, i, i + L - \ell)$ for each non-terminal A and position i, as there are two terms to compute in 3. We use two additional attention heads to compute $\beta(A, i - L + \ell, i)$, resulting in four attention heads for each non-terminal. $\square$

To further reduce embedding size and attention heads, we introduce relative positions and use soft attention. We introduce $2L + 1$ relative position vectors $\{p_t \in \mathbb{R}^d\}_{-L \le t \le L}$, and relative position biases $\{b_{t,\ell} \in \mathbb{R}\}_{-L \le t \le L, 1 \le \ell \le 2L-1}$ that modify the key vectors depending on the relative position of the query and key tokens. For an attention head h in layer ℓ, the attention score $a^h_{i,j}$ is given by

$$a^h_{i,j} = \mathrm{ReLU}(\boldsymbol{K}_h \boldsymbol{e}^{(\ell)}_j + p_{j-i} - b_{j-i,\ell})^\top \boldsymbol{Q}_h \boldsymbol{e}^{(\ell)}_i. \quad (6)$$

Theorem 3.2 (Relative positional embeddings)**.** *There exists a model with attention module (6), $2|\mathcal{N}|L+1$ embeddings, $2L-1$ layers, and $|\mathcal{N}|$ attention heads in each layer that simulate the Inside-Outside algorithm on all sentences with length at most L generated by PCFG $\mathcal{G} = (\mathcal{N}, \mathcal{I}, \mathcal{P}, n, p)$ and embed all inside and outside probabilities.*

The proof is deferred to Appendix B.2. Theorem 3.2 uses one attention head to compute layer-wise inside/outside probabilities per non-terminal, and only requires $|\mathcal{N}|$ heads in each layer. Once we have the inside and outside probabilities for spans, we can directly build the parse tree using the Labelled-Recall algorithm, which acts as a "probe" on the contextual representations of the model.

3.2 Masked language modeling for PCFG

The Inside-Outside algorithm not only can parse but also has a connection to masked language modeling (MLM), the pre-training loss used by BERT. The following theorem shows that, if the language is generated from a PCFG, then the Inside-Outside algorithm achieves the optimal MLM loss.

Theorem 3.3. *Assuming language is generated from a PCFG, the Inside-Outside algorithm reaches the optimal MLM loss.*

The Inside-Outside algorithm optimizes MLM loss on PCFG data, suggesting that pre-training on such data enables implicit learning of the algorithm or its computed quantities. Consequently, intermediate layers can capture syntactic information for parsing, potentially explaining the presence of structural information in language models (Hewitt and Manning, 2019; Vilares et al., 2020; Arps et al., 2022). We validate this conjecture in Section 4.3.

3.3 Towards realistic size

For PCFG learned on the `PTB` training set (`PTB` sections 02-21) with an average sentence length of 25 (Peng, 2021), Section 3.1 requires 1600 attention heads, $3200L$ embedding dimensions, and $2L$ layers to simulate the Inside-Outside algorithm for sentences of length L, which is much larger than BERT. However, by utilizing the inherent sparsity in the English PCFG, we can reduce the number of attention heads and the width of the embeddings while maintaining decent parsing performance. The details are deferred to Appendix C.

First ingredient: finding important non-terminals In the constructions of Theorems 3.1 and 3.2, the number of attention heads and embedding dimensions depend on the number of non-terminals of the PCFG. Thus if we can find a smaller PCFG, we can make the model much smaller. Specifically, if we only compute the probabilities of a specific set of in-terminals $\tilde{\mathcal{I}}$ and pre-terminals $\tilde{\mathcal{P}}$ in eq. 2 and 3, we can reduce the number of attention heads from $|\mathcal{N}|$ to $\max\{|\tilde{\mathcal{I}}|, |\tilde{\mathcal{P}}|\}$.[1]

We sort the non-terminals in terms of their frequency of occurrence in the `PTB` training set and show that restricting the Inside-Outside computation to a few frequent non-terminals has a negligible drop in performance (Table 1). The parsing score is still highly non-trivial, since the naive baseline, Right Branching (RB), can only get $< 40\%$ sentence and corpus F1 scores on `PTB` dataset.

[1] When $|\tilde{\mathcal{P}}| < c|\tilde{\mathcal{I}}|$, we can simulate the computations in the final layer using c layers with $|\tilde{\mathcal{I}}|$ heads instead of $|\tilde{\mathcal{P}}|$ heads. Additionally, we can decrease the embedding size by only storing probabilities for relevant non-terminals.

Approximation	Corpus F1	Sent F1	ppl.
No approx.	75.90	78.77	50.80
$\lvert\tilde{\mathcal{I}}\rvert = 10, \lvert\tilde{\mathcal{P}}\rvert = 45$	57.14	60.32	59.57
$\lvert\tilde{\mathcal{I}}\rvert = 20, \lvert\tilde{\mathcal{P}}\rvert = 45$	68.41	71.91	55.16
$\lvert\tilde{\mathcal{I}}\rvert = 40, \lvert\tilde{\mathcal{P}}\rvert = 45$	72.45	75.43	54.09

Table 1: Restricting computations of the Inside-Outside algorithm to the most frequent in(pre)-terminal subsets $\tilde{\mathcal{I}}$ ($\tilde{\mathcal{P}}$) in the PTB sections 02-21. We report the unlabelled F1 scores on PTB section 22 and the 1-masking perplexity on 200 sentences generated from the PCFG. $|\tilde{\mathcal{I}}| = 20, |\tilde{\mathcal{P}}| = 45$ resulted in a 8.58% increase in perplexity and 8.71% decrease in parsing F1 scores.

Approximation	Corpus F1	Sent F1	ppl.
$\lvert\tilde{\mathcal{I}}\rvert = 10, \lvert\tilde{\mathcal{P}}\rvert = 45$	57.14	60.32	59.57
$\lvert\tilde{\mathcal{I}}\rvert = 20, \lvert\tilde{\mathcal{P}}\rvert = 45$	68.41	71.91	55.16
$k^{(\ell)} = 10, \lvert\tilde{\mathcal{I}}\rvert = 20, \lvert\tilde{\mathcal{P}}\rvert = 45$	61.72	65.31	57.05
$k^{(\ell)} = 15, \lvert\tilde{\mathcal{I}}\rvert = 20, \lvert\tilde{\mathcal{P}}\rvert = 45$	68.20	71.33	55.52

Table 2: Approximate Inside-Outside algorithm using linear transformations $\{\boldsymbol{W}^{(\ell)} \in \mathbb{R}^{k^{(\ell)} \times |\tilde{\mathcal{I}}|}\}$ on the inside/outside probabilities of the selected subset $\tilde{\mathcal{I}}$. We report the F1 scores on PTB section 22 and the 1-masking perplexity on 200 sentences generated from the PCFG. Applying linear transformations can further reduce the number of attention heads in the constructed model to 15 starting from 20 frequent non-terminals subset $\tilde{\mathcal{I}}$, while only changing the performance by at most 1%.

Second ingredient: utilizing structures across non-terminals We still use one attention head to represent the computation for a specific non-terminal, which does not utilize possible underlying correlations between different non-terminals. Specifically, for Theorem 3.2, we use one attention head at layer $\ell < L$ to compute the inside probabilities $\alpha(A, i, j)$ with $j - i = \ell$. If $\alpha(A, i, j)$ for different non-terminals $A \in \tilde{\mathcal{I}}$ lie in a $k^{(\ell)}$-dimensional subspace with $k^{(\ell)} < |\tilde{\mathcal{I}}|$, we can compute all of the inside probabilities using only $k^{(\ell)}$ attention heads by computing the vector $\boldsymbol{W}^{(\ell)} \boldsymbol{\alpha}(i, j)$, where $\boldsymbol{W}^{(\ell)} \in \mathbb{R}^{k^{(\ell)} \times |\tilde{\mathcal{I}}|}$ is the transformation matrix and $\boldsymbol{\alpha}(i, j) \in \mathbb{R}^{|\tilde{\mathcal{I}}|}$ is the concatenation of all inside probabilties $\alpha(A, i, j)_{A \in \tilde{\mathcal{I}}}$. The same procedure can also be applied to the computation of outside probabilities. [2] Although the probabilities should not lie in a low dimensional subspace in reality, we can still try to learn a transformation matrix $\boldsymbol{W}^{(\ell)} \in \mathbb{R}^{k^{(\ell)} \times |\tilde{\mathcal{I}}|}$ and approximately compute the inside probabilities by $\boldsymbol{\alpha}(i, j) = (\boldsymbol{W}^{(\ell)})^{\dagger} \boldsymbol{W}^{(\ell)} \boldsymbol{\alpha}^*(i, j)$ for $j - i = \ell$, where $\boldsymbol{\alpha}^*(i, j)$ denotes the Inside probabilities for non-terminals in $\tilde{\mathcal{I}}$. Please refer to Appendix C.4 for more details.

Learning the transformations For sentence s and a span with length $\ell + 1$, we compute the marginal probabilities of this span $\boldsymbol{\mu}_s^{i,j} \in \mathbb{R}^{|\tilde{\mathcal{I}}|}$, that contains $\mu(A, i, j)$ for each non-terminal $A \in \tilde{\mathcal{I}}$. We then compute the normalized correlation matrix $\boldsymbol{X}^{(\ell)} = \sum_s \boldsymbol{X}_s^{(\ell)} / \|\boldsymbol{X}_s^{(\ell)}\|_{\mathrm{F}}$, where $\boldsymbol{X}_s^{(\ell)} = \sum_{i,j:j-i=\ell} \boldsymbol{\mu}_s^{i,j} (\boldsymbol{\mu}_s^{i,j})^{\top}$, which captures the correlation of $\tilde{\mathcal{I}}$ for spans with length $\ell + 1$ in the entire corpus. We apply the Eigen-decomposition on $\boldsymbol{X}_\ell$ and set $\boldsymbol{W}^{(\ell)}$ as the top $k^{(\ell)}$ Eigen-vectors.

The parsing results and 1-masking perplexity using $\{\boldsymbol{W}^{(\ell)}\}_{\ell \le L}$ with different $k^{(\ell)}$ are shown in Table 2. Utilizing the linear transformations, we obtain 71.33% and 65.31% sentence F1 on PTB with only 15 and 10 attention heads respectively, whereas only computing probabilities for top-10 in-terminals gives 60.32% sentence F1 on PTB. The following theorem summarizes the results.

[2] The computation for $A \in \tilde{\mathcal{P}}$ needs $|\tilde{\mathcal{P}}|$ heads in the last layer and can be simulated by several layers with fewer heads.

Theorem 3.4 (Informal)**.** *There exists a model with attention module (6), $275 + 40L$ embeddings, $2L + 1$ layers, and 15 attention heads in each layer that can approximately execute Inside-Outside algorithm on all sentences with length at most L generated by English PCFG, introducing 8.6% increase in average 1-mask perplexity and resulting in at most 9.45% drop in the parsing performance of Labeled-Recall algorithm.*

4 Probing Masked Language Models for Parsing Information

Section 3 shows that transformers can execute the Inside-Outside algorithm and contain syntactic information in their intermediate states. These results are existential, and it is unclear if models pre-trained under MLM possess similar information.

One difficulty in answering this question is that syntactic probes on BERT-like models may leverage semantic cues to parse. To address this concern, we pre-train multiple RoBERTa models on synthetic datasets derived from English PCFG (Section 4.1), which eliminates semantic dependencies. We then probe the models for parse tree construction (Section 4.2) and marginal probabilities (Section 4.3) to verify if they capture information computed by the Inside-Outside algorithm.

4.1 Pre-training on PCFG

We pre-train RoBERTa models with varying attention heads and layers on synthetic PCFG data. We

Model	Training ppl.	Validation ppl.
A12L12	106.16	106.68
A12L1	111.8	110.57
A12L3	108.09	105.79
A12L6	105.78	104.58
A3L12	120.52	117.39
A24L12	106.28	104.5

Table 3: Perplexity of different models trained on synthetic PCFG data. AiLj refers to a model with i attention heads and j layers. Except for models with few layers (A12L1) and few attention heads (A3L12), trained models have nearly the same perplexity.

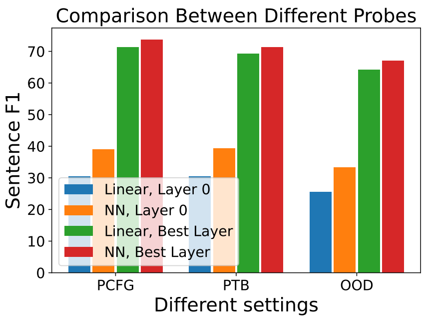


Figure 1: Comparison between different probes (linear or a 2-layer neural net) under different settings. 2-layer probes achieve better parsing performance, compared to linear probes. The large performance gap of the probes on layer 0's embeddings from A12L12 and the best layer shows the existence of meaningful syntactic information in the contextualized embeddings.

denote the models with AiLj, where i and j indicate the number of attention heads and layers, respectively. Additional pre-training details are available in Appendix A.1. Table 3 shows the perplexity for various models. We find that except for models with too few layers (A12L1) and too few attention heads (A3L12), other models have nearly the same perplexity. Further increasing depth and number of heads does not appear to improve the result.

4.2 Probing for constituency parse trees

We probe the language models pre-trained on synthetic PCFG data and show that these models indeed capture the "syntactic information", in particular, the structure of the constituency parse trees.

Experiment setup We mostly follow the probing procedure in Vilares et al. (2020); Arps et al. (2022) that predicts the relative depth of the common ancestors between different token pairs and then constructs the constituency tree. Given a sentence $w_1 w_2 \dots w_L$ with parse tree T, we denote depth$(i, i+1)$ the depth of the least common ancestor of w_i, w_{i+1} in the parse tree T, we want to find a probe $f^{(\ell)}$ to predict the relative depth tar(i) = depth$(i, i+1)$ − depth$(i-1, i)$ for position i. In Vilares et al. (2020), the probe $f^{(\ell)}$ is linear, and the input to the probe $f^{(\ell)}$ at position i is the concatenation of the embeddings at position i and the BOS (or EOS) token. Besides the linear probe $f^{(\ell)}$, we also experiment with the probe where $f^{(\ell)}$ is a 2-layer neural network with 16 hidden neurons. We consider three settings for probing: train and test the probe on synthetic PCFG data (`PCFG`); train and test on `PTB` dataset (`PTB`); and train on the synthetic PCFG data while test on `PTB` (out of distribution, OOD). The OOD setting serves as a baseline for a syntactic probe on `PTB` since semantic relations do not appear in the pre-trained model or the probe.

Experiment results Figure 1 reveals a substantial difference between the probing outcomes of layer 0 embeddings and those of the best layer in all settings. Both probing approaches profit greatly from the representations of subsequent layers.

Table 4 shows probing results for different settings (`PCFG`, `PTB`, and OOD), different probes (linear or a 2-layer neural net) on different models. Except for A12L1 and A3L12, the linear and neural net probes give decent parsing scores ($> 70\%$ sentence F1 for neural net probes) in both `PCFG` and `PTB` settings. As for the OOD setting, the performances achieved by the best layer drop by about 5% compared with `PCFG` and `PTB`, but they are still much better than the performance achieved by the 0-th layer embeddings. In this setting, there is no semantic information even in the probe itself and thus gives a baseline for the probes on `PTB` dataset that only uses syntactic information. As a comparison, the naive baseline, Right-branching (RB), reaches $< 40\%$ for both sentence and corpus F1 score (Li et al., 2020) on `PTB` dataset, and if we use layer 0's embeddings to probe, the sentence F1 is $< 41\%$ in all settings for all models. Our positive results on syntactic parsing support the claim that pre-training language models using MLM loss can indeed capture the structural information of the underlying constituency parse tree.

4.3 Probing for the marginal probabilities

Section 4.2 verifies that language models can capture structure information of the parse trees, but we still don't know if the model executes the Inside-Outside algorithm proposed in Sections 3.1 and 3.2.

			IO	A12L12	A12L1	A12L3	A12L6	A3L12	A24L12
Linear	PCFG	Sent. F1	81.61	**71.34**	63.16	69.96	**71.23**	64.71	**70.76**
		Corpus F1	71.65	**63.01**	54.24	61.54	**62.57**	55.36	**62.56**
	PTB	Sent. F1	78.77	**69.31**	62.99	68.22	68.13	61.56	**68.79**
		Corpus F1	75.90	**65.01**	59.96	**65.21**	**65.01**	58.31	**65.97**
	OOD	Sent. F1	81.61	**64.26**	57.96	63.22	**63.89**	58.00	**63.88**
		Corpus F1	71.65	**60.98**	54.29	59.79	**60.58**	54.39	**60.62**
2-layer NN	PCFG	Sent. F1	81.61	**73.71**	64.80	72.62	**73.60**	62.55	**73.27**
		Corpus F1	71.65	**66.18**	57.16	**65.36**	**66.01**	53.36	**65.92**
	PTB	Sent. F1	78.77	**71.32**	64.89	70.15	**70.33**	63.23	**70.59**
		Corpus F1	75.90	**68.07**	62.09	**67.25**	**67.31**	60.59	**67.93**
	OOD	Sent. F1	81.61	**66.99**	59.89	**66.21**	**66.56**	57.60	**67.18**
		Corpus F1	71.65	**63.89**	56.74	63.30	63.81	54.60	**64.54**

Table 4: Parsing results for different models under different settings using Linear and 2-layer neural net probes, when compared to Inside-Outside algorithm (IO). We report the best F1 score achieved using any of the layer's embeddings. Scores within 1% of the max (except IO) in each row are highlighted. Models except A12L1 and A3L12 give decent parsing F1 scores, and models with more layers or heads tend to get better F1 scores in general.

Span Length	A12L12	A12L1	A12L3	A12L6	A3L12	A24L12
$\ell = 2$	.88 / **.93**	.83 / .88	.88 / .91	.88 / **.92**	.86 / .88	.87 / **.92**
$\ell = 3$	.79 / **.90**	.74 / .84	.80 / .88	.79 / **.89**	.77 / .84	.79 / **.89**
$\ell = 4$	.69 / **.86**	.65 / .77	.69 / .82	.69 / .84	.66 / .78	.69 / **.85**
$\ell = 5$	.62 / .79	.57 / .70	.62 / .77	.61 / **.81**	.58 / .69	.62 / .79
$\ell = 10$	.51 / **.77**	.48 / .68	.51 / .75	.51 / **.78**	.51 / .61	.51 / .73

Table 5: Probing for the "normalized" marginal probabilities of spans at different lengths on different pre-trained models. We report the Pearson correlation between the predicted probabilities and the span marginal probabilities computed by the Inside-Outside algorithm on PTB datasets, for both the linear and the 2-linear net probes (separated by /). The high correlation indicates that the MLM pre-trained models approximately encode the marginal span probabilities of the Inside-Outside algorithm during pre-training.

