R2D2: Recursive Transformer based on Differentiable Tree for Interpretable Hierarchical Language Modeling

Xiang Hu[†*] **Haitao Mi**[†*] **Zujie Wen**[†] **Yafang Wang**[†]
Yi Su[†] **Jing Zheng**[†] **Gerard de Melo**[‡]
Ant Financial Services Group[†]
{aaron.hx, haitao.mi, zujie.wzj, yafang.wyf, yi.su, jing.zheng}
@alibaba-inc.com[†]
Hasso Plattner Institute / University of Potsdam[‡]
gdm@demelo.org[‡]

Abstract

Human language understanding operates at multiple levels of granularity (e.g., words, phrases, and sentences) with increasing levels of abstraction that can be hierarchically combined. However, existing deep models with stacked layers do not explicitly model any sort of hierarchical process. This paper proposes a recursive Transformer model based on differentiable CKY style binary trees to emulate the composition process. We extend the bidirectional language model pre-training objective to this architecture, attempting to predict each word given its left and right abstraction nodes. To scale up our approach, we also introduce an efficient pruned tree induction algorithm to enable encoding in just a linear number of composition steps. Experimental results on language modeling and unsupervised parsing show the effectiveness of our approach.[1]

1 Introduction

The idea of devising a structural model of language capable of learning both representations and meaningful syntactic structure without any human-annotated trees has been a long-standing but challenging goal. Across a diverse range of linguistic theories, human language is assumed to possess a recursive hierarchical structure (Chomsky, 1956, 2014; de Marneffe et al., 2006) such that lower-level meaning is combined to infer higher-level semantics. Humans possess notions of characters, words, phrases, and sentences, which children naturally learn to segment and combine.

Pretrained language models such as BERT (Devlin et al., 2019) have achieved substantial gains across a range of tasks. However, they simply apply layer-stacking with a fixed depth to increase the modeling power (Bengio, 2009; Salakhutdinov, 2014). Moreover, as the core Transformer component (Vaswani et al., 2017) does not capture positional information, one also needs to incorporate additional positional embeddings. Thus, pretrained language models do not explicitly reflect the hierarchical structure of linguistic understanding.

Inspired by Le and Zuidema (2015), Maillard et al. (2017) proposed a fully differentiable CKY parser to model the hierarchical process explicitly. To make their parser differentiable, they primarily introduce an energy function to combine all possible derivations when constructing each cell representation. However, their model is based on Tree-LSTMs (Tai et al., 2015; Zhu et al., 2015) and requires $O(n^3)$ time complexity. Hence, it is hard to scale up to large training data.

In this paper, we revisit these ideas, and propose a model applying recursive Transformers along differentiable trees (R2D2). To obtain differentiability, we adopt Gumbel-Softmax estimation (Jang et al., 2017) as an elegant solution. Our encoder parser operates in a bottom-up fashion akin to CKY parsing, yet runs in linear time with regard to the number of composition steps, thanks to a novel pruned tree induction algorithm. As a training objective, the model seeks to recover each word in a sentence given its left and right syntax nodes. Thus, our model does not require any positional embedding and does not need to mask any words during training. Figure 1 presents an example binary tree induced by our method: Without any syntactic supervision, it acquires a model of hierarchical construction from the word-piece level to words, phrases, and finally the sentence level.

[*]Equal contribution.
[1]The code is available at: https://github.com/alipay/StructuredLM_RTDT

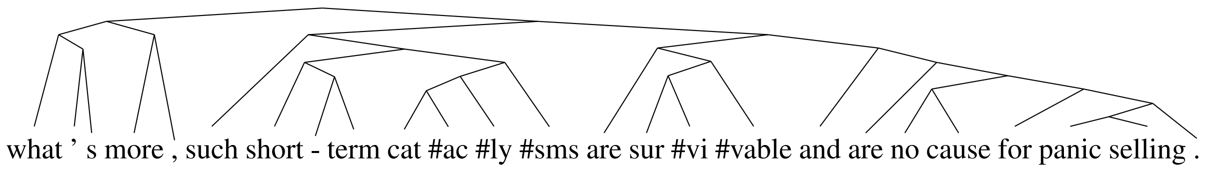

what ' s more , such short - term cat #ac #ly #sms are sur #vi #vable and are no cause for panic selling .

Figure 1: An example output tree emerging from our proposed method.

We make the following contributions:

- Our novel CKY-based recursive Transformer on differentiable trees model is able to learn both representations and tree structure (Section 2.1).
- We propose an efficient optimization algorithm to scale up our approach to a linear number of composition steps (Section 2.2).
- We design an effective pre-training objective, which predicts each word given its left and right syntactic nodes (Section 2.3).

For simplicity and efficiency reasons, in this paper we conduct experiments only on the tasks of language modeling and unsupervised tree induction. The experimental results on language modeling show that our model significantly outperforms baseline models with same parameter size even in fewer training epochs. At unsupervised parsing, our model as well obtains competitive results.

2 Methodology

2.1 Model Architecture

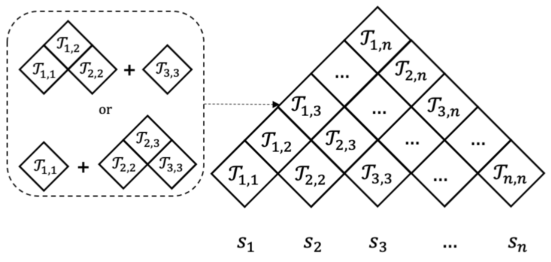


Figure 2: Chart data structure. There are two alternative ways of generating $\mathcal{T}_{1,3}$: combining either ($\mathcal{T}_{1,2}$, $\mathcal{T}_{3,3}$) or ($\mathcal{T}_{1,1}$, $\mathcal{T}_{2,3}$).

Differentiable Tree. We follow Maillard et al. (2017) in defining a differentiable binary parser using a CKY-style (Cocke, 1969; Kasami, 1966; Younger, 1967) encoder. Informally, given a sentence $\mathbf{S} = \{s_1, s_2, ..., s_n\}$ with n words or word-pieces, Figure 2 shows the chart data structure $\mathcal{T}$, where each cell $\mathcal{T}_{i,j}$ is a tuple $\langle e_{i,j}, p_{i,j}, \widetilde{p}_{i,j}\rangle$, $e_{i,j}$ is a vector representation, $p_{i,j}$ is the probability of a single composition step, and $\widetilde{p}_{i,j}$ is the probability of the subtree at span $[i, j]$ over sub-string $s_{i:j}$. At the lowest level, we have terminal nodes $\mathcal{T}_{i,i}$ with $e_{i,i}$ initialized as embeddings of inputs s_i, while $p_{i,i}$ and $\widetilde{p}_{i,i}$ are set to one. When $j > i$, the representation $e_{i,j}$ is a weighted sum of intermediate combinations $c^k_{i,j}$, defined as:

$$c^k_{i,j},\ p^k_{i,j} = f(e_{i,k}, e_{k+1,j}) \tag{1}$$

$$\widetilde{p}^k_{i,j} = p^k_{i,j}\ \widetilde{p}_{i,k}\ \widetilde{p}_{k+1,j} \tag{2}$$

$$\boldsymbol{\alpha}_{i,j} = \text{GUMBEL}(\log(\widetilde{\mathbf{p}}_{i,j})) \tag{3}$$

$$e_{i,j} = [c^i_{i,j}, c^{i+1}_{i,j}, ..., c^{j-1}_{i,j}]\boldsymbol{\alpha}_{i,j} \tag{4}$$

$$[p_{i,j}, \widetilde{p}_{i,j}] = \boldsymbol{\alpha}^{\intercal}_{i,j}[\boldsymbol{p}_{i,j}, \widetilde{\boldsymbol{p}}_{i,j}] \tag{5}$$

Here, k is a split point from i to $j-1$, $f(\cdot)$ is a composition function that we shall further define later on, $p^k_{i,j}$ and $\widetilde{p}^k_{i,j}$ denote the single step combination probability and the subtree probability, respectively, at split point k, $\boldsymbol{p}_{i,j}$ and $\widetilde{\boldsymbol{p}}_{i,j}$ are the concatenation of all $p^k_{i,j}$ or $\widetilde{p}^k_{i,j}$ values, and GUMBEL is the Straight-Through Gumbel-Softmax operation of Jang et al. (2017) with temperature set to one. The $[,]$ notation denotes stacking of tensors.

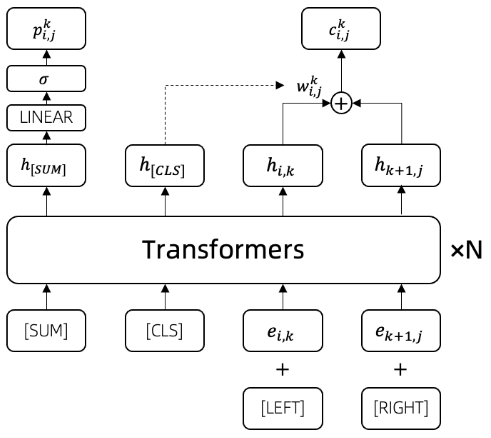


Figure 3: Recursive Transformer-based encoder.

Recursive Transformer. Figure 3 provides a schematic overview of the composition function $f(\cdot)$, comprising N Transformer layers. Taking $c^k_{i,j}$ and $p^k_{i,j}$ as an example, the input is a concatenation of two special tokens `[SUM]` and `[CLS]`, the left cell $e_{i,k}$, and the right cell $e_{k+1,j}$. We also add role embeddings (`[LEFT]` and `[RIGHT]`) to the left and right inputs, respectively. Thus, the input consists of four vectors in $\mathbb{R}^d$. We denote as $h_{\texttt{[SUM]}}, h_{\texttt{[CLS]}}, h_{i,k}, h_{k+1,j} \in \mathbb{R}^d$ the hidden state of the output of N Transformer layers. This is followed by a linear layer over $h_{\texttt{[SUM]}}$ to obtain

$$p^k_{i,j} = \sigma(W_p h_{\texttt{[SUM]}} + b_p), \tag{6}$$

where $W_p \in \mathbb{R}^{1\times d}$, $b_p \in \mathbb{R}$, and σ refers to sigmoid activation. Then, $c^k_{i,j}$ is computed as

$$\begin{aligned} w^k_{i,j} &= \mathrm{softmax}(W_w h_{\texttt{[CLS]}} + b_w) \\ c^k_{i,j} &= [h_{i,k}, h_{k+1,j}] w^k_{i,j}, \end{aligned} \tag{7}$$

where $W_w \in \mathbb{R}^{2\times d}$ with $w^k_{i,j} \in \mathbb{R}^2$ capturing the respective weights of the left and right hidden states $h_{i,k}$ and $h_{k+1,j}$, and the final $c^k_{i,j}$ is a weighted sum of $h_{i,k}$ and $h_{k+1,j}$.

Tree Recovery. As the Straight-Through Gumbel-Softmax picks the optimal splitting point k at each cell in practice, it is straightforward to recover the complete derivation tree, $\texttt{Tree}(\mathcal{T}_{1,n})$, from the root node $\mathcal{T}_{1,n}$ in a top-down manner recursively.