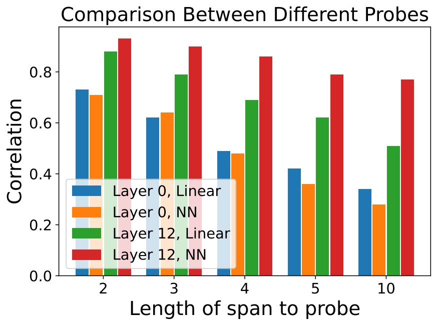
In this subsection, we test if model representations can be used to predict marginal probabilities computed in the Inside-Outside algorithm.

Experiment setup We train a probe to predict the normalized marginal probabilities for spans with a specific length. Fix the span length ℓ, for each sentence $w_1 w_2 \dots w_L$, denote $\boldsymbol{e}_1, \boldsymbol{e}_2, \dots, \boldsymbol{e}_L$ the embeddings from the last layer of the pre-trained language model. We want to find a probe $f^{(\ell)}$ such that for each span $[i, i+\ell-1]$ with length ℓ, the probe $f^{(\ell)}([\boldsymbol{e}_i; \boldsymbol{e}_{i+\ell-1}])$ predicts the normalized marginal probability of span $[i, i+\ell-1]$, i.e. $\text{tar}(i, i+\ell-1) = s(i, i+\ell-1)/\max_{j,j'} s(j, j')$, where $s(i,j) = \max_A \mu(A, i, j)$ is the marginal probability of span $[i, j]$ and $\mu(A, i, j)$ is given by eq. 4. The input to the probe $[\boldsymbol{e}_i; \boldsymbol{e}_{i+\ell-1}] \in \mathbb{R}^{2d}$ is the concatenation of $\boldsymbol{e}_i$ and $\boldsymbol{e}_{i+\ell-1}$. To test the sensitivity of our probe, we also take the embeddings from the 0-th layer as input to the probe $f^{(\ell)}$.

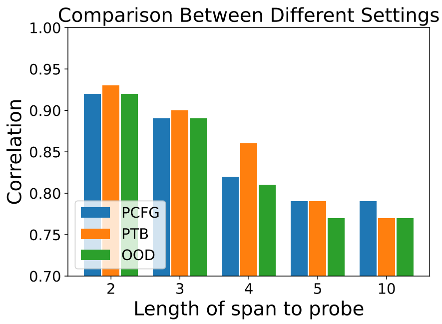
We give two options for the probe $f^{(\ell)}$: (1) linear, and (2) a 2-layer neural network with 16 hidden neurons, since the relation between the embeddings and the target may not be a simple linear function. Similar to the Section 4.2, we also consider three settings: PCFG, PTB, and OOD.

Experiment results Figure 2a reports the correlation between the span marginal probabilities and the predictions of the 4 different probes for A12L12 model. For both linear and 2-layer neural net probes, changing the input from layer 0 to layer 12 drastically increases the predicted correlation, which again suggests that the uncontextualized embeddings don't contain enough information about the marginal probabilities. Besides, the neural net can predict better on layer 12 embeddings, but performs nearly the same on layer 0, suggesting that the neural network is a better probe in this setting.

Figure 2b compares the probing results under three different settings. Surprisingly, we find that the probe can achieve high correlation with the real marginal probabilities under all settings. Furthermore, we observe that there is almost no drop in performance when changing the test dataset from PCFG to PTB (PCFG setting and OOD setting). This result implies that the probe, along with the embeddings, indeed contains the syntactic information computed by the Inside-Outside algorithm and is not overfitting to the training dataset.



(a) Compare linear/2-layer NN probes under `PTB` setting. We observe: (a) 2-layer NN probe has better performance, and (b) the probes give better performance on 12th-layer embeddings.



(b) Performance of 2-layer neural net probe on the 12-th layer embeddings under different settings. The closer correlation performance of the probe across settings (including OOD) indicates true marginal probabilities captured by the trained probe.

Figure 2: Comparison between different probes for marginal probabilities on the A12L12 model. The y-axis denotes correlation between the prediction and the target, and the x-axis denotes probes for different lengths.

Table 5 shows the probing results on different pre-trained models. The results show that the neural network probe is highly correlated with the target for most pre-trained models, except for A12L1 and A3L12 models. Surprisingly, even for length 10 spans, the neural network probe still achieves an F1 score of up to 78% for the best model. The high correlation suggests that the pre-trained models contain certain syntactic information computed by the Inside-Outside algorithm. Overall, the results indicate that MLM training may incentivize the model to approximate the Inside-Outside algorithm, thus validating our constructions in Section 3.

5 Related Works

(Structural) probing Several recent works on probing have aimed to study the encoded information in BERT-like models (Rogers et al., 2020). Hewitt and Manning (2019); Reif et al. (2019); Manning et al. (2020); Vilares et al. (2020); Maudslay et al. (2020); Maudslay and Cotterell (2021); Chen et al. (2021); Arps et al. (2022); Jawahar et al. (2019) have demonstrated that it is possible to predict various syntactic information present in the input sequence, including parse trees or POS tags, from internal states of BERT. In contrast to existing approaches that commonly employ a model pre-trained on natural language, we pre-train our model under PCFG-generated data to investigate the interplay between the data, the MLM objective, and the architecture's capacity for parsing. Besides syntax, probing has also been used to test other linguistic structures like semantics, sentiment, etc. (Belinkov et al., 2017; Reif et al., 2019; Kim et al., 2020; Richardson et al., 2020; Vulić et al., 2020; Conia and Navigli, 2022).

Expressive power of transformers Yun et al. (2020a,b) show that transformers are universal sequence-to-sequence function approximators. Later, Pérez et al. (2021); Bhattamishra et al. (2020b) show that attention models can simulate Turing machines, with Wei et al. (2022) proposing statistically meaningful approximations of Turing machines. To understand the behavior of moderate-size transformer architectures, many works have investigated specific classes of languages, e.g. bounded-depth Dyck languages (Yao et al., 2021), modular prefix sums (Anil et al., 2022), adders (Nanda et al., 2023), regular languages (Bhattamishra et al., 2020a), and sparse logical predicates (Edelman et al., 2022). Merrill et al. (2022) relate saturated transformers with constant depth threshold circuits, and Liu et al. (2022) provide a unified theory on understanding automata within transformers. These works study the expressive power under a class of synthetic language. Compared to the prior works, our results are more related to the natural language, as we consider not only a class of synthetic language (PCFG), but also a specific PCFG tailored to the natural language.

6 Conclusion

In this work, we show that MLM with moderate size has the capacity to parse decently well. We probe BERT-like models pre-trained (with MLM loss) on the synthetic text generated using PCFGs to verify that these models capture syntactic information. Furthermore, we show that the models contain the marginal span probabilities computed by the Inside-Outside algorithm, thus connecting MLM and parsing. We hope our findings may yield new insights into large language models and MLM.

References

Guillaume Alain and Yoshua Bengio. 2017. Understanding intermediate layers using linear classifier probes.

Cem Anil, Yuhuai Wu, Anders Andreassen, Aitor Lewkowycz, Vedant Misra, Vinay Ramasesh, Ambrose Slone, Guy Gur-Ari, Ethan Dyer, and Behnam Neyshabur. 2022. Exploring length generalization in large language models. *arXiv preprint arXiv:2207.04901*.

David Arps, Younes Samih, Laura Kallmeyer, and Hassan Sajjad. 2022. Probing for constituency structure in neural language models. *arXiv preprint arXiv:2204.06201*.

Jiangang Bai, Yujing Wang, Yiren Chen, Yaming Yang, Jing Bai, Jing Yu, and Yunhai Tong. 2021. Syntax-bert: Improving pre-trained transformers with syntax trees. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 3011–3020.

James K Baker. 1979. Trainable grammars for speech recognition. *The Journal of the Acoustical Society of America*, 65(S1):S132–S132.

Yonatan Belinkov, Nadir Durrani, Fahim Dalvi, Hassan Sajjad, and James Glass. 2017. What do neural machine translation models learn about morphology? *arXiv preprint arXiv:1704.03471*.

Satwik Bhattamishra, Kabir Ahuja, and Navin Goyal. 2020a. On the ability and limitations of transformers to recognize formal languages. *arXiv preprint arXiv:2009.11264*.

Satwik Bhattamishra, Arkil Patel, and Navin Goyal. 2020b. On the computational power of transformers and its implications in sequence modeling. In *Proceedings of the 24th Conference on Computational Natural Language Learning*, pages 455–475.

Boli Chen, Yao Fu, Guangwei Xu, Pengjun Xie, Chuanqi Tan, Mosha Chen, and Liping Jing. 2021. Probing bert in hyperbolic spaces. In *International Conference on Learning Representations*.

Shay B Cohen, Karl Stratos, Michael Collins, Dean Foster, and Lyle Ungar. 2012. Spectral learning of latent-variable pcfgs. In *Proceedings of the 50th annual meeting of the association for computational linguistics (Volume 1: Long papers)*, pages 223–231.

Shay B Cohen, Karl Stratos, Michael Collins, Dean P Foster, and Lyle Ungar. 2014. Spectral learning of latent-variable pcfgs: Algorithms and sample complexity. *The Journal of Machine Learning Research*, 15(1):2399–2449.

Simone Conia and Roberto Navigli. 2022. Probing for predicate argument structures in pretrained language models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4622–4632, Dublin, Ireland. Association for Computational Linguistics.

Alexis Conneau, German Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. 2018. What you can cram into a single $&!#* vector: Probing sentence embeddings for linguistic properties. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2126–2136, Melbourne, Australia. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Benjamin L Edelman, Surbhi Goel, Sham Kakade, and Cyril Zhang. 2022. Inductive biases and variable creation in self-attention mechanisms. In *International Conference on Machine Learning*, pages 5793–5831. PMLR.

Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Nova DasSarma, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. 2021. A mathematical framework for transformer circuits. *Transformer Circuits Thread*. Https://transformer-circuits.pub/2021/framework/index.html.

Joshua Goodman. 1996. Parsing algorithms and metrics. In *34th Annual Meeting of the Association for Computational Linguistics*, pages 177–183.

John Hewitt and Percy Liang. 2019. Designing and interpreting probes with control tasks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2733–2743.

John Hewitt and Christopher D Manning. 2019. A structural probe for finding syntax in word representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4129–4138.

Dieuwke Hupkes, Sara Veldhoen, and Willem Zuidema. 2018. Visualisation and'diagnostic classifiers' reveal how recurrent and recursive neural networks process hierarchical structure. *Journal of Artificial Intelligence Research*, 61:907–926.

Peter Izsak, Moshe Berchansky, and Omer Levy. 2021. How to train BERT with an academic budget. In

Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics.

Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. 2019. What does bert learn about the structure of language? In *ACL 2019-57th Annual Meeting of the Association for Computational Linguistics*.

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. 2020. Are pre-trained language models aware of phrases? simple but strong baselines for grammar induction. In *International Conference on Learning Representations*.

Jun Li, Yifan Cao, Jiong Cai, Yong Jiang, and Kewei Tu. 2020. An empirical comparison of unsupervised constituency parsing methods. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3278–3283.

Bingbin Liu, Jordan T Ash, Surbhi Goel, Akshay Krishnamurthy, and Cyril Zhang. 2022. Transformers learn shortcuts to automata. *arXiv preprint arXiv:2210.10749*.

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*.

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

Christopher D Manning, Kevin Clark, John Hewitt, Urvashi Khandelwal, and Omer Levy. 2020. Emergent linguistic structure in artificial neural networks trained by self-supervision. *Proceedings of the National Academy of Sciences*, 117(48):30046–30054.

Mitch Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*, 19(2):313–330.

Rowan Hall Maudslay and Ryan Cotterell. 2021. Do syntactic probes probe syntax? experiments with jabberwocky probing. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 124–131.

Rowan Hall Maudslay, Josef Valvoda, Tiago Pimentel, Adina Williams, and Ryan Cotterell. 2020. A tale of a probe and a parser. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7389–7395.

William Merrill, Ashish Sabharwal, and Noah A Smith. 2022. Saturated transformers are constant-depth threshold circuits. *Transactions of the Association for Computational Linguistics*, 10:843–856.

Neel Nanda, Lawrence Chan, Tom Liberum, Jess Smith, and Jacob Steinhardt. 2023. Progress measures for grokking via mechanistic interpretability. *arXiv preprint arXiv:2301.05217*.

Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova DasSarma, Tom Henighan, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Scott Johnston, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. 2022. In-context learning and induction heads. *Transformer Circuits Thread*. Https://transformer-circuits.pub/2022/in-context-learning-and-induction-heads/index.html.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Haoran Peng. 2021. Spectral learning of latent-variable pcfgs: High-performance implementation.

Jorge Pérez, Pablo Barceló, and Javier Marinkovic. 2021. Attention is turing-complete. *J. Mach. Learn. Res.*, 22(75):1–35.

Emily Reif, Ann Yuan, Martin Wattenberg, Fernanda B Viegas, Andy Coenen, Adam Pearce, and Been Kim. 2019. Visualizing and measuring the geometry of bert. *Advances in Neural Information Processing Systems*, 32.

Kyle Richardson, Hai Hu, Lawrence Moss, and Ashish Sabharwal. 2020. Probing natural language inference models through semantic fragments. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8713–8721.

Anna Rogers, Olga Kovaleva, and Anna Rumshisky. 2020. A primer in bertology: What we know about how bert works. *Transactions of the Association for Computational Linguistics*, 8:842–866.

David Vilares, Michalina Strzyz, Anders Søgaard, and Carlos Gómez-Rodríguez. 2020. Parsing as pretraining. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 9114–9121.

Ivan Vulić, Edoardo Maria Ponti, Robert Litschko, Goran Glavaš, and Anna Korhonen. 2020. Probing pretrained language models for lexical semantics. *arXiv preprint arXiv:2010.05731*.

Colin Wei, Yining Chen, and Tengyu Ma. 2022. Statistically meaningful approximation: a case study on approximating turing machines with transformers. In *Advances in Neural Information Processing Systems*.

Alexander Wettig, Tianyu Gao, Zexuan Zhong, and Danqi Chen. 2022. Should you mask 15% in masked language modeling? *arXiv preprint arXiv:2202.08005*.

Zhiyong Wu, Yun Chen, Ben Kao, and Qun Liu. 2020. Perturbed masking: Parameter-free probing for analyzing and interpreting bert. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4166–4176.

Zenan Xu, Daya Guo, Duyu Tang, Qinliang Su, Linjun Shou, Ming Gong, Wanjun Zhong, Xiaojun Quan, Daxin Jiang, and Nan Duan. 2021. Syntax-enhanced pre-trained model. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 5412–5422.

Shunyu Yao, Binghui Peng, Christos Papadimitriou, and Karthik Narasimhan. 2021. Self-attention networks can process bounded hierarchical languages. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3770–3785.

Chulhee Yun, Srinadh Bhojanapalli, Ankit Singh Rawat, Sashank Reddi, and Sanjiv Kumar. 2020a. Are transformers universal approximators of sequence-to-sequence functions? In *International Conference on Learning Representations*.

Chulhee Yun, Yin-Wen Chang, Srinadh Bhojanapalli, Ankit Singh Rawat, Sashank Reddi, and Sanjiv Kumar. 2020b. O (n) connections are expressive enough: Universal approximability of sparse transformers. *Advances in Neural Information Processing Systems*, 33:13783–13794.

Appendix

A More Experiment Results

In this section, we provide more experiment results for RoBERTa pre-trained on PCFG-generated data. In Appendix A.2, we show more structural probing results related to the experiments in Section 4.2. In Appendix A.5, we do some simple analysis on the attention patterns for RoBERTa pre-trained on PCFG-generated data, trying to gain more understanding of the mechanism beneath large language models.

A.1 Details for pre-training

Experiment setup We generate 10^7 sentences for the training set from the PCFG, with an average length of 25 words. The training set is roughly 10% in size compared to the training set of the original RoBERTa which was trained on a combination of Wikipedia (2500M words) plus BookCorpus (800M words). We also keep a small validation set of 5×10^4 sentences generated from the PCFG to track the MLM loss. We follow (Izsak et al., 2021; Wettig et al., 2022) to pre-train all our models within a single day on a cluster of 8 RTX 2080 GPUs. Specifically, we train our models with AdamW (Loshchilov and Hutter, 2017) optimization, using 4096 sequences in a batch and hyperparameters $(\beta_1, \beta_2, \epsilon) = (0.9, 0.98, 10^{-6})$. We follow a linear warmup schedule for 1380 training steps with the peak learning rate of 2×10^{-3}, after which the learning rate drops linearly to 0 (with the max-possible training step being 2.3×10^4). We report the performance of all our models at step 5×10^3 where the loss seems to converge for all the models.

Architecture To understand the impact of different components in the encoder model, we pre-train different models by varying the number of attention heads and layers in the model. To understand the role of the number of layers in the model, we start from the RoBERTa-base architecture, which has 12 layers and 12 attention heads, and vary the number of layers to 1,3,6 to obtain 3 different architectures. Similariy, to understand the role of the number of attention heads in the model, we start from the RoBERTa-base architecture and vary the number of attention heads to 3 and 24 to obtain 2 different architectures.

Data generation from PCFG Strings are generated from the PCFG $\mathcal{G} = (\mathcal{N}, \mathcal{I}, \mathcal{P}, n, p)$ as follows: We always maintain a string $s_t \in ([n] \cup \mathcal{N})^*$ at step t. The initial string $s_1 =$ ROOT. At step t, if all characters in s_t belong to $[n]$, the generation process ends, and s_t is the resulting string. Otherwise, we pick a character $A \in s_t$ such that $A \in \mathcal{N}$. If $A \in \mathcal{P}$, we replace the character A to w with probability $\Pr[A \to w]$. If $A \in \mathcal{I}$, we replace the character A to two characters BC with probability $\Pr[A \to BC]$.