2.2 Complexity Optimization

Algorithm 1 Pruned Tree Induction Algorithm

```
Require: $\mathcal{T}$ = 2-d array holding cell references
Require: $m$ = pruning threshold
 1: function PRUNING($\mathcal{T}$, $m$)
 2:     $u \leftarrow$ FIND ($\mathcal{T}$, $m$)          ▷ Find optimal merge point
 3:     $n \leftarrow \mathcal{T}.len$
 4:     $\mathcal{T}' \leftarrow [n-1][n-1]$          ▷ Create a new 2-d array
 5:     for $i \in 1$ to $n-1$ do
 6:         for $j \in i$ to $n-1$ do
 7:             $i' \leftarrow i \geq u+1\ ?\ i+1\ :\ i$
 8:             $j' \leftarrow j \geq u\ ?\ j+1\ :\ j$
 9:             $\mathcal{T}'_{i,j} \leftarrow \mathcal{T}_{i',j'}$ ▷ Skip dark gray cells in Fig. 4
10:     return $\mathcal{T}'$
11: function TREEINDUCTION($\mathcal{T}$, $m$)
12:     $\mathcal{T}' \leftarrow \mathcal{T}$
13:     for $t \in 1$ to $\mathcal{T}.len - 1$ do
14:         if $t \geq m$ then
15:             $\mathcal{T}' \leftarrow$ PRUNING ($\mathcal{T}'$,$m$)
16:         $l \leftarrow \min(t+1, m)$          ▷ Clamp the span length
17:         for $i \in 1$ to $\mathcal{T}'.len - l + 1$ do
18:             $j \leftarrow i + l - 1$
19:             if $\mathcal{T}'_{i,j}$ is empty then
20:                 Compute cell $\mathcal{T}'_{i,j}$ with Equation 1
21:     return $\mathcal{T}$
```

As the core computation comes from the composition function $f(\cdot)$, our *pruned tree induction algorithm* aims to reduce the number of composition calls from $O(n^3)$ in the original CKY algorithm to linear.

Our intuition is based on the conjecture that locally optimal compositions are likely to be retained and participate in higher-level feature combination. Specifically, taking $\mathcal{T}^2$ in Figure 4 (c) as an example, we only pick locally optimal nodes from the second row of $\mathcal{T}^2$. If $\mathcal{T}^2_{4,5}$ is locally optimal and non-splittable, then all the cells highlighted in dark gray in (d) may be pruned, as they break span $[4, 5]$. For any later encoding, including higher-level ones, we can merge the nodes and treat $\mathcal{T}^2_{4,5}$ as a new non-splittable terminal node (see (e) to (g)).

Algorithm 2 Find the best merge point

```
Require: $\mathcal{T}$ = 2-d array holding cell references
Require: $m$ = pruning threshold
 1: function FIND($\mathcal{T}$, $m$)
 2:     $n \leftarrow \mathcal{T}.len$
 3:     $\mathcal{L} \leftarrow [n-1]$                    ▷ Create an array
 4:     for $i \in 1$ to $n-1$ do
 5:         $\mathcal{L}[i] \leftarrow \mathcal{T}_{i,i+1}$        ▷ Collect cells on the 2nd row
 6:     $\tau \leftarrow \emptyset$
 7:     for $i \in 1$ to $n-m+1$ do         ▷ Iterate to $m$-th row
 8:         $j = i + m - 1$
 9:         $\tau \leftarrow \tau \cup \{c \mid c \in \texttt{Tree}(\mathcal{T}_{i,j}) \wedge c \in \mathcal{L}\}$
10:     $l \leftarrow$ new List()
11:     for cell $x \in \tau$ do
12:         $i \leftarrow \mathcal{L}$.index($x$)
13:         $\overline{p}_l \leftarrow 1 - \mathcal{L}[i-1].p$
14:         $\overline{p}_r \leftarrow 1 - \mathcal{L}[i+1].p$
15:                         ▷ If index out of boundary then set to 0
16:         $l$.append($x.p \cdot \overline{p}_l \cdot \overline{p}_r$)
17:     return $\mathrm{argmax}_i\ l[i]$
```

Figure 4 walks through the steps of processing a sentence of length 6, where $s_{i:j}$ denotes a substring from s_i to s_j. Algorithm 1 constructs our chart table $\mathcal{T}$ sequentially row-by-row. Let t be the time step and m be the pruning threshold. First, we invoke TREEINDUCTION $(\mathcal{T}, m)$, and compute a row of cells at each time step when $t < m$ as in regular CKY parsing, leading to result (b) in Figure 4. When $t \geq m$, we call PRUNING $(\mathcal{T}, m)$ in Line 15. As mentioned, the PRUNING function aims to find the locally optimal combination node in $\mathcal{T}$, prunes some cells, and returns a new table omitting the pruned cells. Algorithm 2 shows how we FIND the locally optimal combination node. Again, the candidate set for the locally optimal node is the second row of $\mathcal{T}$, and we also take advantage of the subtrees derived from all nodes in the m-th row to limit the candidate set. Lines 6 to 9 in Algorithm 2 generate the candidate set. Each candidate must be in the second row of $\mathcal{T}$ and also must be used in a subtree of any node in the m-th row. Given the candidate set, we find the least ambiguous one as the optimal selection (Lines 11 to

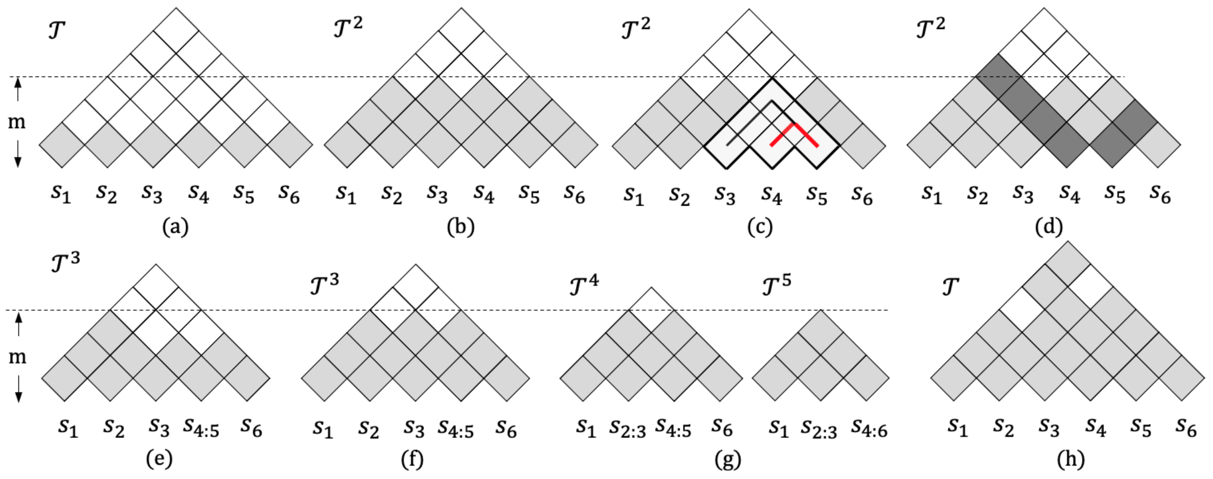


Figure 4: Example of encoding. (a) Initialized chart table. (b) Row-by-row encoding up to pruning threshold m. (c) For each cell in the m-th row, recover its subtree and collect candidate nodes, each of which must appear in the subtree and also must be in the 2nd row, e.g., the tree of $\mathcal{T}^2_{3,5}$ is within the dark line, and the candidate node is $\mathcal{T}^2_{4,5}$. (d) Find locally optimal node, which is $\mathcal{T}^2_{4,5}$ here, and treat span $s_{4:5}$ as non-splittable. Thus, the dark gray cells become prunable. (e) Construct a new chart table $\mathcal{T}^3$ treating cell $\mathcal{T}^2_{4,5}$ as a new *terminal* node and eliminating the prunable cells. (f) Compute empty cells in m-th row. (g) Keep pruning and growing the tree until no further empty cells remain. (h) Final discrete chart table.

17), i.e., the node with maximum own probability while adjacent bi-gram node probabilities (Lines 13 and 14) are as low as possible. After selecting the best merge point u, cells in $\{\mathcal{T}^t_{i,j} \mid j = u\} \cup \{\mathcal{T}^t_{i,j} \mid i = u+1\}$ are pruned (highlighted in dark gray in (d)), and we generate a new table $\mathcal{T}^{t+1}$ by removing pruned nodes (Lines 4 to 9 in Algorithm 1). Then we obtain (e), and compute the empty cells on the m-th row of $\mathcal{T}^3$ to obtain (f). We continue with the loop in Line 13, trigger PRUNING again, and obtain a new table $\mathcal{T}^{t+1}$, and then fill empty cells on the m-th row $\mathcal{T}^{t+1}$. Continuing with the process until all cells are computed, as shown in (g), we finally obtain a discrete chart table as given in (h).

In terms of the time complexity, when $t \geq m$, there are at most m cells to update, so the complexity of each step is less than $O(m^2)$. When $t \leq m$, the complexity is $O(t^3) \leq O(m^2 t)$. Thus, the overall times to call the composition function is $O(m^2 n)$, which is linear considering m is a constant.

2.3 Pretraining

Different from the masked language model training of BERT, we directly minimize the sum of all negative log probabilities of all words or word-pieces s_i given their left and right contexts.

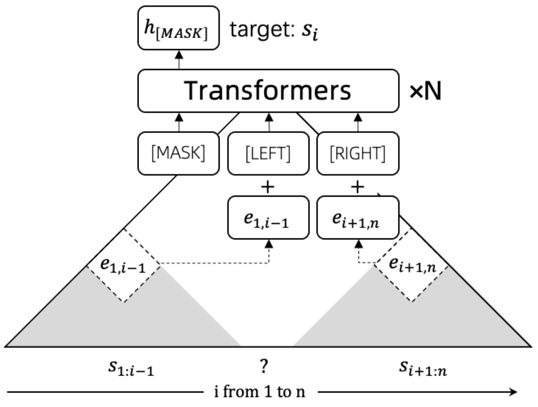


Figure 5: The objective for our pretrained model.

$$\min_{\theta} \sum_{i=1}^{n} -\log p_\theta(s_i \mid s_{1:i-1}, s_{i+1:n}) \qquad (8)$$

As shown in Figure 5, after invoking our recursive encoder on a sentence **S**, we directly use $e_{1,i-1}$ and $e_{i+1,n}$ as the left and right contexts, respectively, for each word s_i. To distinguish from the encoding task, the input consists of a concatenation of a special token `[MASK]`, $e_{1,i-1}$, and $e_{i+1,n}$. We apply the same composition function $f(\cdot)$ as in Figure 3, and feed $h_{\texttt{[MASK]}}$ through an output softmax to predict the distribution of s_i over the complete vocabulary. Finally, we compute the cross-entropy over the prediction and ground truth distributions.

In cases where $e_{1,i-1}$ or $e_{i+1,n}$ is missing due to the pruning algorithm in Section 2.2, we simply use the left or right longest adjacent non-empty cell. For example, $\mathcal{T}_{x,i-1}$ means the longest non-empty cell assuming we cannot find any non-empty $\mathcal{T}_{x',i-1}$ for all $x' < x$. Analogously, $\mathcal{T}_{i+1,y}$ is defined as the longest non-empty right cell. Note that although the final table is sparse, the sentence representation $e_{1,n}$ is always established.

3 Experiments

As our approach (R2D2) is able to learn both representations and intermediate structure, we evaluate its representation learning ability on bidirectional language modeling and evaluate the intermediate structures on unsupervised parsing.