A.2 More results on constituency parsing

More details on probing experiments In Section 4.2, we mention that there are three settings: `PCFG`, `PTB`, and OOD. We generate two synthetic PCFG datasets according to the PCFG generation process: the first contains 10,000 sentences, which serves as the training set for probes, and the second contains 2,000 sentences, which serves as the test set for probes. As for the `PTB`, the training set for the probes consists of the first 10,000 sentences from sections 02-21, and we use `PTB` section 22 as the test set for the probes. In the `PCFG` setting, we train on the PCFG training set we generated, and test on the PCFG test set. In the `PTB` setting, we train on the `PTB` training set (10,000 sentences in sections 02-21) and test on the `PTB` test set (section 22). In the OOD setting, we train on the `PCFG` training set, while test on the `PTB` test set (section 22).

For the linear probe, we directly use Scikit-learn (Pedregosa et al., 2011). For the 2-layer NN probe, we train the neural net with Adam optimizer with learning rate $1e-3$. We optimize for 800 epochs, and we apply a multi-step learning rate schedule with milestones $200, 400, 600$ and decreasing factor 0.1. The batch size for Adam is chosen to be 4096.

Probing on embeddings from different layers In Section 4.2, we show the probing results on the embeddings either from 0-th layer or from the best layer (the layer that achieves the highest F1 score) of different pre-trained models. In this section, we show how the F1 score changes with different layers.

Figure 3 shows sentence F1 scores for linear probes $f(\cdot)$ trained on different layers' embeddings for different pre-trained models. We show the results under the `PCFG` and `PTB` settings. From Figure 3, we observe that using the embeddings from the 0-th layer can only get sentence F1 scores close to (or even worse than) the naive Right-branching baseline for all the pre-trained models. However, except for model A3L12, the linear probe can get at least 60% sentence F1 using the embeddings from layer 1. Then, the sentence F1 score increases as the layer increases, and gets nearly saturated at layer 3 or 4. The F1 score for the latter layers may be better than the F1 score at layer 3 or 4, but the improvement is not significant. The observations still hold if we change the linear probe to a neural network, consider the OOD setting instead of `PCFG` and `PTB`, or change the measurement from sentence F1 to corpus F1.

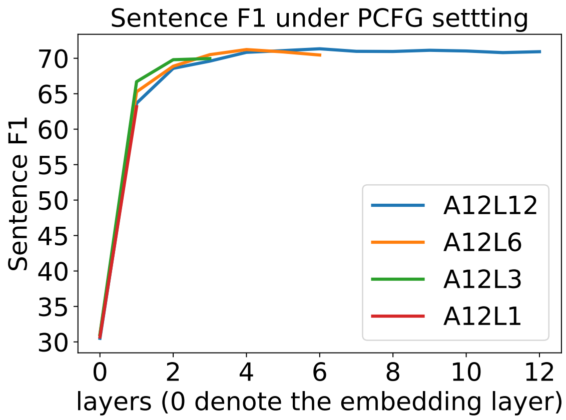
Our observations suggest that most of the constituency parse tree information can be encoded in the lower layers, and a lot of the parse tree information can be captured even in the first layer. Although our constructions (Theorems 3.1 and 3.2) and approximations (Theorems 3.4 and C.2) try to reduce the number of attention heads and the number of embedding dimensions close to the real language models, we don't know how to reduce the number of layers close to BERT or RoBERTa (although our number is acceptable since GPT-3 has 96 layers). More understanding of how language models can process such information in such a small number of layers is needed.

Comparison with probes using other input structures In Section 4.2, we train a probe $f(\cdot)$ to predict the relative depth $\mathrm{tar}(i) = \mathrm{depth}(i, i+1) - \mathrm{depth}(i-1, i)$, and the input to the probe f is the concatenation of the embedding $\boldsymbol{e}_i^{(\ell)}$ at position i and the embedding $\boldsymbol{e}_{\mathrm{EOS}}^{(\ell)}$ for the EOS token at some layer ℓ. Besides taking the concatenation $[\boldsymbol{e}_i^{(\ell)}; \boldsymbol{e}_{\mathrm{EOS}}^{(\ell)}]$ as the input structure of the probe, it is also natural to use the concatenation $[\boldsymbol{e}_{i-1}^{(\ell)}; \boldsymbol{e}_i^{(\ell)}; \boldsymbol{e}_{i+1}^{(\ell)}]$ to predict the relative depth $\mathrm{tar}(i)$. In this part, we compare the performances of probes with different input structures. We use EOS to denote the probe that takes $[\boldsymbol{e}_i^{(\ell)}; \boldsymbol{e}_{\mathrm{EOS}}^{(\ell)}]$ as the input and predicts the relative depth, while ADJ (Adjacent embeddings) to denote the probe the takes $[\boldsymbol{e}_{i-1}^{(\ell)}; \boldsymbol{e}_i^{(\ell)}; \boldsymbol{e}_{i+1}^{(\ell)}]$ as input.

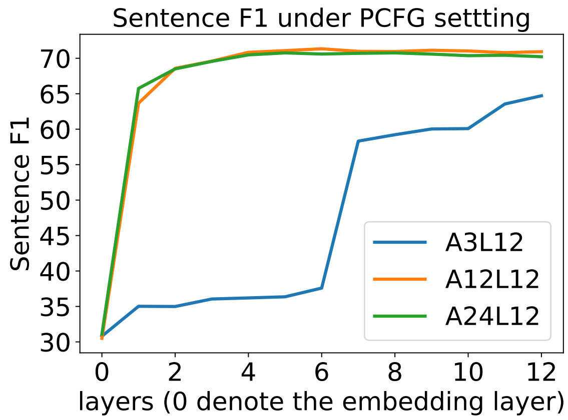
Figure 4 shows the probing results on A12L12, the model with 12 attention heads and 12 layers. We compare the probes with different inputs structure (EOS or ADJ), and the input embeddings come from different layers (the 0-th layer or the layer that achieves the best F1 score). We observe that: (1) the probes using ADJ input structure have better parsing scores than the probes using EOS input structure, and (2) the sentence F1 for the probes using the ADJ input structure is high even if the input comes from layer 0 of the model ($> 55\%$ for linear $f(\cdot)$ and $> 60\%$ for neural network $f(\cdot)$). Although the probe using ADJ has better parsing scores than the probe using EOS, it is harder to test whether it is a good probe, since the concatenation of adjacent embeddings $[\boldsymbol{e}_{i-1}^{(0)}; \boldsymbol{e}_i^{(0)}; \boldsymbol{e}_{i+1}^{(0)}]$ from layer 0 is already contextualized, and it is hard to find a good baseline to show that the probe is *sensitive* to the information we want to test. Thus, we choose to follow Vilares et al. (2020); Arps et al. (2022) and use the probe with input structure $[\boldsymbol{e}_i^{(\ell)}; \boldsymbol{e}_{\mathrm{EOS}}^{(\ell)}]$ in Section 4.2.

Nonetheless, the experiment results for probes taking $[\boldsymbol{e}_{i-1}^{(0)}; \boldsymbol{e}_i^{(0)}; \boldsymbol{e}_{i+1}^{(0)}]$ as input are already surprising: by knowing three adjacent word identities and their position (the token embedding $\boldsymbol{e}_i^{(0)}$ contains both the word embedding and the positional embedding) and train a 2-layer neural network on top of that, we can get $62.67\%, 63.91\%, 57.02\%$ sentence F1 scores under `PCFG`, `PTB`, and OOD settings respectively. As a comparison, the probe taking $[\boldsymbol{e}_i^{(\ell)}; \boldsymbol{e}_{\mathrm{EOS}}^{(\ell)}]$ as input (Vilares et al., 2020; Arps et al., 2022) only get $39.06\%, 39.31\%, 33.33\%$ sentence F1 under `PCFG`, `PTB`, and OOD settings respectively. It shows that lots of syntactic information (useful for parsing) can be captured by just using adjacent words without more context.

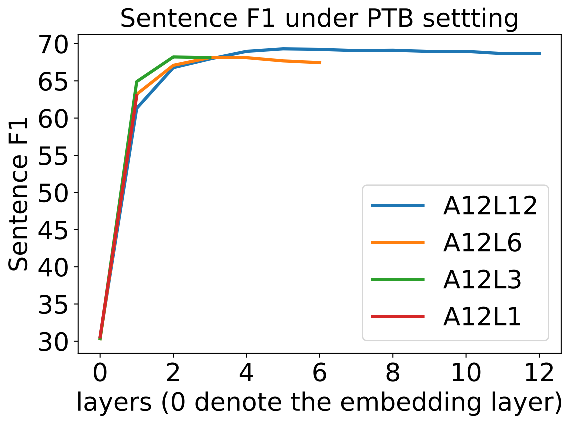
More discussion on probing measurement (Unlabelled) F1 score is the default performance measurement in the constituency parsing and syntactic probing literature. However, we would like to point out that only focusing on the F1 score may cause some bias. Because all the spans have equal weight when computing the F1 score, and most of the spans in a tree have a short length (if the parse tree is perfectly balanced, then length 2 spans consist of half of the spans in the parse tree), one can get a decently well F1 score by only getting correct on short spans. Besides, we also show that by taking the inputs $[\boldsymbol{e}_{i-1}^{(0)}; \boldsymbol{e}_i^{(0)}; \boldsymbol{e}_{i+1}^{(0)}]$ from layer 0 of the model (12 attention heads and 12 layers), we can already capture a lot of the syntactic information useful to recover the constituency parse tree (get a decently well F1 score). Thus, the F1 score for the whole parse tree may cause people to focus less on the long-range dependencies or long-range structures, and focus more on the short-range de-



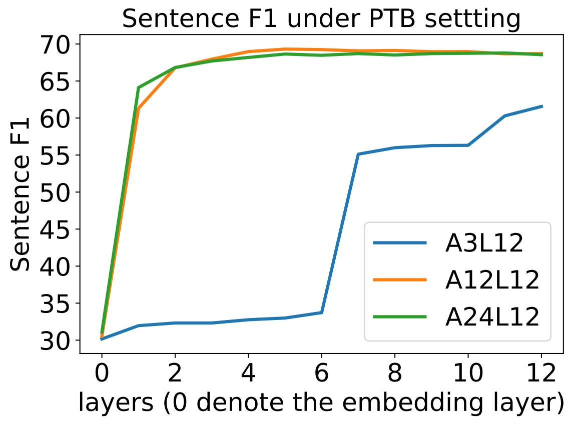
(a) Comparison under PCFG setting. We compare the models with different number of layers.



(b) Comparison under PCFG setting. We compare the models with different number of attention heads.



(c) Comparison under PTB setting. We compare the models with different number of layers.



(d) Comparison under PTB setting. We compare the models with different number of attention heads.

Figure 3: Sentence F1 for linear probes $f(\cdot)$ trained on different layers' embeddings for different pre-trained models. We show the results under PCFG and PTB settings. AiLj denotes the pre-trained model with i attention heads and j layers.

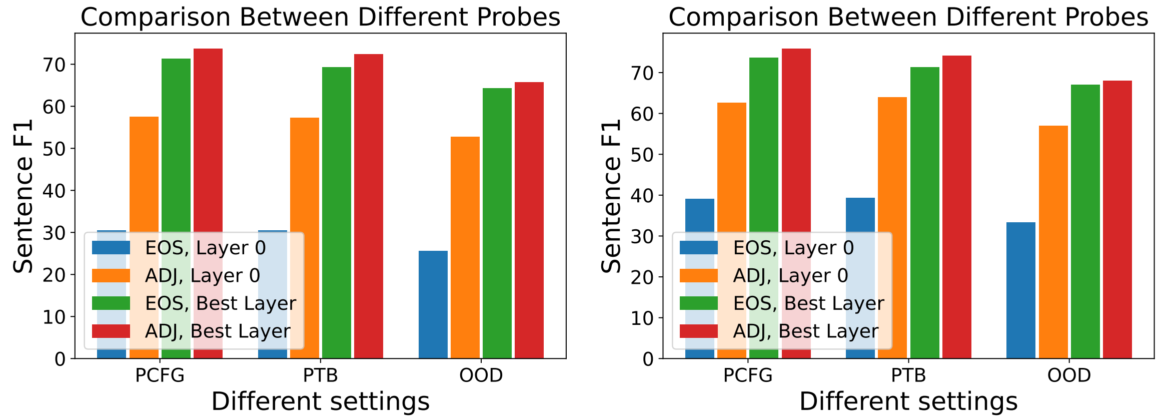
pendencies or structures.

To mitigate this problem, Vilares et al. (2020) computed the F1 score not only for the whole parse tree, but also for each length of spans. Vilares et al. (2020) showed that BERT trained on natural language can get a very good F1 score when the spans are short (for length 2 spans, the probing F1 is over 80%), but when the span becomes longer, the F1 score quickly drops. Even for spans with length 5, the F1 score is less than 70%, and for spans with length 10, the F1 score is less than 60%. Our experiments that probe the marginal probabilities for different lengths of spans (Section 4.3) can also be viewed as an approach to mitigate the problem.

A.3 More results on probing marginal probabilities

In Section 4.3, we conduct probing experiments to demonstrate the predictability of the "normalized marginal probabilities" computed by the Inside-Outside algorithm using transformer representations. Our objective is to establish a strong correlation, measured through the Pearson correlation coefficient. However, we have not provided a comprehensive explanation for our preference for Pearson correlation over alternative metrics such as Spearman correlation. In the following section, we show the experiment results measured by the Spearman correlation, and give an explanation of why we prefer the Pearson correlation over the Spearman correlation.

Measure with Spearman correlation Table 6 summarizes the correlations between the predicted probabilities and the span marginal probabilities computed by the Inside-Outside algorithm on PTB datasets for the 2-linear net probes. It is evident that the Spearman correlation is significantly lower than



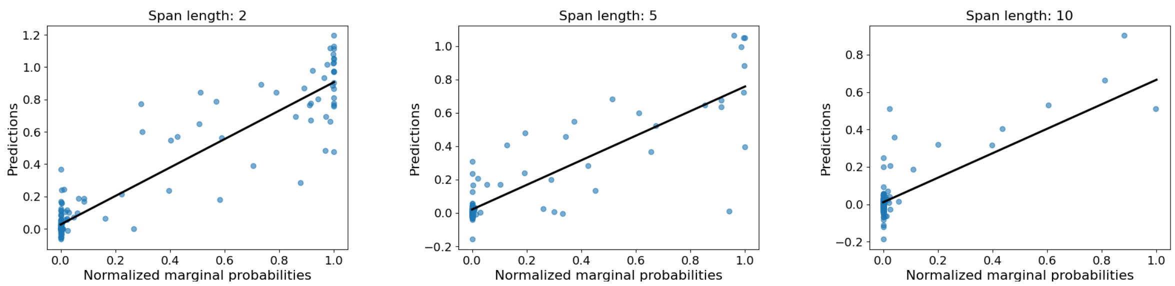
(a) Comparison of different inputs under different settings when the probe $f(\cdot)$ is linear.

(b) Comparison of different inputs under different settings when the probe $f(\cdot)$ is a 2-layer neural network.

Figure 4: Comparison of the probes with different inputs under different settings. We probe the model with 12 attention heads and 12 layers, and report the scores with $f(\cdot)$ taking embeddings from layer 0 or the embeddings from the best layer. EOS denotes the probe that takes $[\boldsymbol{e}_i^{(\ell)}; \boldsymbol{e}_{\text{EOS}}^{(\ell)}]$ as input and predicts the relative depth $\text{tar}(i)$, and ADJ (Adjacent embeddings) denotes the probe that takes $[\boldsymbol{e}_{i-1}^{(\ell)}; \boldsymbol{e}_i^{(\ell)}; \boldsymbol{e}_{i+1}^{(\ell)}]$ as input.

Span Length	A12L12	A12L1	A12L3	A12L6	A3L12	A24L12
$\ell = 2$	.71 / **.93**	.69 / .88	.75 / **.93**	.71 / **.93**	.76 / .86	.75 / **.92**
$\ell = 5$	.59 / **.82**	.54 / .64	.47 / .79	.49 / .79	.54 / .71	.48 / .79
$\ell = 10$	.43 / **.78**	.48 / .68	.59 / .73	.45 / .75	.33 / .62	.39 / .72

Table 6: Probing for the "normalized" marginal probabilities of spans at different lengths on different pre-trained models. We report the Spearman and Pearson correlations (separated by /) between the predicted probabilities and the span marginal probabilities computed by the Inside-Outside algorithm on `PTB` datasets for the 2-linear net probe.



(a) Span length to probe: $\ell = 2$. (b) Span length to probe: $\ell = 5$. (c) Span length to probe: $\ell = 10$.

Figure 5: The predicted probability versus true normalized marginal probability plot for different span lengths ℓ using 2-layer NN probe with the 12-th layer's representations from A12L12 model. In each figure, we sample 200 points (each point corresponds to a span) to plot from the test set. The y-axis denotes the predicted probabilities and the x-axis denotes the true normalized marginal probabilities. The line shows the best linear fit for all the spans in the test set. We can observe that there are lots of points that have very small normalized marginal probabilities, and it is very hard to predict their rank correctly, thus resulting in a low Spearman correlation.

the Pearson correlation, indicating that the probe primarily captures "linear" correlations rather than rank-based relationships.

In order to investigate the underlying cause of this phenomenon, we plot the predicted probabilities against the true normalized marginal probabilities, as shown in Figure 5. Numerous points have extremely small normalized marginal probabilities, particularly when the probe length ℓ is large (e.g., $\ell = 5, 10$). This observation aligns with the intuition that the probability of a randomly selected span existing in the constituency parse tree is low.

However, accurately predicting the exact rank for the points clustered near the origin proves to be extremely challenging, leading to a relatively low Spearman correlation. In contrast, when considering the Pearson correlation, the noise associated with predicting spans having low normalized marginal probabilities is relatively small compared to the overall "variance" of the data points. Furthermore, it is evident that the probe exhibits greater efficacy in capturing the "influential spans" characterized by large normalized marginal probabilities. Achieving relatively accurate predictions for these influential spans accounts for a significant portion of the observed variation, leading to a relatively high Pearson correlation.

A.4 Control tasks

In probing experiments, it is crucial to ensure that the probing performance accurately reflects the presence of the specific information we intend to test. Consequently, it is undesirable for the probe to possess excessive power and be capable of learning all aspects (see Section 2 for further discussions). Chen et al. (2021) utilize the concept of "sensitivity" to assess the extent to which the probe captures the targeted information. The "sensitivity" of a probe is defined as the difference in probing performance between the layer of interest and the 0-th layer (see Section 4.2 and Section 4.3 for further details). Intuitively, a large gap indicates that the probe fails to perform adequately using representations from the 0-th layer but achieves better performance when utilizing representations from a later layer, thus confirming the presence of the targeted information. In situations where there are two probe choices (e.g., a linear classifier or a 2-layer neural network), the option exhibiting greater "sensitivity" should be selected as it captures a relatively higher amount of the targeted information.