3.1 Bidirectional Language Modeling

3.1.1 Setup

Baselines and Evaluation. As the objective of our model is to predict each word with its left and right context, we use the *pseudo-perplexity* (PPPL) metric of Salazar et al. (2020) to evaluate bidirectional language modeling.

$$\mathcal{L}(\mathbf{S}) = \frac{1}{n}\sum_{i=1}^{n} \log P(s_i \mid s_{1:i-1}, s_{i+1:n}, \theta)$$

$$\text{PPPL}(\mathbf{S}) = \exp\left(-\frac{1}{N}\sum_{j=1}^{N} \mathcal{L}(\mathbf{S}^j)\right)$$

PPPL is a bidirectional version of perplexity, establishing a macroscopic assessment of the model's ability to deal with diverse linguistic phenomena.

We compared our approach with SOTA autoencoding and autoregressive language models capable of capturing bidirectional contexts, including BERT, XLNet (Yang et al., 2019), and ALBERT (Lan et al., 2020). For a fair apples to apples comparison, all models use the same vocabulary and are trained from scratch on a language modeling corpus. The models are all based on the open source Transformers library[2]. To compute PPPL for models based on sequential Transformers, for each word s_i, we only mask s_i while others remain visible to predict s_i. When we evaluate our R2D2 model, for each word s_i, we treat the left $s_{1:i-1}$ and right $s_{i+1:n}$ as two complete sentences separately, then encode them separately, and pick the root nodes as the final representations of left and right contexts. In the end, we predict word s_i by running our Transformers as in Figure 5.

[2]https://github.com/huggingface/transformers

	#param	#layer	#epoch	cplx	PPPL
BERT	46M	3	10	$O(n^2)$	441.42
XLNet	46M	3	10	$O(n)$	301.87
ALBERT	46M	12	10	$O(n^2)$	219.20
XLNet	116M	12	10	$O(n)$	127.74
BERT	109M	12	10	$O(n^2)$	103.54
T-LSTM (m=4)	46M	1	10	$O(n)$	820.57
Ours (m=4)	45M	3	10	$O(n)$	83.10
Ours (m=8)	45M	3	10	$O(n)$	**57.40**
BERT	46M	3	60	$O(n^2)$	112.17
XLNet	46M	3	60	$O(n)$	105.64
ALBERT	46M	12	60	$O(n^2)$	71.52
XLNet	116M	12	60	$O(n)$	59.74
BERT	109M	12	60	$O(n^2)$	**44.70**
Ours (m=4)	45M	3	60	$O(n)$	55.70
Ours (m=8)	45M	3	60	$O(n)$	54.60

Table 1: Comparison with state-of-the-art models trained from scratch on WikiText-2 with different settings (number of Transformer layers and training epochs). m is the pruning threshold.

Data. The English language WikiText-2 corpus (Merity et al., 2017) serves as training data. The dataset is split at the sentence level, and sentences longer than 128 after tokenization are discarded (about 0.03% of the original data). The total number of sentences is 68,634, and the average sentence length is 33.4.

Hyperparameters. The tree encoder of our model uses 3-layer Transformers with 768-dimensional embeddings, 3,072-dimensional hidden layer representations, and 12 attention heads. Other models based on the Transformer share the same setting but vary on the number of layers. Training is conducted using Adam optimization with weight decay with a learning rate of 5×10^{-5}. The batch size is set to 8 for m=8 and 32 for m=4, though we also limit the maximum total length for each batch, such that excess sentences are moved to the next batch. The limit is set to 128 for m=8 and 512 for m=4. It takes about 43 hours for 10 epochs of training with $m = 8$ and about 9 hours with m=4, on 8 v100 GPUs.

3.1.2 Results and Discussion

Table 1 presents the results of all models with different parameters. Our model outperforms other models with the same parameter size and number

	emb. $\times$ *hid.* $\times$ *lay.*	training time
Ours (m=4)	$768 \times 3072 \times 3$	7h
-w/o pruning	$12 \times 12 \times 1$	1125h
-w/o pruning*	$768 \times 3072 \times 3$	5×10^7h

Table 2: Training time for one epoch on a single v100 GPU, where *emb.* and *hid.* represent the dimensions of word embeddings and hidden state respectively, and *lay.* is the number of transformer layers. * means proportionally estimated time.

of training epochs. These results suggest that our model architecture utilizes the training data more efficiently. Comparing the different pruning thresholds m=4 and m=8 (last two rows), the two models actually converge to a similar place after 60 epochs, confirming the effectiveness of the pruned tree induction algorithm. We also replace Transformers with Tree-LSTMs as in Jang et al. (2017), denoted as T-LSTM, finding that the perplexity is significantly higher compared to other models.

The best score is from the BERT model with 12 layers at epoch 60. Although our model has a linear time complexity, it is still a sequential encoding model, and hence its training time is not comparable to that of fully parallelizable models. Thus, we do not have results of 12-layer Transformers in Table 1. The experimental results comparing models with the same parameter size suggest that our model may perform even better with further deep layers.

Table 2 shows the training time of our R2D2 with and without pruning. The last row is proportionally estimated by running the small setting ($12 \times 12 \times 1$). It is clear that it is not feasible to run our R2D2 without pruning.

3.2 Unsupervised Constituency Parsing

We next assess to what extent the trees that naturally arise in our model bear similarities with human-specified parse trees.

3.2.1 Setup

Baselines and Evaluation. For comparison, we further include four recent strong models for unsupervised parsing with open source code: BERT masking (Wu et al., 2020), Ordered Neurons (Shen et al., 2019), DIORA (Drozdov et al., 2019) and C-PCFG (Kim et al., 2019a). Following Htut et al. (2018), we train all systems on a training set consisting of raw text, and evaluate and report the results on an annotated test set. As an evaluation metric, we adopt sentence-level unlabeled F_1 computed using the script from Kim et al. (2019a). We compare against the non-binarized gold trees per convention. The best checkpoint for each system is picked based on scores on the validation set.