Hewitt and Liang (2019) introduced another metric, known as "selectivity", to assess the degree to which the probe captures the targeted information. Broadly speaking, Hewitt and Liang (2019) devised a specific task referred to as the "control task" to evaluate the probe's capability to align with specific types of random labels. Subsequently, "selectivity" is defined as the difference in performance between the probe for the original task, utilizing the layer of interest, and the probe for the control task, also utilizing the layer of interest. Intuitively, a large gap suggests that the probe lacks sufficient expressive power, resulting in the performance boost originating from the representations of the layer being probed, thus confirming the presence of specific information. Similarly, in scenarios involving two probe choices (e.g., a linear classifier or a 2-layer neural network), the option exhibiting greater "selectivity" should be preferred as it captures a relatively higher amount of the targeted information.

It is important to note that a probe with higher "sensitivity" does not necessarily imply larger "selectivity". Nevertheless, as demonstrated in the subsequent parts, the metrics of "sensitivity" and "selectivity" align for both the constituency parsing probes (Section 4.2) and the marginal probability probes (Section 4.3).

Control task Hewitt and Liang (2019) considered control task for sequence labeling problems: Given a sentence $x_{1:T}$, the goal is to label each word $y_{1:T}$. For example, the Part-of-speech tagging problem and the dependency parsing all belong to the sequence labeling category, since for Part-of-speech tagging, y_i is the POS tag of x_i, and for dependency parsing, y_i is the parent of x_i in the parse tree. For a sequence labeling problem, the control task for this sequence labeling problem consists of two key components:

1. Structure: the output $\hat{y}_i$ of a word x_i is a deterministic function of x_i, i.e., $\hat{y}_i = \phi(x_i)$.

2. Randomness: The output $\hat{y}_i$ for each word x_i is sampled independently at random.

Then, the goal of the control task is to fit the labels $\hat{y}_{1:T}$ using the probe with the input $h_{1:T}$ where $h_{1:T}$ denote the hidden representations of the specific layer of the transformer. Please refer to Section 2 of Hewitt and Liang (2019) for more details and examples on control task.

Control task for constituency parsing probe For the constituency parsing probe in Section 4.2, it is easy to design a control task since in Section 4.2 we reduce the constituency parsing problem to a sequence labeling problem that predicts the relative depth of the common ancestors between words. Specifically, we have $y_i = \text{tar}(i) = \text{depth}(i, i+1) - \text{depth}(i-1, i)$ for position i. Then for the control task, for each word w, we uniformly sample $\phi(w) \in \{-1, 0, 1\}$, and then define the labels for the control task as $\hat{y}_{1:T} = [\phi(x_1), \phi(x_2), \ldots, \phi(x_T)]$.

Control task for marginal probability probe For the marginal probability probe in Section 4.3, we need to generalize the original control task from sequence labeling problem to span labeling problem. Given a span $x_{i:j}$, the original goal is to predict the normalized marginal probability $y_{i,j} = \text{tar}(i,j) = s(i,j)/\max_{j_1,j_2} s(j_1,j_2)$ where $s(i,j)$ is the marginal probability for span $i : j$ computed by the Inside-Outside algorithm. Now for each pair of words w_1, w_2, we uniformly sample $\phi(w_1, w_2) \in [0, 1]$. Then for the sequence $x_{1:T}$, we have the label for the control task $\hat{y}_{i,j} = \phi(x_i, x_j)$.

Selectivity* is aligned with *Sensitivity Table 7 and Table 8 provide a summary of the performance of the constituency parsing probe and the marginal probability probes, employing different architectures (linear classifier and a 2-layer neural network with 16 hidden neurons), on the original task, control task, as well as the selectivity.

Based on the results presented in Table 7, it is observed that the probe with a 2-layer neural network achieves slightly higher accuracy in predicting the relative depth of common ancestors, leading to a higher F1 score in constituency parsing. However, its performance on the control task surpasses that of the probe with a linear classifier by a significant margin. This suggests that when using the "selectivity" metric, the linear probe outperforms the 2-layer neural network probe in recovering the constituency parse tree, aligning with the conclusions drawn using the "sensitivity metric" (see Figure 1, where the sensitivity of the linear probe is greater than that of the 2-layer neural network probe).

Based on the information presented in Table 8, it is evident that the probe utilizing a 2-layer neural network demonstrates superior performance in predicting span probabilities for the control task. Nonetheless, compared to the linear probe, the 2-layer neural network probe achieves significantly better results on the original task, resulting in a larger "selectivity". Analyzing Figure 2a, we observe that the 2-layer NN probe exhibits significantly stronger predictive correlation than the linear probe at the 12-th layer of A12L12, while displaying similar performance at the 0-th layer, which contributes to a higher "sensitivity". Consequently, the "selectivity" metric aligns with the "sensitivity" metric for marginal probability probes, indicating that 2-layer NN probes capture a relatively greater amount of syntactic information.

A.5 Analysis of attention patterns

In Section 4.2, we probe the embeddings of the models pre-trained on synthetic data generated from PCFG and show that model training on MLM indeed *captures* syntactic information that can recover the constituency parse tree. Theorem 3.3 builds the connection between MLM and the Inside-Outside algorithm, and the connection is also verified in Section 4.3, which shows that the embeddings also contain the marginal probability information computed by the Inside-Outside algorithm. However, we only build up the correlation between the Inside-Outside algorithm and the attention models, and we still don't know the mechanism inside the language models: the model may be executing the Inside-Outside algorithm (or some approximations of the Inside-Outside algorithm), but it may also use some mechanism far from the Inside-Outside algorithm but happens to contain the marginal probability information. We leave for future work the design of experiments to interpret the content of the contextualized embeddings and thus "reverse-engineer" the learned model. In this section, we take a small step to understand more about the mechanism of language models: we need to *open up the black box* and go further than probing, and this section serves as one step to do so.

General idea The key ingredient that distinguishes current large language models and the fully-connected neural networks is the self-attention module. Thus besides probing for certain information, we can also look at the attention score matrix and discover some patterns. In particular, we are interested in how far an attention head looks at, which we called the "averaged attended distance".

Averaged attended distance For a model and a particular attention head, given a sentence s with length L_s, the head will generate an $L_s \times L_s$ matrix

		L0	L1	L2	L3	L4	L5	L6	L7	L8	L9	L10	L11	L12
Linear	pred. rel. depth	.606	.760	.789	.796	.800	.803	.803	.803	.802	.801	.800	.800	.799
	control task	.758	.677	.645	.626	.620	.610	.608	.617	.599	.595	.612	.606	.608
	selectivity	-.152	.083	.144	.170	.180	.193	.195	.186	.203	**.206**	.188	.194	.191
NN	pred. rel. depth	.616	.771	.804	.810	.814	.807	.815	.802	.795	.810	.806	.803	.776
	control task	.861	.793	.758	.667	.728	.653	.653	.668	.678	.693	.680	.697	.687
	selectivity	-.245	-.022	.046	.143	.086	.154	**.162**	.134	.117	.117	.126	.106	.089

Table 7: Computing the selectivity of constituency parsing probes with linear and 2-layer NN architectures (see Section 4.2 and Appendix A.4). The "pred. rel. depth" rows denote the probing results for the relative depth of common ancestors in the constituency parse tree using different layers' representations of A12L12. We report the predicting accuracy under the `PTB` setting where the probe is trained and tested on `PTB` dataset. The "control task" rows denote the predicting accuracy for the control task on `PTB` dataset using different layers' representations of A12L12. The selectivity is the difference between the original task performance and the control task performance. We can observe that for all layers representations, the probe with a linear classifier has a larger selectivity.

	Probe span length	2	3	4	5	10
Linear	pred. marginal prob.	.88	.79	.69	.62	.51
	control task	.62	.55	.53	.60	.58
	selectivity	.26	.24	.16	.02	-.07
NN	pred. marginal prob.	.93	.90	.86	.79	.77
	control task	.66	.66	.69	.66	.68
	selectivity	.27	.24	.17	.13	.09

Table 8: Computing the selectivity of marginal probability probes with linear and 2-layer NN architectures (see Section 4.3 and Appendix A.4). The "pred. marginal prob." rows denote the probing results for the "normalized" marginal probabilities of spans at different lengths using the 12-th layer of A12L12. We report the Pearson correlation between the predicted probabilities and the span marginal probabilities computed by the Inside-Outside algorithm on `PTB` dataset. The "control task" rows denote the Pearson correlation between the predicted probabilities and the probabilities generated from the control task on `PTB` dataset using the 12-th layer of A12L12. The selectivity is the difference between the original task performance and the control task performance. We can observe that for spans with all lengths tested, the probe with 2-layer NN has a larger selectivity, especially when the probe length is large.

$\boldsymbol{A}$ containing the pair-wise attention score, where each row of $\boldsymbol{A}$ sums to 1. Then we compute the following quantity "Averaged attended distance"

$$\mathrm{AD}_s = \frac{1}{L_s} \sum_{1 \le i,j \le L_s} |i-j| \cdot \boldsymbol{A}_{i,j},$$

which can be intuitively interpreted as "the average distance this attention head is looking at". We then take the average of the quantity for all sentences. We compute "Averaged attended distance" for three models on the synthetic PCFG dataset and PTB dataset. The models all have 12 attention heads in each layer but have 12, 6, 3 layers respectively.

Experiment results Figure 6 shows the results of the "Averaged attented distance" for each attention head in different models. Figures 6a, 6c and 6e show the results on the synthetic PCFG dataset, and Figures 6b, 6d and 6f show the results on the PTB dataset. We sort the attention heads in each layer according to the "Averaged attended distance".

From Figures 6a, 6c and 6e, we can find that for all models, there are several attention heads in the first layer that look at very close tokens ("Averaged attended distance" less than 3). Then as the layer increases, the "Averaged attended distance" also increases in general, meaning that the attention heads are looking at further tokens. Then at some layer, there are some attention heads looking at very far tokens ("Averaged attended distance" larger than 12).[3] This finding also gives some implication that the model is doing something that correlates with our construction: it looks longer spans as the layer increases. However, different from our construction that the attention head only looks at a fixed length span, models trained using MLM look at different lengths of spans at each layer, which cannot be explained by our current construction, and suggests a further understanding of the mechanism of large language models.

Besides, we can find that the patterns are nearly the same for the synthetic PCFG dataset and PTB dataset, and thus the previous finding can also be transferred to the PTB dataset.

B Missing Proofs in Section 3

In this section, we show the detailed proof for Theorem 3.1, Theorem 3.2, and Theorem 3.3.

[3]Note that the average length of the sentences in the synthetic PCFG dataset is around 24, if the attention head gives 0.5 attention score to the first and the last token for every token, the "Averaged attended distance" will be 12.

B.1 Proof of Theorem 3.1

Proof. The first $L-1$ layers simulate the recursive formulation of the Inside probabilities from eq. 2, and the last $L-1$ layers simulate the recursive formulation of the outside probabilities from eq. 3. The model uses embeddings of size $4|\mathcal{N}|L + L$, where the last L coordinates serve as one-hot positional embeddings and are kept unchanged throughout the model.

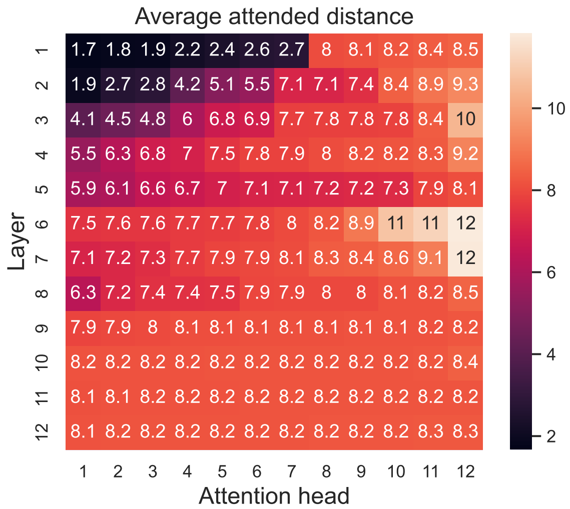
Notations: For typographical simplicity, we will divide our embeddings into 5 sub-parts. We will use the first $2|\mathcal{N}|L$ coordinates to store the inside probabilities, the second $2|\mathcal{N}|L$ coordinates to store the outside probabilities, and the final L coordinates to store the one-hot positional encodings. For every position i and span length $\ell+1$, we store the inside probabilities $\{\alpha(A, i, i+\ell)\}_{A\in\mathcal{N}}$ after computation in its embedding at coordinates $[|\mathcal{N}|\ell, |\mathcal{N}|(\ell+1))$. Similarly we store $\{\alpha(A, i-\ell, i)\}_{A\in\mathcal{N}}$ at $[|\mathcal{N}|(L+\ell), |\mathcal{N}|(L+\ell+1))$, $\{\beta(A, i, i+\ell)\}_{A\in\mathcal{N}}$ at $[|\mathcal{N}|(2L+\ell), |\mathcal{N}|(2L+\ell+1))$, and $\{\beta(A, i-\ell, i)\}_{A\in\mathcal{N}}$ at $[|\mathcal{N}|(3L+\ell), |\mathcal{N}|(3L+\ell+1))$ respectively. For simplicity of presentation, we won't handle cases where $i+\ell$ or $i-\ell$ is outside the range of 1 to L - those coordinates will be fixed to 0.

Token Embeddings: The initial embeddings for each token w will contain $\Pr[A \to w]$ for all $A \in \mathcal{P}$. This is to initiate the inside probabilities of all spans of length 1. Furthermore, the tokens will have a one-hot encoding of their positions in the input in the last L coordinates.

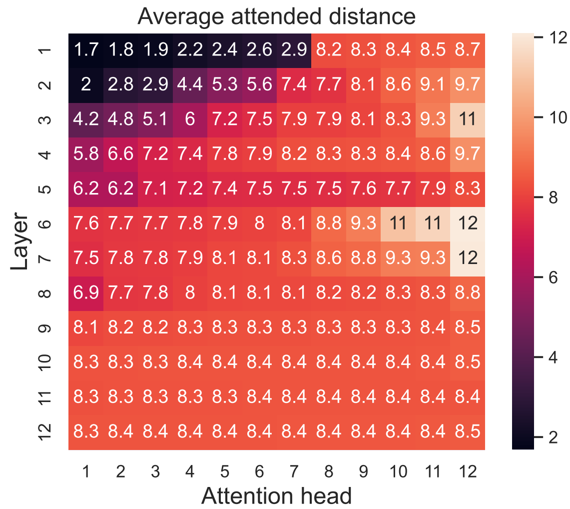
Inside probabilities: The contextual embeddings at position i after the computations of any layer $\ell < L$ contains the inside probabilities of all spans of length at most $\ell+1$ starting and ending at position i, i.e. $\alpha(A, i, i+k)$ and $\alpha(A, i-k, i)$ for all $A \in \mathcal{N}$ and $k \le \ell$. The rest of the coordinates, except the position coordinates, contain 0.

Layer $1 \le \ell < L$: At each position i, this layer computes the inside probabilities of spans of length $\ell+1$ starting and ending at i, using the recursive formulation from eq. 2.

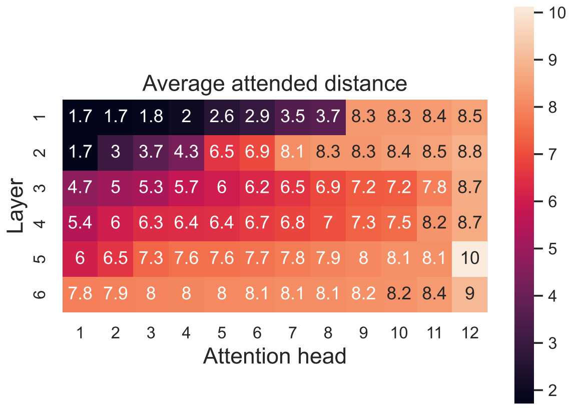
For every non-terminal $A \in \mathcal{N}$, we will use a unique attention head to compute $\alpha(A, i, i+\ell)$ at each token i. Specifically, the attention head representing non-terminal $A \in \mathcal{N}$ will represent the following operation at each position i:



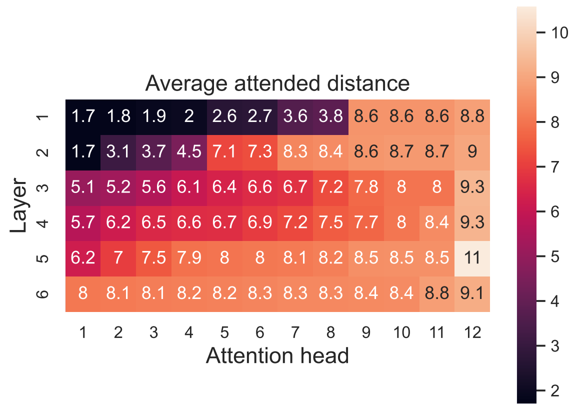
(a) 12 attention heads and 12 layers, `PCFG` dataset.



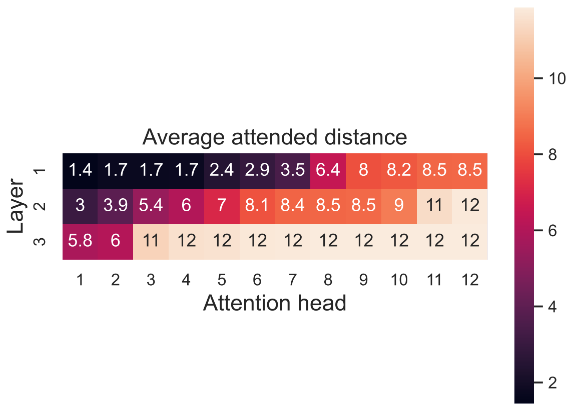
(b) 12 attention heads and 12 layers, `PTB` dataset.