As our model is a pretrained model based on word-pieces, for a fair comparison, we test all models with two types of input: word level (W) and word-piece level (WP)[3]. To support word-piece level evaluation, we convert gold trees to word-piece level trees by simply breaking each terminal node into a non-terminal node with its word-pieces as terminals, e.g., (NN discrepancy) into (NN (WP disc) (WP ##re) (WP ##pan) (WP ##cy). We set the pruning threshold m to 8 for our tree encoder.

To support a word-level evaluation, since our model uses word-pieces, we force it to not prune or select spans that conflict with word spans during prediction, and then merge word-pieces into words in the final output. However, note that this constraint is only used for word-level prediction.

For training, we use the same hyperparameters as in Section 3.1.1. Our model pretrained on WikiText-2 is finetuned on the training set with the same unsupervised loss objective. For Chinese, we use a subset of Chinese Wikipedia for pretraining, specifically the first 100,000 sentences shorter than 150 characters.

Data. We test our approach on the Penn Treebank (PTB) (Marcus et al., 1993) with the standard splits (2-21 for training, 22 for validation, 23 for test) and the same preprocessing as in recent work (Kim et al., 2019a), where we discard punctuation and lower-case all tokens. To explore the universality of the model across languages, we also run experiments on Chinese Penn Treebank (CTB) 8 (Xue et al., 2005), on which we also remove punctuation. Note that in all settings, the training is conducted entirely on raw unannotated text.

3.2.2 Results and Discussion

Table 3 provides the unlabeled F_1 scores of all systems on the WSJ and CTB test sets. It is clear that all systems perform better than left/right branching and random trees. Word-level C-PCFG (W) performs best on both the WSJ and CTB test sets when measuring against word-level gold standard trees. Our system performs better than ON-LSTM (W), but worse than DIORA (W) and C-PCFG (W). Still,

[3] As DIORA relies on ELMO word embeddings, to support word-piece level inputs, we use BERT word-piece embeddings instead.

Model	cplx	WSJ F_1(M)	WSJ F_1	CTB F_1
Left Branching (W)	$O(n)$	-	8.15	11.28
Right Branching (W)	$O(n)$	-	39.62	27.53
Random Trees (W)	$O(n)$	-	17.76	20.17
BERT-MASK (WP)	$O(n^4)$	-	37.39	33.24
ON-LSTM (W)	$O(n)$	50.0†	47.72	24.73
DIORA (W)	$O(n^3)$	58.9†	51.42	-
C-PCFG (W)	$O(n^3)$	**60.1†**	**54.08**	**49.95**
Ours (WP)	$O(n)$	-	48.11	44.85
DIORA (WP)	$O(n^3)$	-	43.94	-
C-PCFG (WP)	$O(n^3)$	-	49.76	60.34
Ours (WP)	$O(n)$	-	**52.28**	**63.94**

Table 3: Unsupervised parsing results with word (W) or word-piece (WP) as input. Values with † are taken from Kim et al. (2019a). F_1(M) describes the max. score of 4 runs with different random seeds. The F_1 column shows results of our runs with a random seed. The bottom three systems take word-pieces as input, and are also measured against word-piece level golden trees.

	(WP)	WD	NNP	NP	VP	SBAR
WSJ	DIORA	81.65	77.83	71.24	17.30	22.16
WSJ	C-PCFG	74.26	66.44	65.01	23.63	**40.40**
WSJ	Ours	**99.24**	**86.76**	**72.59**	**24.74**	39.81
CTB	C-PCFG	89.34	-	46.74	**39.53**	-
CTB	Ours	**97.16**	-	**61.26**	37.90	-

Table 4: Recall of constituents and words at word-piece level. WD means word.

this is a remarkable result. Note that models such as C-PCFG are specially designed for unsupervised parsing, e.g., adopting 30 nonterminals, 60 preterminals, and a training objective that is well-aligned with unsupervised parsing. In contrast, the objective of our model is that of bi-directional language modeling, and the derived binary trees are merely a by-product of our model that happen to emerge naturally from the model's preference for structures that are conducive to better language modeling.

Another factor is the mismatch between our training and evaluation, where we train our model at the word-piece level, but evaluate against word-level gold trees. For comparison, we thus also considered DIORA (WP), C-PCFG (WP), and our system all trained on word-piece inputs, and evaluated against word-piece level gold trees. The last three lines show the results, with our system achieving the best F_1. As breaking words into word-pieces introduces word boundaries as new spans, while word boundaries are easier to recognize, the overall F_1 score may increase, especially on Chinese.

Analysis. In order to better understand why our model works better when evaluating on word-piece level golden trees, we compute the recall of constituents following Kim et al. (2019b) and Drozdov et al. (2020). Besides standard constituents, we also compare the recall of word-piece chunks and proper noun chunks. Proper noun chunks are extracted by finding adjacent unary nodes with same parent and tag NNP.

Table 4 reports the recall scores for constituents and words on the WSJ and CTB test sets. Our model and DIORA perform better for small semantic units, while C-PCFG better matches larger semantic units such as VP and SBAR. The recall of word chunks (WD) of our system is almost perfect and significantly better than for other algorithms. Please note that all word-piece level models are trained fairly without using any boundary information. Although it is trivial to recognize English word boundaries among word-pieces using rules, this is non-trivial for Chinese. Additionally, the recall of proper noun segments is as well significantly better for our model compared to other algorithms.

3.3 Dependency Tree Compatibility

We compared examples of trees inferred by our model with the corresponding ground truth constituency trees (see Appendix), encountering reasonable structures that are different from the constituent structure posited by the manually defined gold trees. Experimental results of previous work (Drozdov et al., 2020; Kim et al., 2019a) also show significant variance with different random seeds. Thus, we hypothesize that an isomorphy-focused F_1 evaluation with respect to gold constituency trees is insufficient to evaluate how reasonable the induced structures are. In contrast, dependency grammar encodes semantic and syntactic relations directly, and has the best interlingual phrasal cohesion properties (Fox, 2002). Therefore, we introduce dependency compatibility as an additional metric and re-evaluate all system outputs.