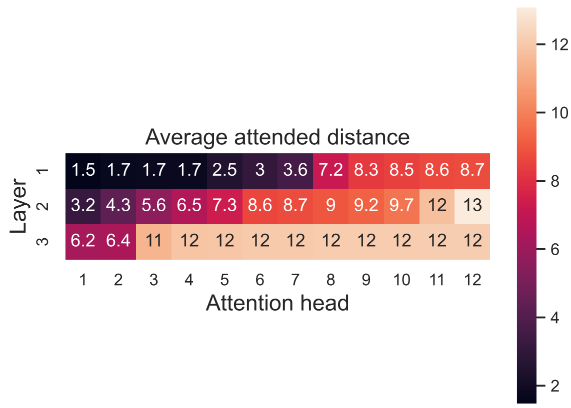
(c) 12 attention heads and 6 layers, `PCFG` dataset.



(d) 12 attention heads and 6 layers, `PTB` dataset.



(e) 12 attention heads and 3 layers, `PCFG` dataset.



(f) 12 attention heads and 3 layers, `PTB` dataset.

Figure 6: "Averaged attented distance" of each attention heads for different models on `PCFG` and `PTB` datasets. Figures 6a, 6c and 6e show the results on the synthetic `PCFG` dataset, and Figures 6b, 6d and 6f show the results on the `PTB` dataset.

$$
\begin{aligned}
&\alpha(A, i, j) \\
&= \sum_{B,C\in\mathcal{N}} \sum_{k=i}^{j-1} \Pr[A \to BC] \cdot \alpha(B, i, k) \cdot \alpha(C, k+1, j) \\
&= \sum_{B,C\in\mathcal{N}} \sum_{\substack{\ell_1,\ell_2\geq 0\\ \ell_1+\ell_2=\ell-1}} \Pr[A \to BC] \\
&\quad \cdot \alpha(B, i, i+\ell_1) \cdot \alpha(C, j-\ell_2, j), \qquad (7)
\end{aligned}
$$

where $j = i + \ell$. In the final step, we modified the formulation to represent the interaction of spans of different lengths starting at i and ending at j. We represent this computation as the attention score $a_{i,j}$ using a key matrix $\boldsymbol{K}_A^{(\ell)}$ and query matrix $\boldsymbol{Q}_A^{(\ell)}$.

Computing Eq. 7 We set the Key matrix $\boldsymbol{K}_A^{(\ell)}$ as $\boldsymbol{I}$. The Query matrix $\boldsymbol{Q}_A^{(\ell)}$ is set such that if we define $\boldsymbol{P}_A \in \mathbb{R}^{|\mathcal{N}|\times|\mathcal{N}|}$ that contains $\{\Pr[A \to BC]\}_{B,C\in\mathcal{N}}$, $\boldsymbol{P}_A$ appears at positions $(|\mathcal{N}|(L+\ell_2), |\mathcal{N}|\ell_1)$ for all $\ell_1, \ell_2 \geq 0$ with $\ell_1 + \ell_2 = \ell - 1$. Finally, $\boldsymbol{Q}_A^{(\ell)}$ contains $\boldsymbol{Q}_p \in \mathbb{R}^{L\times L}$ at position $(4|\mathcal{N}|L, 4|\mathcal{N}|L)$, such that $\boldsymbol{Q}_p[i, i+\ell] = 0$ for $0 \leq i < L$, with the rest set to $-\zeta$ for some large constant ζ. The rest of the blocks are set as 0. We give an intuition behind the structure of $\boldsymbol{Q}_A^{(\ell)}$ below.

Intuition behind $\boldsymbol{Q}_A^{(\ell)}$: For any position i and range $\ell_1 \leq \ell$, $\boldsymbol{e}_i^{(\ell-1)}$ contains the inside probabilities $\{\alpha(C, i-\ell_1, i)\}_{C\in\mathcal{N}}$ in the coordinates $[|\mathcal{N}|(L+\ell_1), |\mathcal{N}|(L+\ell_1+1))$, while it contains the inside probabilities $\{\alpha(B, i, i+\ell_1)\}_{B\in\mathcal{N}}$ in the coordinates $[|\mathcal{N}|\ell_1, |\mathcal{N}|(\ell_1+1))$. Hence, if we set the block at position $(|\mathcal{N}|(L+\ell_2), |\mathcal{N}|\ell_1)$ in $\boldsymbol{Q}_A^{(\ell)}$ to $\boldsymbol{P}_A$ for some $0 \leq \ell_1, \ell_2 \leq \ell$, with the rest set to 0, we can get for any two positions i, j,

$$
\begin{aligned}
&(\boldsymbol{K}_A^{(\ell)}\boldsymbol{e}_j^{(\ell-1)})^\top \boldsymbol{Q}_A^{(\ell)} \boldsymbol{e}_i^{(\ell-1)} \\
=&\sum_{B,C\in\mathcal{N}} \Pr[A \to BC] \cdot \alpha(B, i, i+\ell_1) \cdot \alpha(C, j-\ell_2, j).
\end{aligned}
$$

Because we want to involve the sum over all ℓ_1, ℓ_2 pairs with $\ell_1 + \ell_2 = \ell - 1$, we will set blocks at positions $\{(|\mathcal{N}|(L+\ell_2), |\mathcal{N}|\ell_1)\}_{\ell_1,\ell_2:\ell_1+\ell_2=\ell-1}$ to $\boldsymbol{P}_A$, while setting the rest to 0. This gives us

$$
\begin{aligned}
&(\boldsymbol{K}_A^{(\ell)}\boldsymbol{e}_j^{(\ell-1)})^\top \boldsymbol{Q}_A^{(\ell)} \boldsymbol{e}_i^{(\ell-1)} \\
=&\sum_{B,C\in\mathcal{N}} \sum_{\substack{\ell_1,\ell_2\geq 0\\ \ell_1+\ell_2=\ell-1}} \Pr[A \to BC] \cdot \alpha(B, i, i+\ell_1) \\
&\cdot \alpha(C, j-\ell_2, j).
\end{aligned}
$$

However, we want $(\boldsymbol{K}_A^{(\ell)}\boldsymbol{e}_j^{(\ell-1)})^\top \boldsymbol{Q}_A^{(\ell)} \boldsymbol{e}_i^{(\ell-1)}$ to compute $\alpha(A, i, j)$ iff $j = i + \ell$ and 0 otherwise, so we will use the final block in $\boldsymbol{Q}_A^{(\ell)}$ that focuses on the one-hot position encodings of i and j to differentiate the different location pairs. Specifically, the final block $\boldsymbol{Q}_p$ will return 0 if $j = i + \ell$, while it returns $-\zeta$ for some large constant ζ if $j \neq i + \ell$. This gives us

$$
\begin{aligned}
&(\boldsymbol{K}_A^{(\ell)}\boldsymbol{e}_j^{(\ell-1)})^\top \boldsymbol{Q}_A^{(\ell)} \boldsymbol{e}_i^{(\ell-1)} \\
=&\zeta(\mathbb{I}[j-i=\ell]-1) + \sum_{B,C\in\mathcal{N}} \sum_{\substack{\ell_1,\ell_2\geq 0\\ \ell_1+\ell_2=\ell-1}} \Pr[A \to BC] \\
&\cdot \alpha(B, i, i+\ell_1) \cdot \alpha(C, j-\ell_2, j). \qquad (8)
\end{aligned}
$$

With the inclusion of the term $\zeta(\mathbb{I}[j-i=\ell]-1)$, we make $(\boldsymbol{K}_A^{(\ell)}\boldsymbol{e}_j^{(\ell-1)})^\top \boldsymbol{Q}_A^{(\ell)} \boldsymbol{e}_i^{(\ell-1)}$ positive if $j - i = \ell$, and negative if $j - i \neq \ell$. Applying a ReLU activation on top will zero out the unnecessary terms, leaving us with $\alpha(A, i, i+\ell)$ at each location i.

Similarly, we use another $|\mathcal{N}|$ attention heads to compute $\alpha(A, i-\ell, i)$. In the end, we use the residual connections to copy the previously computed inside probabilities $\alpha(A, i-\ell', i)$ and $\alpha(A, i, i+\ell')$ for $\ell' < \ell$.

Outside probabilities: In addition to all the inside probabilities, the contextual embeddings at position i after the computations of any layer $(2L-1)-\ell$ $(\geq L)$ contain the outside probabilities of all spans of length at least $\ell+1$ starting and ending at position i, i.e. $\beta(A, i, i+k)$ and $\beta(A, i-k, i)$ for all $A \in \mathcal{N}$ and $k \geq \ell$. The rest of the coordinates, except the position coordinates, contain 0.

Layer L In this layer, we initialize the outside probabilities $\beta(\text{ROOT}, 1, L) = 1$ and $\beta(A, 1, L) = 0$ for $A \neq$ ROOT. Furthermore, we move the inside probabilities $\alpha(A, i+1, i+k)$ from position $i+1$ to position i, and $\alpha(A, i-k, i-1)$ from position $i-1$ to position i using 2 attention heads.

Layer $L+1 \leq \tilde{\ell} := (2L-1)-\ell \leq 2L-1$: At each position i, this layer computes the outside probabilities of spans of length $\ell+1$ starting and ending at i, using the recursive formulation from eq. 3. The recursive formulation for $\beta(A, i, i+\ell)$ for a non-terminal $A \in \mathcal{N}$ has two terms, given by

$$
\begin{aligned}
\beta(A, i, j) =& \beta_1(A, i, j) + \beta_2(A, i, j), \text{ with} \\
\beta_1(A, i, j) =& \sum_{C,B\in\mathcal{N}} \sum_{k=1}^{i-1} \Pr[B \to CA] \\
&\cdot \alpha(C, k, i-1)\beta(B, k, j), \text{ and} \qquad (9) \\
\beta_2(A, i, j) =& \sum_{B,C\in\mathcal{N}} \sum_{k=j+1}^{L} \Pr[B \to AC] \\
&\cdot \alpha(C, j+1, k)\beta(B, i, k), \qquad (10)
\end{aligned}
$$

where $j = i + \ell$. For each non-terminal $A \in \mathcal{N}$, we will use two unique heads to compute $\beta(A, i, i+\ell)$, each representing one of the two terms in the above formulation. We outline the construction for β_1; the construction for β_2 follows similarly.

Computing Eq. 9 We build the attention head in the same way we built the attention head to represent the inside probabilities in eq. 8. Similar to 8, we modify the formulation of β_1 to highlight the interaction of spans of different lengths.

$$\beta_1(A,i,j) = \sum_{B,C\in\mathcal{N}} \sum_{\substack{\ell_1,\ell_2\geq 0\\ \ell_2-\ell_1=\ell}} \Pr[B\to CA] \cdot \alpha(C,i-\ell_1,i-1)\beta(B,j-\ell_2,j), \quad (11)$$

where $j=i+\ell$. We represent this computation as the attention score $a_{i,i+\ell}$ using a key matrix $\boldsymbol{K}_{A,1}^{(\tilde{\ell})}$ and query matrix $\boldsymbol{Q}_{A,1}^{(\tilde{\ell})}$. First, we set the Key matrix $\boldsymbol{K}_{A,1}^{(\tilde{\ell})}$ as $\boldsymbol{I}$. If we define $\boldsymbol{P}_{A,r}\in\mathbb{R}^{|\mathcal{N}|\times|\mathcal{N}|}$ as a matrix that contains $\{\Pr[B\to CA]\}_{B,C\in\mathcal{N}}$, which is the set of all rules where A appears as the right child, $\boldsymbol{Q}_{A,1}^{(\tilde{\ell})}$ is set such that $\boldsymbol{P}_{A,r}$ appears at positions $[|\mathcal{N}|(3L+\ell_2), |\mathcal{N}|(L+\ell_1))$ for all $0\leq \ell_1,\ell_2\leq L$ that satisfy $\ell_2-\ell_1=\ell$. Finally, $\boldsymbol{Q}_{A,1}^{(\tilde{\ell})}$ contains $\boldsymbol{Q}_p\in\mathbb{R}^{L\times L}$ at position $(4|\mathcal{N}|L, 4|\mathcal{N}|L)$, such that $\boldsymbol{Q}_p[i,i+\ell]=0$ for $0\leq i<L$, with the rest set to $-\zeta$ for some large constant ζ. The rest of the blocks are set as 0. We give an intuition behind the structure of $\boldsymbol{Q}_{A,1}^{(\tilde{\ell})}$ below.

Intuition for $\boldsymbol{Q}_{A,1}^{(\tilde{\ell})}$: For position i and any ranges $1\leq \ell_1<L$, $\ell+1\leq\ell_2\leq L$, $\boldsymbol{e}_i^{(\tilde{\ell}-1)}$ contains the inside probabilities $\{\alpha(C,i-\ell_1,i-1)\}_{C\in\mathcal{N}}$ in the coordinates $[|\mathcal{N}|(L+\ell_1), |\mathcal{N}|(L+\ell_1+1))$, while it contains the outside probabilities $\{\beta(B,i-\ell_2,i)\}_{B\in\mathcal{N}}$ in the coordinates $[|\mathcal{N}|(3L+\ell_2), |\mathcal{N}|(3L+\ell_2+1))$. Hence, if we set the block at position $(|\mathcal{N}|(3L+\ell_2), |\mathcal{N}|(L+\ell_1))$ to $\boldsymbol{P}_A$ for some $0\leq\ell_1\leq L, \ell+1\leq\ell_2\leq L$, with the rest set to 0, we can get for any two positions i,j,

$$\begin{aligned}&(\boldsymbol{K}_A^{(\tilde{\ell})}\boldsymbol{e}_j^{(\tilde{\ell}-1)})^\top \boldsymbol{Q}_A^{(\tilde{\ell})}\boldsymbol{e}_i^{(\tilde{\ell}-1)}\\ =&\sum_{B,C\in\mathcal{N}}\Pr[B\to CA]\cdot\alpha(C,i-\ell_1,i-1)\cdot\beta(B,j-\ell_2,j).\end{aligned}$$

Because we want to include the sum over ℓ_1,ℓ_2 pairs with $\ell_2-\ell_1=\ell$, we will only set blocks at positions $[|\mathcal{N}|(3L+\ell_2), |\mathcal{N}|(L+\ell_1))$ for all $0\leq\ell_1,\ell_2\leq L$ that satisfy $\ell_2-\ell_1=\ell$ to $\boldsymbol{P}_{A,r}$, while setting the rest to 0. This gives us

$$\begin{aligned}&(\boldsymbol{K}_A^{(\tilde{\ell})}\boldsymbol{e}_j^{(\tilde{\ell}-1)})^\top \boldsymbol{Q}_A^{(\tilde{\ell})}\boldsymbol{e}_i^{(\tilde{\ell}-1)}\\ =&\sum_{B,C\in\mathcal{N}}\sum_{\substack{\ell_1,\ell_2\geq 0\\ \ell_2-\ell_1=\ell}}\Pr[B\to CA]\\ &\cdot\alpha(C,i-\ell_1,i-1)\cdot\beta(B,j-\ell_2,j).\end{aligned}$$

Because we want $(\boldsymbol{K}_A^{(\tilde{\ell})}\boldsymbol{e}_j^{(\tilde{\ell}-1)})^\top \boldsymbol{Q}_A^{(\tilde{\ell})}\boldsymbol{e}_i^{(\tilde{\ell}-1)}$ to compute $\beta_1(A,i,j)$ with $j=i+\ell$ and 0 otherwise, we will use the final block in $\boldsymbol{Q}_A^{(\ell)}$ that focuses on the one-hot position encodings of i and j to differentiate the different location pairs. Specifically, the final block $\boldsymbol{Q}_p$ will return 0 if $j=i+\ell$, while it returns $-\zeta$ for some large constant ζ, if $j\neq i+\ell$. This gives us

$$\begin{aligned}&(\boldsymbol{K}_A^{(\tilde{\ell})}\boldsymbol{e}_j^{(\tilde{\ell}-1)})^\top \boldsymbol{Q}_A^{(\tilde{\ell})}\boldsymbol{e}_i^{(\tilde{\ell}-1)}\\ =&\zeta(\mathbb{I}[j-i=\ell]-1)+\sum_{B,C\in\mathcal{N}}\sum_{\substack{\ell_1,\ell_2\geq 0\\ \ell_2-\ell_1=\ell}}\Pr[B\to CA]\\ &\cdot\alpha(C,i-\ell_1,i-1)\cdot\beta(B,j-\ell_2,j)\end{aligned}$$

With the inclusion of the term $\zeta(\mathbb{I}[j-i=\ell]-1)$, we make $(\boldsymbol{K}_A^{(\tilde{\ell})}\boldsymbol{e}_j^{(\tilde{\ell}-1)})^\top \boldsymbol{Q}_A^{(\tilde{\ell})}\boldsymbol{e}_i^{(\tilde{\ell}-1)}$ positive if $j-i=\ell$, and negative if $j-i\neq\ell$. Applying a ReLU activation on top will zero out the unnecessary terms, leaving us with $\beta_1(A,i,i+\ell)$ at each location i.

Besides, we also need $2|\mathcal{N}|$ additional heads for the outside probabilities $\beta(A,i-\ell,i)$. In the end, we use the residual connections to copy the previously computed inside probabilities $\beta(A,i-\ell',i)$ and $\alpha(A,i,i+\ell')$ for $\ell'>\ell$. □

B.2 Proof of Theorem 3.2

Similar to the proof of Theorem 3.1, the first $L-1$ layers simulate the recursive formulation of the Inside probabilities from eq. 2, and the last $L-1$ layers simulate the recursive formulation of the outside probabilities from eq. 3. The model uses embeddings of size $2|\mathcal{N}|L$ and uses $4L+2$ relative position embeddings.

Notations: For typographical simplicity, we will divide our embeddings into 2 sub-parts. We will use the first $|\mathcal{N}|L$ coordinates to store the inside probabilities, and the second $|\mathcal{N}|L$ coordinates to store the outside probabilities. For every position i and span length $\ell+1$, we store the inside probabilities $\{\alpha(A,i-\ell,i)\}_{A\in\mathcal{N}}$ after computation in its embedding at coordinates $[|\mathcal{N}|\ell, |\mathcal{N}|(\ell+1))$, where the coordinates for embeddings start from 0. Similarly we store $\{\beta(A,i,i+\ell)\}_{A\in\mathcal{N}}$ at $[|\mathcal{N}|(L+\ell), |\mathcal{N}|(L+\ell+1))$. For simplicity of presentation, we won't handle cases where $i+\ell$ or $i-\ell$ is outside the range of 1 to L - those coordinates will be fixed to 0.

Token Embeddings: The initial embeddings for each token w will contain $\Pr[A \to w]$ for all $A \in \mathcal{P}$. This is to initiate the inside probabilities of all spans of length 1.