3.3.1 Setup

Baselines and Data. As our approach is a word-piece level pretrained model, to enable a fair comparison, we train all models on word-pieces and

Model	WSJ				CTB			
	$\%_{\text{all}}$	$\%_{n\leq 10}$	$\%_{n\leq 20}$	$\%_{n\leq 40}$	$\%_{all}$	$\%_{n\leq 10}$	$\%_{n\leq 20}$	$\%_{n\leq 40}$
BERT-MASK (W)	53.53	77.03	55.46	44.66	48.56	68.89	47.27	36.62
ON-LSTM (W)	61.43†	77.05†	62.99†	55.94†	36.48	58.57	34.08	26.59
DIORA (W)	67.76	78.08	68.80	64.15	—	—	—	—
C-PCFG (W)	**72.74**†	**85.10**†	**74.65**†	**67.19**†	**64.41**	**75.54**	**65.89**	**58.16**
DIORA (WP)	54.73	68.80	55.68	49.22	—	—	—	—
C-PCFG (WP)	67.18	**83.09**	68.20	61.03	62.25	**74.98**	61.04	52.52
Ours (WP)	**69.29**	80.29	**70.29**	**64.79**	**64.74**	74.42	**63.86**	**59.20**

Table 5: Compatibility with dependency trees. (W) denotes word level inputs, (WP) refers to word-piece level inputs. $\%_{\text{all}}$ denotes the accuracy on all test sentences, while $\%_{n\leq x}$ is the accuracy on sentences of up to x words. Values with † are evaluated with predicted trees from Kim et al. (2019a)

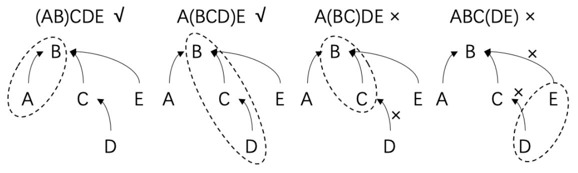


Figure 6: Examples of compatible and incompatible spans.

learn models with the same settings as in the original papers. Evaluation at the word-piece level reveals the model's ability to learn structure from a smaller granularity. In this section, we keep the word-level gold trees unchanged and invoke Stanford CoreNLP (Manning et al., 2014) to convert the WSJ and CTB into dependency trees.

Evaluation. Our metric is based on the notion of quantifying the compatibility of a tree by counting how many spans comply with dependency relations in the gold dependency tree. Specifically, as illustrated in Figure 6, a span is deemed compatible with the ground truth if and only if this span forms an independent subtree.

Formally, given a gold dependency tree $\mathcal{D}$, we denote as $\mathcal{S}(\mathcal{D})$ the raw token sequence for $\mathcal{D}$. Considering predicting a binary tree for word-level input, predicted spans in the binary tree are denoted as $\mathcal{Z}$. For any span $z \in \mathcal{Z}$, the subgraph of $\mathcal{D}$ including nodes in z and directional edges between them is referred to as $\mathcal{G}_z$. $\mathcal{O}(\mathcal{G}_z)$ is defined as the set of nodes with parent nodes not in $\mathcal{G}_z$ and $\mathcal{I}(\mathcal{G}_z)$ denotes the set of nodes whose child nodes are not in $\mathcal{G}_z$. Thus, $|\mathcal{O}(\mathcal{G}_z)|$ and $|\mathcal{I}(\mathcal{G}_z)|$ are the out-degree and in-degree of the subgraph $\mathcal{G}_z$. Let $\mathrm{I}(z)$ denote whether z is valid, defined as

$$I(z)\begin{cases}1, & |\mathcal{O}(\mathcal{G}_z)| = 1 \text{ and } \mathcal{I}(\mathcal{G}_z) \subseteq \mathcal{O}(\mathcal{G}_z) \\ 0, & \text{otherwise.}\end{cases} \quad (9)$$

For binary tree spans for word-piece level input, if z breaks word-piece spans, then $\mathrm{I}(z) = 0$. Otherwise, word-pieces are merged to words and the word-level logic is followed. Specifically, to make the results at the word and word-piece levels comparable, $\mathrm{I}(z)$ is forced to be zero if z only covers a single word. The final compatibility for $\mathcal{Z}$ is $\dfrac{\sum_{z\in\mathcal{Z}} \mathrm{I}(z)}{|\mathcal{S}(\mathcal{D})| - 1}$.

3.3.2 Results and Discussion

Table 5 lists system results on the WSJ and CTB test sets. $\%_{\text{all}}$ refers to the accuracy on all test sentences, while $\%_{n\leq x}$ is the accuracy on sentences with up to x words. It is clear that the smaller granularity at the word-piece level makes this task harder. Our model performs better than other systems at the word-piece level on both English and Chinese and even outperforms the baselines in many cases at the word level. It is worth noting that the result is evaluated on the same binary predicted trees as we use for unsupervised constituency parsing, yet our model outperforms baselines that perform better in Table 3. One possible interpretation is that our approach learns to prefer structures different from human-defined phrase structure grammar but self-consistent and compatible with a tree structure. To further understand the strengths and weaknesses of each baseline, we analyzed the compatibility of different sentence length ranges. Interestingly, we find that our approach performs better on long sentences compared with C-PCFG at the word-piece level. This shows that a bidirectional language modeling objective can learn to induce accurate structures even on very long sentences, on which custom-tailored methods may not work as well.