Relative position embeddings: We introduce $2L + 1$ relative position vectors $\{p_t \in \mathbb{R}^{2|\mathcal{N}|L}\}_{-L \le t \le L}$, that modify the key vectors depending on the relative position of the query and key tokens. Furthermore, we introduce $(2L-1)L$ relative position-dependent biases $\{b_{t,\ell} \in \mathbb{R}\}_{-L \le t \le L, 1 \le \ell \le 2L-1}$. We introduce the structures of the biases in the contexts of their intended uses.

Structure of $\{p_t\}_{-L \le t \le L}$: For $t < 0$, we define p_t such that all coordinates in $[|\mathcal{N}|(-t-1), |\mathcal{N}|(-t))$ are set to 1, with the rest set to 0. For $t > 0$, we define p_t such that all coordinates in $[|\mathcal{N}|(L+t-1), |\mathcal{N}|(L+t))$ are set to 1, with the rest set to 0. p_0 is set as all 0s.

Attention formulation: At any layer $1 \le \ell \le 2L-1$ except L, we define the attention score $a^h_{i,j}$ between $\boldsymbol{e}_i^{(\ell-1)}$ and $\boldsymbol{e}_j^{(\ell-1)}$ for any head h with Key and Query matrices $\boldsymbol{K}_h^{(\ell)}$ and $\boldsymbol{Q}_h^{(\ell)}$ as

$$a^h_{i,j} = \mathrm{ReLU}(\boldsymbol{K}_h^{(\ell)} \boldsymbol{e}_j^{(\ell-1)} + p_{j-i} - b_{j-i,\ell})^\top \boldsymbol{Q}_h^{(\ell)} \boldsymbol{e}_i^{(\ell-1)}. \tag{12}$$

For layer L, we do not use the relative position embeddings, i.e. we define the attention score $a^h_{i,j}$ between $\boldsymbol{e}_i^{(L-1)}$ and $\boldsymbol{e}_j^{(L-1)}$ for any head h with Key and Query matrices $\boldsymbol{K}_h^{(L)}$ and $\boldsymbol{Q}_h^{(L)}$ as

$$a^h_{i,j} = \mathrm{ReLU}(\boldsymbol{K}_h^{(L-1)} \boldsymbol{e}_j^{(L-1)} - b_{j-i,L})^\top \boldsymbol{Q}_h^{(\ell)} \boldsymbol{e}_i^{(L-1)}. \tag{13}$$

Inside probabilities: The contextual embeddings at position i after the computations of any layer $\ell < L$ contains the inside probabilities of all spans of length at most $\ell + 1$ ending at position i, i.e. $\alpha(A, i-k, i)$ for all $A \in \mathcal{N}$ and $k \le \ell$. The rest of the coordinates contain 0.

Structure of $\{b_{t,\ell}\}_{-L \le t \le L, 1 \le \ell \le L-1}$: For any $1 \le \ell \le L-1$, for all $t \ge 0$ and $t < -\ell$, we set $b_{t,\ell}$ as ζ for some large constant ζ. All other biases are set as 1.

Layer $1 \le \ell < L$: At each position i, this layer computes the inside probabilities of spans of length $\ell + 1$ ending at i, using the recursive formulation from eq. 2.

For every non-terminal $A \in \mathcal{N}$, we will use a unique attention head to compute $\alpha(A, i-\ell, i)$ at each token i. Specifically, the attention head representing non-terminal $A \in \mathcal{N}$ will represent the following operation at each position i:

$$\begin{aligned}
&\alpha(A, i-\ell, i) \\
&= \sum_{B,C \in \mathcal{N}} \sum_{j=i-\ell}^{i-1} \Pr[A \to BC] \alpha(B, i-\ell, j) \alpha(C, j+1, i) \\
&= \sum_{j=i-\ell}^{i-1} \sum_{B,C \in \mathcal{N}} \Pr[A \to BC] \alpha(B, i-\ell, j) \alpha(C, j+1, i).
\end{aligned} \tag{14}$$

In the final step, we swapped the order of the summations to observe that the desired computation can be represented as a sum over individual computations at locations $j < i$. That is, we represent $\sum_{B,C \in \mathcal{N}} \Pr[A \to BC] \cdot \alpha(B, i-\ell, j) \cdot \alpha(C, j+1, i)$ as the attention score $a_{i,j}$ for all $i - \ell \le j \le i$, while $\alpha(A, i-\ell, i)$ will be represented as $\sum_{i-\ell \le j < i-1} a_{i,j}$.

Structure of $\boldsymbol{Q}_A^{(\ell)}$ and $\boldsymbol{K}_A^{(\ell)}$ to compute Eq. 14:

1. $\boldsymbol{K}_A^{(\ell)}$ is a rotation matrix such that in $\boldsymbol{K}_A^{(\ell)} \boldsymbol{e}_i^{(\ell)}$, for all $\ell_1 \le \ell$, the inside probabilities $\{\alpha(B, i-\ell_1, i)\}_{B \in \mathcal{N}}$ appears in the coordinates $[|\mathcal{N}|(\ell - \ell_1), |\mathcal{N}|(\ell - \ell_1 + 1))$. Note that $\boldsymbol{K}_A^{(\ell)}$ are the same for different A, and only depend on ℓ.

2. The Query matrix $\boldsymbol{Q}_A^{(\ell)}$ is a block diagonal matrix, such that if we define $\boldsymbol{P}_A \in \mathbb{R}^{|\mathcal{N}| \times |\mathcal{N}|}$ that contains $\{\Pr[A \to BC]\}_{B,C \in \mathcal{N}}$, $\boldsymbol{P}_A$ appears in the first ℓ blocks along the diagonal, i.e. it occurs at all positions starting at $(|\mathcal{N}|\ell_1, |\mathcal{N}|\ell_1)$ for all $\ell_1 < \ell$. The rest of the blocks are set as 0s.

Intuition behind $\boldsymbol{Q}_A^{(\ell)}$, $\boldsymbol{K}_A^{(\ell)}$, the relative position embeddings and the biases: For any position i and range $\ell_1 < \ell$, $\boldsymbol{e}_i^{(\ell-1)}$ contains the inside probabilities $\{\alpha(C, i-\ell_1, i)\}_{C \in \mathcal{N}}$ in the coordinates $[|\mathcal{N}|\ell_1, |\mathcal{N}|(\ell_1 + 1))$. With the application of $\boldsymbol{K}_A^{(\ell)}$, $\boldsymbol{K}_A^{(\ell)} \boldsymbol{e}_i^{(\ell-1)}$ contains the inside probabilities $\{\alpha(C, i-\ell_1, i)\}_{C \in \mathcal{N}}$ in the coordinates $[|\mathcal{N}|(\ell - 1 - \ell_1), |\mathcal{N}|(\ell - \ell_1))$. Hence, if we set the block at position $(|\mathcal{N}|\ell_1, |\mathcal{N}|\ell_1)$ in $\boldsymbol{Q}_A^{(\ell)}$ to $\boldsymbol{P}_A$ for some $0 \le \ell_1 < \ell$, with the rest set to 0, we can get for any two positions i, j,

$$\begin{aligned}
&(\boldsymbol{K}_A^{(\ell)} \boldsymbol{e}_j^{(\ell-1)})^\top \boldsymbol{Q}_A^{(\ell)} \boldsymbol{e}_i^{(\ell-1)} \\
&= \sum_{B,C \in \mathcal{N}} \Pr[A \to BC] \cdot \alpha(B, i-\ell_1, i)
\end{aligned}$$

$$\cdot \alpha(C, j-(\ell-1-\ell_1), j).$$

Setting the first ℓ diagonal blocks in $\boldsymbol{Q}_A^{(\ell)}$ to $\boldsymbol{P}_A$ can get for any two positions i, j,

$$\begin{aligned}&(\boldsymbol{K}_A^{(\ell)} \boldsymbol{e}_j^{(\ell-1)})^\top \boldsymbol{Q}_A^{(\ell)} \boldsymbol{e}_i^{(\ell-1)} \\ =& \sum_{\ell_1 \le \ell-1} \sum_{B,C \in \mathcal{N}} \Pr[A \to BC] \cdot \alpha(B, i-\ell_1, i) \\ &\cdot \alpha(C, j-(\ell-\ell_1-1), j).\end{aligned}$$

However, for $\alpha(A, i-\ell, i)$, the attention score above should only contribute with $\ell_1 = i-j-1$. Moreover, we also want the above sum to be 0 if $j \ge i$ or $j \le i-\ell-1$. Hence, we will use the relative position vector p_{j-i}, bias $b_{j-i,\ell}$ and the ReLU activation to satisfy the following conditions:

1. $i-\ell \le j \le i-1$.
2. The portion containing $\{\alpha(C, j-(\ell-\ell_1-1), j)\}_{C\in\mathcal{N}}$ in $\boldsymbol{K}_A^{(\ell)} \boldsymbol{e}_j^{(\ell-1)}$ is activated only if $\ell_1 = i-j-1$.

For any positions i, j and $\ell_1 < \ell$, $\boldsymbol{K}_A^{(\ell)} \boldsymbol{e}_j^{(\ell-1)} + p_{j-i} - b_{j-i,\ell}$ will contain $\{\alpha(C, j-(\ell-\ell_1-1), j) + \mathbb{I}[\ell_1 = i-j-1] - 1 - \zeta\mathbb{I}[j < i-\ell \text{ or } j > i-1]\}_{C\in\mathcal{N}}$ in coordinates $[|\mathcal{N}|\ell_1, |\mathcal{N}|(\ell_1+1))$, which will give us

$$\begin{aligned}&\mathrm{ReLU}(\boldsymbol{K}_A^{(\ell)} \boldsymbol{e}_j^{(\ell-1)} + p_{j-i} - b_{j-i,\ell})^\top \boldsymbol{Q}_A^{(\ell)} \boldsymbol{e}_i^{(\ell-1)} \\ =& \sum_{B,C\in\mathcal{N}} \Pr[A \to BC] \cdot \alpha(B, j+1, i) \cdot \alpha(C, i-\ell, j),\end{aligned}$$

if $i-\ell \le j \le i-1$ and 0 otherwise. Summing over all locations j gives us $\alpha(A, i-\ell, i)$.

Outside probabilities: In addition to all the inside probabilities, the contextual embeddings at position i after the computations of any layer $(2L-1)-\ell$ $(\ge L)$ contain the outside probabilities of all spans of length at least $\ell+1$ starting at position i, i.e. $\beta(A, i, i+k)$ for all $A \in \mathcal{N}$ and $k \ge \ell$. The rest of the coordinates contain 0.

Layer L In this layer, we initialize the outside probabilities $\beta(\text{ROOT}, 1, L) = 1$ and $\beta(A, 1, L) = 0$ for $A \ne \text{ROOT}$. Furthermore, we move the inside probabilities $\alpha(A, i-k, i-1)$ from position $i-1$ to position i using 1 attention head. For the attention head, $b_{-1,L}$ is set as 0, while the rest are set as ζ for some large constant ζ so that the attention heads only attend to position $i-1$ at any position i.

Layer $L+1 \le \tilde{\ell} := (2L-1)-\ell \le 2L-1$: At each position i, this layer computes the outside probabilities of spans of length $\ell+1$ starting at i, using the recursive formulation from eq. 3. The recursive formulation for $\beta(A, i, i+\ell)$ for a non-terminal $A \in \mathcal{N}$ has two terms, given by

$$\beta(A, i, i+\ell) = \beta_1(A, i, i+\ell) + \beta_2(A, i, i+\ell), \text{ with} \tag{15}$$

$$\begin{aligned}\beta_1(A, i, i+\ell) =& \sum_{j=1}^{i-1} \sum_{C,B\in\mathcal{N}} \Pr[B \to CA] \\ &\cdot \alpha(C, j, i-1)\beta(B, j, i+\ell), \text{ and}\end{aligned} \tag{16}$$

$$\begin{aligned}\beta_2(A, i, i+\ell) =& \sum_{j=i+\ell+1}^{L} \sum_{B,C\in\mathcal{N}} \Pr[B \to AC] \\ &\cdot \alpha(C, i+\ell+1, j)\beta(B, i, j).\end{aligned} \tag{17}$$

For each non-terminal $A \in \mathcal{N}$, we will use a single unique head to compute $\beta(A, i, i+\ell)$ with query matrix $\boldsymbol{Q}_A^{(\tilde{\ell})}$ and key matrix $\boldsymbol{K}_A^{(\tilde{\ell})}$. Combining the operations of both β_1 and β_2 in a single attention head is the main reason behind the decrease in the number of necessary attention heads, compared to Theorem 3.1.

Structure of $\{b_{t,\ell}\}_{-L\le t\le L, L+1\le\ell\le 2L-1}$: For any $L+1 \le \ell \le 2L-1$, for $0 \le t \le \ell+1$, $b_{t,\ell}$ is set as ζ for some large constant ζ. All other biases are set as 1.

Structure of Query and key matrices:

1. $\boldsymbol{K}_A^{(\tilde{\ell})}$ is a rotation matrix such that in $\boldsymbol{K}_A^{(\tilde{\ell})} \boldsymbol{e}_i^{(\ell)}$, for all $L > \ell_1 > \ell$, the outside probabilities $\{\beta(B, i, i+\ell_1)\}_{B\in\mathcal{N}}$ appears in the coordinates $[|\mathcal{N}|(\ell_1-\ell-1), |\mathcal{N}|(\ell_1-\ell))$. Furthermore, for all $0 \le \ell_1 \le L-\ell-2$, the inside probabilities $\{\alpha(C, i-1-\ell_1, i-1)\}_{C\in\mathcal{N}}$ appears in the coordinates $[|\mathcal{N}|(L+\ell+\ell_1+1), |\mathcal{N}|(L+\ell+\ell_1+2))$. Note that $\boldsymbol{K}_A^{(\tilde{\ell})}$ is same for all A, and only depends on ℓ.
2. The Query matrix $\boldsymbol{Q}_A^{(\tilde{\ell})}$ is a block diagonal matrix. If we define $\boldsymbol{P}_{A,r} \in \mathbb{R}^{|\mathcal{N}|\times|\mathcal{N}|}$ as a matrix that contains $\{\Pr[B \to CA]\}_{B,C\in\mathcal{N}}$, which is the set of all rules where A appears as the right child, $\boldsymbol{P}_{A,r}$ appears at positions $(|\mathcal{N}|\ell_1, |\mathcal{N}|\ell_1)$ for all $\ell_1 < L$, which is the set of the first L blocks along the diagonal. Furthermore, if we define $\boldsymbol{P}_{A,l} \in \mathbb{R}^{|\mathcal{N}|\times|\mathcal{N}|}$ as a matrix that contains $\{\Pr[B \to AC]\}_{B,C\in\mathcal{N}}$,

which is the set of all rules where A appears as the left child, $\boldsymbol{P}_{A,l}^{\top}$ appears at positions $(|\mathcal{N}|\ell_1, |\mathcal{N}|\ell_1)$ for all $\ell_1 \geq L + \ell + 1$, which is a set of $L - \ell - 2$ blocks along the diagonal located towards the end.

Intuition behind $\boldsymbol{Q}_A^{(\tilde{\ell})}$, $\boldsymbol{K}_A^{(\tilde{\ell})}$, the relative position embeddings and the biases: Considering any location i, we split the computation of $\beta(A, i, i+\ell)$ with the attention head into the computation of β_1 (eq. 16) and β_2 (eq. 17). For β_1, we express each term $\sum_{C,B\in\mathcal{N}} \Pr[B \to CA]\alpha(C, j, i-1)\beta(B, j, i+\ell)$ as the attention score $a_{i,j}$ and then express β_1 as $\sum_{j\leq i-1} a_{i,j}$. Similarly, for β_2, we express each term $\sum_{B,C\in\mathcal{N}} \Pr[B \to AC]\alpha(C, i+\ell+1, j)\beta(B, i, j)$ as the attention score $a_{i,j}$ and then express β_1 as $\sum_{j\geq i+\ell+1} a_{i,j}$. The relative position vectors and biases help to differentiate the operations on the left and right-hand sides of i, as we showcase below.

Computing β_1 (eq. 16): For any position i and $\ell_1 \geq 0$, $\boldsymbol{e}_i^{(\tilde{\ell}-1)}$ contains the inside probabilities $\{\alpha(C, i-1-\ell_1, i-1)\}_{C\in\mathcal{N}}$ in the coordinates $[|\mathcal{N}|\ell_1, |\mathcal{N}|(\ell_1+1))$. With the application of $\boldsymbol{K}_A^{(\tilde{\ell})}$, for $\ell_1 > \ell$, $\boldsymbol{K}_A^{(\tilde{\ell})}\boldsymbol{e}_i^{(\tilde{\ell}-1)}$ contains the outside probabilities $\{\beta(B, i, i+\ell_1)\}_{B\in\mathcal{N}}$ in the coordinates $[|\mathcal{N}|(\ell_1-\ell-1), |\mathcal{N}|(\ell_1-\ell))$. Hence, if we set the block at position $(|\mathcal{N}|\ell_1, |\mathcal{N}|\ell_1)$ in $\boldsymbol{Q}_A^{(\ell)}$ to $\boldsymbol{P}_{A,r}$ for some $L > \ell_1 \geq 0$, with the rest set to 0, we can get for any two positions i, j,

$$
\begin{aligned}
&(\boldsymbol{K}_A^{(\tilde{\ell})}\boldsymbol{e}_j^{(\tilde{\ell}-1)})^{\top}\boldsymbol{Q}_A^{(\tilde{\ell})}\boldsymbol{e}_i^{(\tilde{\ell}-1)} \\
&= \sum_{B,C\in\mathcal{N}} \Pr[B \to CA] \cdot \alpha(C, i-1-\ell_1, i-1) \\
&\quad \cdot \beta(B, j, j+\ell+\ell_1+1).
\end{aligned}
$$

Setting the first L diagonal blocks in $\boldsymbol{Q}_A^{(\tilde{\ell})}$ to $\boldsymbol{P}_{A,r}$ can get for any two positions i, j,

$$
\begin{aligned}
&(\boldsymbol{K}_A^{(\tilde{\ell})}\boldsymbol{e}_j^{(\tilde{\ell}-1)})^{\top}\boldsymbol{Q}_A^{(\tilde{\ell})}\boldsymbol{e}_i^{(\tilde{\ell}-1)} \\
&= \sum_{\ell_1\geq 0}\sum_{B,C\in\mathcal{N}} \Pr[B \to CA] \cdot \alpha(C, i-1-\ell_1, i-1) \\
&\quad \cdot \beta(B, j, j+\ell+\ell_1+1).
\end{aligned}
$$

However, for $\beta_1(A, i, i+\ell)$, the attention score above should only contribute with $\ell_1 = i-j-1$. Moreover, we also want the above sum to be 0 if $j \geq i$. Hence, we will use the relative position vector p_{j-i}, bias $b_{j-i,\tilde{\ell}}$ and the ReLU activation to satisfy the following conditions:

1. $j < i$.
2. The portion containing $\{\beta(B, j, j+\ell+\ell_1+1)\}_{C\in\mathcal{N}}$ in $\boldsymbol{K}_A^{(\tilde{\ell})}\boldsymbol{e}_j^{(\tilde{\ell}-1)}$ is activated only if $\ell_1 = i-j-1$.

For any positions i, j and $0 \leq \ell_1 \leq L$, $\boldsymbol{K}_A^{(\tilde{\ell})}\boldsymbol{e}_j^{(\tilde{\ell}-1)} + p_{j-i} - b_{j-i,\tilde{\ell}}$ will contain $\{\beta(B, j, j+\ell+\ell_1+1) + \mathbb{I}[\ell_1 = i-j-1] - 1 - \zeta\mathbb{I}[i \leq j \leq i+\ell]\}_{B\in\mathcal{N}}$ in coordinates $[|\mathcal{N}|\ell_1, |\mathcal{N}|(\ell_1+1))$, which will give us

$$
\begin{aligned}
&\mathrm{ReLU}(\boldsymbol{K}_A^{(\tilde{\ell})}\boldsymbol{e}_j^{(\tilde{\ell}-1)} + p_{j-i} - b_{j-i,\tilde{\ell}})^{\top}\boldsymbol{Q}_A^{(\tilde{\ell})}\boldsymbol{e}_i^{(\tilde{\ell}-1)} \\
&= \sum_{C,B\in\mathcal{N}} \Pr[B \to CA]\alpha(C, j, i-1)\beta(B, j, i+\ell),
\end{aligned}
$$

iff $j < i$ and 0 otherwise. Summing over all locations gives us $\beta_1(A, i, i+\ell)$.

Computing β_2 (eq. 17): For any position i and $L > \ell_1 > \ell$, $\boldsymbol{e}_i^{(\tilde{\ell}-1)}$ contains the outside probabilities $\{\beta(B, i, i+\ell_1)\}_{B\in\mathcal{N}}$ in the coordinates $[|\mathcal{N}|(L+\ell_1), |\mathcal{N}|(L+\ell_1+1))$. With the application of $\boldsymbol{K}_A^{(\tilde{\ell})}$, for $L > \ell_1 > \ell$, $\boldsymbol{K}_A^{(\tilde{\ell})}\boldsymbol{e}_i^{(\tilde{\ell}-1)}$ contains the inside probabilities $\{\alpha(C, i-1-\ell_1, i-1)\}_{C\in\mathcal{N}}$ in the coordinates $[|\mathcal{N}|(L+\ell+\ell_1+1), |\mathcal{N}|(L+\ell+\ell_1+2))$. Hence, if we set the block at position $(|\mathcal{N}|\ell_1, |\mathcal{N}|\ell_1)$ in $\boldsymbol{Q}_A^{(\tilde{\ell})}$ to $\boldsymbol{P}_{A,l}^{\top}$ for some $\ell_1 \geq L+\ell+1$, with the rest set to 0, we can get for any two positions i, j,

$$
\begin{aligned}
&(\boldsymbol{K}_A^{(\tilde{\ell})}\boldsymbol{e}_j^{(\tilde{\ell}-1)})^{\top}\boldsymbol{Q}_A^{(\tilde{\ell})}\boldsymbol{e}_i^{(\tilde{\ell}-1)} \\
&= \sum_{B,C\in\mathcal{N}} \Pr[B \to AC] \cdot \alpha(C, j-\ell_1+\ell+L, j-1) \\
&\quad \cdot \beta(B, i, i+\ell_1-L).
\end{aligned}
$$

Setting diagonal blocks at positions $\{(|\mathcal{N}|\ell_1, |\mathcal{N}|\ell_1)\}_{\ell_1\geq L+\ell+1}$ in $\boldsymbol{Q}_A^{(\tilde{\ell})}$ to $\boldsymbol{P}_{A,l}^{\top}$ can get for any two positions i, j,

$$
\begin{aligned}
&(\boldsymbol{K}_A^{(\tilde{\ell})}\boldsymbol{e}_j^{(\tilde{\ell}-1)})^{\top}\boldsymbol{Q}_A^{(\tilde{\ell})}\boldsymbol{e}_i^{(\tilde{\ell}-1)} \\
&= \sum_{\ell_1\geq\ell+1}\sum_{B,C\in\mathcal{N}} \Pr[B \to AC] \cdot \alpha(C, j-\ell_1+\ell, j-1) \\
&\quad \cdot \beta(B, i, i+\ell_1).
\end{aligned}
$$

However, for $\beta_1(A, i, i+\ell)$, the attention score above should only contribute with $\ell_1 = j-i-1$. Moreover, we also want the above sum to be 0 if $j \leq i+\ell$. We will use the relative position vector p_{j-i}, bias $b_{j-i,\tilde{\ell}}$ and the ReLU activation to satisfy the following conditions:

1. $j > i + \ell$.

2. The portion containing $\{\alpha(C, j - \ell_1 + \ell, j - 1)\}_{C \in \mathcal{N}}$ in $\boldsymbol{K}_A^{(\tilde{\ell})} \boldsymbol{e}_j^{(\tilde{\ell}-1)}$ is activated only if $\ell_1 = j - i - 1$.

Thus, for any positions i, j and $0 \leq \ell_1 \leq L$, $\boldsymbol{K}_A^{(\tilde{\ell})} \boldsymbol{e}_j^{(\tilde{\ell}-1)} + p_{j-i} - b_{j-i,\tilde{\ell}}$ will contain $\{\alpha(C, j - \ell_1 + \ell, j - 1) + \mathbb{I}[\ell_1 = i - j - 1] - 1 - \zeta \mathbb{I}[i \leq j \leq i + \ell]\}_{C \in \mathcal{N}}$ in coordinates $[|\mathcal{N}|\ell_1, |\mathcal{N}|(\ell_1 + 1))$, which will give us

$$
\begin{aligned}
&\mathrm{ReLU}(\boldsymbol{K}_A^{(\tilde{\ell})} \boldsymbol{e}_j^{(\tilde{\ell}-1)} + p_{j-i} - b_{j-i,\tilde{\ell}})^\top \boldsymbol{Q}_A^{(\tilde{\ell})} \boldsymbol{e}_i^{(\tilde{\ell}-1)} \\
&= \sum_{j=i+\ell+1}^{L} \sum_{B,C \in \mathcal{N}} \Pr[B \to AC] \alpha(C, i + \ell + 1, j) \beta(B, i, j),
\end{aligned}
$$

iff $j > i + \ell + 1$ and 0 otherwise. Summing over all locations gives us $\beta_2(A, i, i + \ell)$.

Computing $\beta_1 + \beta_2$ (eq. 15): From our construction, β_1 requires the dot product of the inside probabilities stored at the query vector and the outside probabilities stored at the key vector. However, β_2 requires the dot product of the outside probabilities stored at the query vector and the inside probabilities stored at the key vector. Since β_1 and β_2 are computed on the left and the right-hand side of the query respectively, we use the relative position embeddings to separate the two operations. The vector p_{j-i} activates only the outside probabilities in the key vector when $j > i$ and activates only the inside probabilities in the key vector when $j < i$. Thus, we can compute $\beta_1 + \beta_2$ as the sum of the attention scores of a single head, where the computation of β_1 and β_2 have been restricted to the left and the right-hand side of the query respectively.

B.3 Proof of Theorem 3.3

Proof of Theorem 3.3. We first focus on 1-mask predictions, where given an input of tokens $w_1, w_2, \cdots, w_L$, and a randomly selected index i, we need to predict the token at position i given the rest of the tokens, i.e. $\Pr\{w | w_{-i}\}$. Under the generative rules of the PCFG model, we have

$$
\begin{aligned}
&\Pr[w | w_{-i}] \\
&= \sum_A \Pr[A \to w] \cdot \Pr[A \text{ generates word at pos } i | w_{-i}] \\
&= \sum_A \Pr[A \to w] \cdot \frac{\beta(A, i, i)}{\sum_B \beta(B, i, i)}. \qquad (18)
\end{aligned}
$$

Note that $\Pr[A \to w]$ can be extracted from the PCFG and $\{\beta(B, i, i)\}_{B \in \mathcal{N}}$ can be computed by the Inside-outside algorithm. Thus, Inside-outside can solve the 1-masking problem optimally.

Now we consider the case where we randomly mask $m\%$ (e.g., 15%) of the tokens and predict these tokens given the rest. In this setting, if the original sentence is generated from PCFG $\mathcal{G} = (\mathcal{N}, \mathcal{I}, \mathcal{P}, n, p)$, one can modify the PCFG to get $\mathcal{G}' = (\mathcal{N}, \mathcal{I}, \mathcal{P}, n + 1, p')$ with $n + 1$ denote the mask token $text[MASK]$ and for each preterminal $A \in \mathcal{P}$, $p'(A \to \text{[MASK]}) = m\%$ and $p'(A \to w) = (1 - m\%) p(A \to w)$, for all $w \neq \text{[MASK]}$. Then, the distribution of the randomly masked sentences follows the distribution of sentences generated from the modified PCFG $\mathcal{G}'$. Similar to the 1-masking setting, we can use the Inside-outside algorithm to compute the optimal token distribution at a masked position.

□

C Omitted Details in Section 3.3

In Section 3.3, we claim that it is possible to approximately execute the Inside-Outside algorithm for PCFG learned on PTB dataset, and can drastically reduce the size of our constructed model with minimal impact on the 1-masking predictions and parsing performance (Theorem 3.4) by applying two ingredients: restricting the computations to few non-terminals and utilizing the underlying low-rank structure between the non-terminals. This section is organized as follows: In Appendix C.1, we show more intuition and experiment results on why we can restrict the computation of the inside-outside algorithm to a small subset of non-terminals. In Appendix C.2, we add more discussions on the second ingredient (utilizing the low-rank structure). Then in Appendix C.3, we show the details why restricting the computations of few non-terminals can reduce the size of the attention model. In Appendix C.4, we show the detailed proof of Theorem 3.4. Finally in Appendix C.5, we show the experiment details in Section 3.3.

C.1 More discussions on computation with few non-terminals

We hypothesize that we can focus only on a few non-terminals while retaining most of the performance.

Hypothesis C.1. *For the PCFG $\mathcal{G} = (\mathcal{N}, \mathcal{I}, \mathcal{P}, n, p)$ learned on the English cor-*

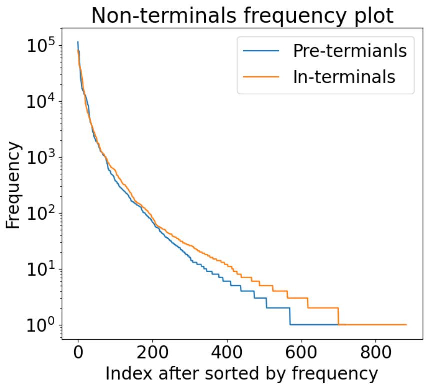


Figure 7: Plot for the frequency distribution of in-terminals ($\mathcal{I}$) and pre-terminals ($\mathcal{P}$). We compute the number of times a specific non-terminal appears in a span of a parse tree in the PTB training set. We then sort the non-terminals according to their normalized frequency and then show the frequency vs. index plot.

pus, there exists $\tilde{\mathcal{I}} \subset \mathcal{I}, \tilde{\mathcal{P}} \subset \mathcal{P}$ with $|\tilde{\mathcal{I}}| \ll |\mathcal{I}|, |\tilde{\mathcal{P}}| \ll |\mathcal{P}|$, such that simulating Inside-Outside algorithm with $\tilde{\mathcal{I}} \cup \tilde{\mathcal{P}}$ non-terminals introduces small error in the 1-mask perplexity and has minimal impact on the parsing performance of the Labeled-Recall algorithm.

To find candidate sets $\tilde{\mathcal{I}}, \tilde{\mathcal{P}}$ for our hypothesis, we check the frequency of different non-terminals appearing at the head of spans in the parse trees of the PTB (Marcus et al., 1993) training set. We consider the Chomsky-transformed (binarized) parse trees for sentences in the PTB training set, and collect the labeled spans $\{(A, i, j)\}$ from the parse trees of all sentences. For all non-terminals A, we compute $\text{freq}(A)$, which denotes the number of times non-terminal A appears at the head of a span. Figure 7 shows the plot of $\text{freq}(A)$ for in-terminals and pre-terminals, with the order of the non-terminals sorted by the magnitude of $\text{freq}(\cdot)$. We observe that an extremely small subset of non-terminals have high frequency, which allows us to restrict our computation for the inside and outside probabilities to the few top non-terminals sorted by their freq scores. We select the top frequent non-terminals as possible candidates for forming the set $\tilde{\mathcal{N}}$.

We verify the effect of restricting our computation to the frequent non-terminals on the 1-mask perplexity and the unlabeled F1 score of the approximate Inside-Outside algorithm in Table 1. Recall from Theorem 3.3, the 1-mask probability distribution for a given sentence $w_1, \cdots, w_L$ at any index i is given by Equation (18), and thus we can use Equation (18) to compute the 1-mask perplexity on the corpus. To measure the impact on 1-mask language modeling, we report the perplexity of the original and the approximate Inside-Outside algorithm on 200 sentences generated from PCFG.

We observe that restricting the computation to the top-40 and 45 frequent in-terminals and pre-terminals leads to $< 6.5\%$ increase in average 1-mask perplexity. Furthermore, the Labeled-Recall algorithm observes at most 4.24% drop from the F1 performance of the original PCFG. If we further restrict the computation to the top-20 and 45 in-terminals and pre-terminals, we can still get 71.91% sentence F1 score, and the increase in average 1-mask perplexity is less than 8.6%. However, restricting the computation to 10 in-terminals leads to at least 15% drop in parsing performance.

Thus combining Theorem 3.2 and Table 1, we have the following informal theorem.

Theorem C.2 (Informal)**.** *Given the PCFG $\mathcal{G} = (\mathcal{N}, \mathcal{I}, \mathcal{P}, n, p)$ learned on the English corpus, there exist subsets $\tilde{\mathcal{I}} \subset \mathcal{I}, \tilde{\mathcal{P}} \subset \mathcal{P}$ with $|\tilde{\mathcal{I}}| = 20, |\tilde{\mathcal{P}}| = 45$, and an attention model with soft relative attention modules (6) with embeddings of size $275 + 40L$, $2L + 1$ layers, and 20 attention heads in each layer, that can simulate the Inside-Outside algorithm restricted to $\tilde{\mathcal{I}}, \tilde{\mathcal{P}}$ on all sentences of length at most L generated from $\mathcal{G}$. The restriction introduces a 9.29% increase in average 1-mask perplexity and 8.71% drop in the parsing performance of the Labeled-Recall algorithm.*

If we plug in the average length $L \approx 25$ for sentences in PTB, we can get a model with 20 attention heads, 1275 hidden dimension, and 51 layers. Compared with the construction in Theorem 3.2, the size of the model is much closer to reality. The proof of Theorem C.2 is shown in Appendix C.3.

C.2 More discussions on low-rank approximation

We hypothesize that we can find linear transformation matrices $\{\boldsymbol{W}^{(\ell)}\}_{\ell \le L}$ that can reduce the computations while retaining most of the performance, and our hypothesis is formalized as follow:

Hypothesis C.3. *For the PCFG $\mathcal{G} = (\mathcal{N}, \mathcal{I}, \mathcal{P}, n, p)$ learned on the English corpus, there exists transformation matrices $\boldsymbol{W}^{(\ell)} \in \mathbb{R}^{k^{(\ell)} \times |\tilde{\mathcal{I}}|}$ for every $\ell \le L$, such that approximately simulating the Inside-Outside algorithm with $\{\boldsymbol{W}^{(\ell)}\}_{\ell \le L}$ introduces small error in the 1-mask perplexity and has minimal impact* *on the parsing performance of the Labeled-Recall algorithm.*

Table 2 verifies our hypothesis, and lead to Theorem 3.4. Compared with the parsing results from Theorem C.2, the corpus and sentence F1 scores are nearly the same, and we further reduce the number of attention heads in each layer from 20 to 15. If we only use 10 attention heads to approximately execute the Inside-Outside algorithm, we can still get 61.72% corpus F1 and 65.31% sentence F1 on PTB dataset, which is still much better than the Right-branching baseline. Theorem 3.4 shows that attention models with a size much closer to the real models (like BERT or RoBERTa) still have enough capacity to parse decently well (>70% sentence F1 on PTB).

It is also worth noting that approximately executing the Inside-Outside algorithm using the transformation matrices $\{\boldsymbol{W}^{(\ell)}\}_{\ell \leq L}$ is very different from reducing the size of the PCFG grammar, since we use different matrix $\boldsymbol{W}^{(\ell)}$ when computing the probabilities for spans with different length. If we choose to learn the same transformation matrix $\boldsymbol{W}$ for all the layers ℓ, the performance drops.

More discussions on the transformation matrix $\boldsymbol{W}^{(\ell)}$ We can observe that by introducing the transformation matrix $\boldsymbol{W}^{(\ell)}$ generalized the first ingredient that only computes a small set of in-terminals $\tilde{\mathcal{I}}$ and pre-terminals $\tilde{\mathcal{P}}$, and in theory we can directly learn the transformation matrix $\boldsymbol{W}^{(\ell)}$ from the original PCFG without reducing the size at first, i.e., $\boldsymbol{W}^{(\ell)} \in \mathbb{R}^{k^{(\ell)} \times |\mathcal{I}|}$. However empirically, if we directly learn $\boldsymbol{W}^{(\ell)}$ from all the in-terminals $\mathcal{I}$ but not from the top-20 frequent in-terminals $\tilde{\mathcal{I}}$, the performance drops. Thus, we choose to learn the matrix $\boldsymbol{W}^{(\ell)}$ starting from the most frequent in-terminals $\tilde{\mathcal{I}}$. One possible explanation is that the learning procedure is also heuristic, and certainly may not learn the best transformation matrix.

Besides, we use the same transformation matrix $\boldsymbol{W}^{(\ell)}$ when computing the inside and outside probabilities, and it is also natural to use different transformation matrices when computing the inside and outside probabilities. Recall that we learn the transformation $\boldsymbol{W}^{(\ell)}$ by the Eigenvalue decomposition on matrix $\boldsymbol{X}^{(\ell)}$, where $\boldsymbol{X}^{(\ell)} = \sum_s \boldsymbol{X}_s^{(\ell)} / \left\|\boldsymbol{X}_s^{(\ell)}\right\|_{\mathrm{F}}$ and $\boldsymbol{X}_s^{(\ell)} = \sum_{i,j:j-i=\ell} \boldsymbol{\mu}_s^{i,j} (\boldsymbol{\mu}_s^{i,j})^\top$. Then, we can also learn two matrices $\boldsymbol{W}_{\text{inside}}^{(\ell)}$ and $\boldsymbol{W}_{\text{outside}}^{(\ell)}$ through the Eigenvalue decomposition on matrices $\boldsymbol{X}_{\text{inside}}^{(\ell)}$ and $\boldsymbol{X}_{\text{outside}}^{(\ell)}$ respectively, where

$$
\begin{aligned}
\boldsymbol{X}_{\text{inside}}^{(\ell)} &= \sum_s \boldsymbol{X}_{s,\text{inside}}^{(\ell)} / \left\|\boldsymbol{X}_{s,\text{inside}}^{(\ell)}\right\|_{\mathrm{F}}, \\
\boldsymbol{X}_{s,\text{inside}}^{(\ell)} &= \sum_{i,j:j-i=\ell} \boldsymbol{\alpha}_s^{i,j} (\boldsymbol{\alpha}_s^{i,j})^\top, \\
\boldsymbol{X}_{\text{outside}}^{(\ell)} &= \sum_s \boldsymbol{X}_{s,\text{outside}}^{(\ell)} / \left\|\boldsymbol{X}_{s,\text{outside}}^{(\ell)}\right\|_{\mathrm{F}}, \\
\boldsymbol{X}_{s,\text{outside}}^{(\ell)} &= \sum_{i,j:j-i=\ell} \boldsymbol{\beta}_s^{i,j} (\boldsymbol{\beta}_s^{i,j})^\top.
\end{aligned}
$$

However empirically, we also find that the performance drops by using different transformation matrices for inside and outside probabilities computation, which may also be attributed to the non-optimality of our method to learn the transformation matrix.

C.3 Proof for Theorem C.2

Note that in both Theorem 3.1 and Theorem 3.2, in every layer $1 \leq \ell \leq L-1$, we use one attention head with parameters $\boldsymbol{K}_A^{(\ell)}, \boldsymbol{Q}_A^{(\ell)}, \boldsymbol{V}_A^{(\ell)}$ to compute all the inside probabilities $\alpha(A, i, j)$ for all spans with length $\ell + 1$, i.e. $j - i = \ell$. For layer $L + 1 \leq \ell \leq 2L - 1$, the model constructed in Theorem 3.1 uses two attention heads to compute the outside probabilities $\beta(A, i, j)$ for a specific non-terminal A for spans with length $2L - \ell$, and the model constructed in Theorem 3.2 uses one attention heads to compute the outside probabilities $\beta(A, i, j)$ for a specific non-terminal A for spans with length $2L - \ell$. Now to show how restricting the computations to certain non-terminals $\tilde{\mathcal{I}} \cup \tilde{\mathcal{P}}$ can reduce the size of the constructed models in Theorems 3.1 and 3.2 we classify the inside and outside probabilities into four categories: (1) the inside probabilities for pre-terminals, $\alpha(A, i, i)$ for $A \in \mathcal{P}$; (2) the inside probabilities for in-terminals, $\alpha(A, i, j)$ for $A \in \mathcal{I}$; (3) the outside probabilities for in-terminals, $\beta(A, i, j)$ for $A \in \mathcal{I}$; and (4) the outside probabilities for pre-terminals, $\beta(A, i, i)$ for $A \in \mathcal{P}$.

Category (1): the inside probabilities for pre-terminals Recall that in the constructed model in Theorems 3.1 and 3.2, the inside probabilities for pre-terminals $\alpha(A, i, i)$ for $A \in \mathcal{P}$ is directly initialized from the PCFG rules, and thus do not need attention heads to compute. Thus, we can just use $O(|\mathcal{P}|)$ dimensions to store all the inside probabilities for pre-terminals $\alpha(A, i, i)$ for $A \in \mathcal{P}$.

Although we can also only initialize the inside probabilities only for the pre-terminals $\tilde{\mathcal{P}}$, i.e. initialize $\alpha(A,i,i)$ for $A \in \tilde{\mathcal{P}}$ and use less embedding dimensions, empirically the performance will drop and thus we initialize all the probabilities $\alpha(A,i,i)$ for $A \in \mathcal{P}$. Although we should store the probabilities for pre-terminals larger than the set $\tilde{\mathcal{P}}$, there is indeed another technique to reduce the embedding dimensions. Note that since in the future computations, we only compute the probabilities for the in-terminals $\tilde{\mathcal{I}}$, and not every pre-terminal $A \in \mathcal{P}$ can be produced by in-terminals $B \in \tilde{\mathcal{I}}$. Thus, we only need to store the pre-terminals $\mathcal{P}_{\tilde{\mathcal{I}}}$ that can be produced from $\tilde{\mathcal{I}}$. Empirically, for PCFG learned on `PTB` dataset, $|\mathcal{P}| = 720$, but if we choose $|\tilde{\mathcal{I}}| = 20$, the number of pre-terminals that can be produced from $\tilde{\mathcal{I}}$ drops to $|\mathcal{P}_{\tilde{\mathcal{I}}}| = 268 < 270$. Specifically for the model in Theorem 3.2, we need $|\mathcal{P}_{\tilde{\mathcal{I}}}|$ coordinates at each position to store these inside probabilities.

Category (2): the inside probabilities for in-terminals The computation of the inside probabilities for in-terminals, $\alpha(A,i,j)$ for $A \in \mathcal{I}$ happens from layer 1 to layer $L-1$ in the constructed model in Theorems 3.1 and 3.2. Note that from layer 1 to layer $L-1$, the model only computes the probabilities for the in-terminals, since a span with a length larger than 1 cannot be labeled by a pre-terminal. Thus, if we only compute the inside probabilities for in-terminals $|\tilde{\mathcal{I}}|$, we can reduce the number of attention heads in layer 1 to layer $L-1$ from $O(|\mathcal{I}|)$ to $O(|\tilde{\mathcal{I}}|)$ since in Theorems 3.1 and 3.2 we use a constant number of attention heads to compute the probabilities for a single in-terminal. Specifically for the model in Theorem 3.2, we only need $|\tilde{\mathcal{I}}|$ attention heads from layer 1 to layer $L-1$; besides, we need $(L-1)|\tilde{\mathcal{I}}|$ coordinates at each position to store these inside probabilities.

Category (3): the outside probabilities for in-terminals The computation of the outside probabilities for in-terminals, $\beta(A,i,j)$ for $A \in \mathcal{I}$ happens from layer L to layer $L-2$ in the constructed model in Theorems 3.1 and 3.2. Note that in layer L, we only need to initialize the outside probabilities $\beta(A,1,L)$ for $A \in \mathcal{I}$, thus do not need attention heads for computation (however we need attention heads to move the inside and outside probabilities in this layer, which cost 2 attention heads). Then from layer $L+1$ to layer $L-2$, the model computes the outside probabilities for the in-terminals $\beta(A,i,j)$ for $A \in \tilde{\mathcal{I}}$. Thus if we only compute the outside probabilities for in-terminals $|\tilde{\mathcal{I}}|$, we can reduce the number of attention heads in layer 1 to layer $L-1$ from $O(|\mathcal{I}|)$ to $O(|\tilde{\mathcal{I}}|)$. Specifically for the model in Theorem 3.2, we only need $|\tilde{\mathcal{I}}|$ attention heads from layer L to layer $L-2$; besides, we need $(L-1)|\tilde{\mathcal{I}}|$ coordinates at each position to store these outside probabilities for in-terminals $\tilde{\mathcal{I}}$.

Category (4): the outside probabilities for pre-terminals The outside probabilities for pre-terminals $\beta(A,i,i)$ for $A \in \mathcal{P}$ is only computed in the final layer in Theorems 3.1 and 3.2. Thus if we choose to compute the probabilities for only $\tilde{\mathcal{P}}$, we can reduce the number of attention heads in layer $2L-1$ from $O(|\mathcal{I}|)$ to $O(|\tilde{\mathcal{I}}|)$. Specifically for the model in Theorem 3.2, we only need $|\tilde{\mathcal{P}}|$ attention heads in layer $L-1$; besides, we need $|\tilde{\mathcal{P}}|$ coordinates at each position to store these outside probabilities for in-terminals $\tilde{\mathcal{P}}$. Also as mentioned in Section 3.3, if $|\tilde{\mathcal{P}}| < c|\tilde{\mathcal{I}}|$ for some constant c, we can also simulate the computations in the last layer with $|\tilde{\mathcal{P}}|$ heads by c layers with $|\tilde{\mathcal{I}}|$ heads. In particular, if we choose $|\tilde{\mathcal{P}}| = 45, |\tilde{\mathcal{I}}| = 20$, we can use 3 layers with 20 attention heads in each layer to simulate the last layer with 45 attention heads in the original construction.

Put everything together: proof of Theorem C.2 We choose $|\tilde{\mathcal{P}}| = 45, |\tilde{\mathcal{I}}| = 20$. We can use 20 attention heads in each layer, and we now count the number of layers and the embedding dimension we need. The number of layers is easy to compute, since we just need to use 3 layers with 20 attention heads to simulate the original 1 layer with 45 attention heads, thus the total number of layers is $2L-1+(3-1) = 2L+1$. As for the embedding dimension, we need

$$\begin{aligned}
d =& |\mathcal{P}_{\tilde{\mathcal{I}}}| + (L-1)|\tilde{\mathcal{I}}| + (L-1)|\tilde{\mathcal{I}}| + |\tilde{\mathcal{P}}| \\
\le& 270 + (2L-2)|\tilde{\mathcal{I}}| + |\tilde{\mathcal{P}}| \\
=& 275 + 2L|\tilde{\mathcal{I}}| \\
=& 275 + 40L.
\end{aligned}$$

C.4 Proof for Theorem 3.4

In this section, we show the details of how to further reduce the number of attention heads using structures across non-terminals, and add more discussion on how we learn the transformation matrices $\{\boldsymbol{W}^{(\ell)}\}_{\ell \le L}$

Reducing the number of attention heads We focus on reducing the number of attention heads to compute the inside and outside probabilities for the in-terminals $\tilde{\mathcal{I}}$, since the computation for the outside probabilities for pre-terminals $\tilde{\mathcal{P}}$ only happens in the final layer of the constructed model, and thus can use multiple layers to compute as long as $\tilde{\mathcal{P}}$ is not too large.

For simplicity, we only show the details of how to reduce the number of attention heads to compute the inside probabilities for in-terminals $\tilde{\mathcal{I}}$ in Theorem 3.2, and the technique can be easily applied to the computation of outside probabilities for in-terminals $\tilde{\mathcal{I}}$ in Theorem 3.2, and the inside and outside probabilities for $\tilde{\mathcal{I}}$ in Theorem 3.1.

Recall from the proof of Theorem 3.2 that we at each layer ℓ, we use a single attention head $\boldsymbol{K}_A^{(\ell)}, \boldsymbol{Q}_A^{(\ell)}$ to compute the inside probabilities $\alpha(A, i, j)$ for spans with length $\ell+1$, i.e., $j-i=\ell$. Specifically, for the attention head $\boldsymbol{K}_A^{(\ell)}, \boldsymbol{Q}_A^{(\ell)}$ at layer ℓ, we want to compute and store the probability $\alpha(A, i-\ell, i)$ at position i. Thus we construct $\boldsymbol{K}_A^{(\ell)}, \boldsymbol{Q}_A^{(\ell)}$ such that the attention score $a_{i,j}^{A,(\ell)}$ when the position i attends to position j satisfies

$$
\begin{aligned}
& a_{i,j}^{A,(\ell)} \\
=& \mathrm{ReLU}(\boldsymbol{K}_A^{(\ell)} \boldsymbol{e}_j^{(\ell-1)} + p_{j-i} - b_{j-i,\ell})^\top \boldsymbol{Q}_A^{(\ell)} \boldsymbol{e}_i^{(\ell-1)} \\
=& \sum_{B,C \in \mathcal{N}} \Pr[A \to BC] \cdot \alpha(B, j+1, i) \cdot \alpha(C, i-\ell, j),
\end{aligned}
$$

if $i - \ell \leq j \leq i-1$ and 0 otherwise. Then, summing over all locations j gives us $\alpha(A, i-\ell, i)$. Also, a key property of $\boldsymbol{K}_A^{(\ell)}$ is that this key matrix does not depend on the non-terminal A, but only depends on ℓ. Thus, if we have a set of coefficients $\{\omega_A^{(\ell)}\}_{A \in \mathcal{I}}$, we can compute the linear combination of the inside probability $\sum_{A \in \tilde{\mathcal{I}}} \omega_A^{(\ell)} \alpha(A, i-\ell, i)$ using one attention head, since if we choose

$$
\boldsymbol{Q}^{(\ell)} = \sum_{A \in \tilde{\mathcal{I}}} \omega_A^{(\ell)} \boldsymbol{Q}_A^{(\ell)}, \quad \boldsymbol{K}^{(\ell)} = \boldsymbol{K}_A^{(\ell)}, \forall A \in \tilde{\mathcal{I}},
$$

we have the attention score

$$
\begin{aligned}
& a_{i,j}^{(\ell)} \\
=& \mathrm{ReLU}(\boldsymbol{K}^{(\ell)} \boldsymbol{e}_j^{(\ell-1)} + p_{j-i} - b_{j-i,\ell})^\top \boldsymbol{Q}^{(\ell)} \boldsymbol{e}_i^{(\ell-1)} \\
=& \mathrm{ReLU}(\boldsymbol{K}^{(\ell)} \boldsymbol{e}_j^{(\ell-1)} + p_{j-i} - b_{j-i,\ell})^\top \\
& \cdot \left(\sum_{A \in \tilde{\mathcal{I}}} \omega_A^{(\ell)} \boldsymbol{Q}_A^{(\ell)} \right) \boldsymbol{e}_i^{(\ell-1)} \\
=& \sum_{A \in \tilde{\mathcal{I}}} \omega_A^{(\ell)} \\
& \cdot \mathrm{ReLU}(\boldsymbol{K}^{(\ell)} \boldsymbol{e}_j^{(\ell-1)} + p_{j-i} - b_{j-i,\ell})^\top \boldsymbol{Q}_A^{(\ell)} \boldsymbol{e}_i^{(\ell-1)} \\
=& \sum_{A \in \tilde{\mathcal{I}}} \omega_A^{(\ell)} \\
& \cdot \mathrm{ReLU}(\boldsymbol{K}_A^{(\ell)} \boldsymbol{e}_j^{(\ell-1)} + p_{j-i} - b_{j-i,\ell})^\top \boldsymbol{Q}_A^{(\ell)} \boldsymbol{e}_i^{(\ell-1)} \\
=& \sum_{A \in \tilde{\mathcal{I}}} \omega_A^{(\ell)} \\
& \cdot \left(\sum_{B,C \in \mathcal{N}} \Pr[A \to BC] \cdot \alpha(B, j+1, i) \cdot \alpha(C, i-\ell, j) \right),
\end{aligned}
$$

if $i - \ell \leq j \leq i-1$ and 0 otherwise. Then, summing over all locations j gives us $\sum_{A \in \tilde{\mathcal{I}}} \omega_A^{(\ell)} \alpha(A, i-\ell, i)$. Then if we have a transformation matrix $\boldsymbol{W}^{(\ell)} \in \mathbb{R}^{k^{(\ell)} \times |\tilde{\mathcal{I}}|}$, we can use $k^{(\ell)}$ attention heads to compute $\boldsymbol{W}^{(\ell)} \boldsymbol{\alpha}(i-\ell, i)$, where $\boldsymbol{\alpha}(i-\ell, i) \in \mathbb{R}^{|\tilde{\mathcal{I}}|}$ is the vector that contains $\alpha(A, i-\ell, i)$ for all $A \in \tilde{\mathcal{I}}$. Then after we use $k^{(\ell)}$ attention heads to compute the probabilities $\boldsymbol{W}^{(\ell)} \boldsymbol{\alpha}(i-\ell, i)$ and stored them in position i's embeddings, we can then use linear layer on position i to recover the original probabilities by $\tilde{\boldsymbol{\alpha}}(i-\ell, i) = (\boldsymbol{W}^{(\ell)})^\dagger \boldsymbol{W}^{(\ell)} \boldsymbol{\alpha}(i-\ell, i)$, and use $\tilde{\alpha}(A, i-\ell, i)$ for $A \in \tilde{\mathcal{I}}$ for the future computations.

Put everything together: proof of Theorem 3.4 We choose $k^{(\ell)} = 15, |\tilde{\mathcal{P}}| = 45, |\tilde{\mathcal{I}}| = 20$. Note that the embedding dimension doesn't change if we apply the approximation technique, and only the number of attention heads reduces from 20 to 15. Thus, the embedding dimension is still

$$
\begin{aligned}
d =& |\mathcal{P}_{\tilde{\mathcal{I}}}| + (L-1)|\tilde{\mathcal{I}}| + (L-1)|\tilde{\mathcal{I}}| + |\tilde{\mathcal{P}}| \\
\leq& 270 + (2L-2)|\tilde{\mathcal{I}}| + |\tilde{\mathcal{P}}| \\
=& 275 + 2L|\tilde{\mathcal{I}}| \\
=& 275 + 40L.
\end{aligned}
$$

Also note that $|\tilde{\mathcal{P}}| = 45 = 3 \times 15$, and thus we can compute all the outside probabilities for pre-terminals $\tilde{\mathcal{P}}$ by 3 layers where each layer has 15 attention heads.

C.5 Experiment details in Section 3.3

In this section, we provide the experiment details in Section 3.3. We use and modify the code (Peng, 2021) to learn the PCFG from the PTB dataset and conduct the experiments with approximated computations. Peng (2021) implements the spectral learning method to learn PCFG (Cohen et al., 2012, 2014) and is under MIT licence. We follow all the default hyperparameters in Peng (2021), and we also follow the split of PTB: using PTB section 02-21 as the training set and PTB section 22 as the development set.