4 Related Work

Pre-trained models. Pre-trained models have achieved significant success across numerous tasks. ELMo (Peters et al., 2018), pretrained on bidirectional language modeling based on bi-LSTMs, was the first model to show significant improvements across many downstream tasks. GPT (Radford et al., 2018) replaces bi-LSTMs with a Transformer (Vaswani et al., 2017). As the global attention mechanism may reveal contextual information, it uses a left-to-right Transformer to predict the next word given the previous context. BERT (Devlin et al., 2019) proposes masked language modeling (MLM) to enable bidirectional modeling while avoiding contextual information leakage by directly masking part of input tokens. As masking input tokens results in missing semantics, XLNET (Yang et al., 2019) proposes permuted language modeling (PLM), where all bi-directional tokens are visible when predicting masked tokens. However, all aforementioned Transformer based models do not naturally capture positional information on their own and do not have explicit interpretable structural information, which is an essential feature of natural language. To alleviate the above shortcomings, we extend pre-training and the Transformer model to structural language models.

Representation with structures. In the line of work on learning a sentence representation with structures, Socher et al. (2011) proposed the first neural network model applying recursive autoencoders to learn sentence representations, but their approach constructs trees in a greedy way, and it is still unclear how autoencoders can perform against large pre-trained models (e.g., BERT). Yogatama et al. (2017) jointly train their shift-reduce parser and sentence embedding components. As their parser is not differentiable, they have to resort to reinforcement training, but the learned structures collapse to trivial left/right branching trees. The work of URNNG (Kim et al., 2019b) applies variational inference over latent trees to perform unsupervised optimization of the RNNG (Dyer et al., 2016), an RNN model that estimates a joint distribution over sentences and trees based on shift-reduce operations. Maillard et al. (2017) propose an alternative approach, based on CKY parsing. The algorithm is made differentiable by using a soft-gating approach, which approximates discrete candidate selection by a probabilistic mixture of the constituents available in a given cell of the chart. This makes it possible to train with backpropagation. However, their model runs in $O(n^3)$ and they use Tree-LSTMs.

5 Conclusion and Outlook

In this paper, we have proposed an efficient CKY-based recursive Transformer to directly model hierarchical structure in linguistic utterances. We have ascertained the effectiveness of our approach on language modeling and unsupervised parsing. With the help of our efficient linear pruned tree induction algorithm, our model quickly learns interpretable tree structures without any syntactic supervision, which yet prove highly compatible with human-annotated trees. As future work, we are investigating pre-training our model on billion word corpora as done for BERT, and fine-tuning our model on downstream tasks.

Acknowledgements

We thank Liqian Sun, the wife of Xiang Hu, for taking care of their newborn baby during critical phases, which enabled Xiang to polish the work and perform experiments.

References

Yoshua Bengio. 2009. *Learning deep architectures for AI*. Now Publishers Inc.

Noam Chomsky. 1956. Three models for the description of language. *IRE Trans. Inf. Theory*, 2(3):113–124.

Noam Chomsky. 2014. *Aspects of the Theory of Syntax*, volume 11. MIT press.

John Cocke. 1969. *Programming Languages and Their Compilers: Preliminary Notes*. New York University, USA.

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

Andrew Drozdov, Subendhu Rongali, Yi-Pei Chen, Tim O'Gorman, Mohit Iyyer, and Andrew McCallum. 2020. Unsupervised parsing with S-DIORA: Single tree encoding for deep inside-outside recursive autoencoders. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4832–4845, Online. Association for Computational Linguistics.

Andrew Drozdov, Patrick Verga, Mohit Yadav, Mohit Iyyer, and Andrew McCallum. 2019. Unsupervised latent tree induction with deep inside-outside recursive auto-encoders. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1129–1141, Minneapolis, Minnesota. Association for Computational Linguistics.

Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A. Smith. 2016. Recurrent neural network grammars. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 199–209, San Diego, California. Association for Computational Linguistics.

Heidi Fox. 2002. Phrasal cohesion and statistical machine translation. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP 2002)*, pages 304–3111. Association for Computational Linguistics.

Phu Mon Htut, Kyunghyun Cho, and Samuel Bowman. 2018. Grammar induction with neural language models: An unusual replication. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4998–5003, Brussels, Belgium. Association for Computational Linguistics.

Eric Jang, Shixiang Gu, and Ben Poole. 2017. Categorical reparameterization with gumbel-softmax. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

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

Yoon Kim, Chris Dyer, and Alexander Rush. 2019a. Compound probabilistic context-free grammars for grammar induction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2369–2385, Florence, Italy. Association for Computational Linguistics.

Yoon Kim, Alexander Rush, Lei Yu, Adhiguna Kuncoro, Chris Dyer, and Gábor Melis. 2019b. Unsupervised recurrent neural network grammars. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1105–1117, Minneapolis, Minnesota. Association for Computational Linguistics.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. ALBERT: A lite BERT for self-supervised learning of language representations. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Phong Le and Willem Zuidema. 2015. The forest convolutional network: Compositional distributional semantics with a neural chart and without binarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1155–1164, Lisbon, Portugal. Association for Computational Linguistics.

Jean Maillard, Stephen Clark, and Dani Yogatama. 2017. Jointly learning sentence embeddings and syntax with unsupervised tree-lstms. *CoRR*, abs/1705.09189.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60, Baltimore, Maryland. Association for Computational Linguistics.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC'06)*, Genoa, Italy. European Language Resources Association (ELRA).

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2017. Pointer sentinel mixture models. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.

Ruslan Salakhutdinov. 2014. Deep learning. In *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, New York, NY, USA - August 24 - 27, 2014*, page 1973. ACM.

Julian Salazar, Davis Liang, Toan Q. Nguyen, and Katrin Kirchhoff. 2020. Masked language model scoring. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2699–2712, Online. Association for Computational Linguistics.

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

Richard Socher, Jeffrey Pennington, Eric H. Huang, Andrew Y. Ng, and Christopher D. Manning. 2011. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 151–161, Edinburgh, Scotland, UK. Association for Computational Linguistics.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1556–1566, Beijing, China. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.

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, Online. Association for Computational Linguistics.

Naiwen Xue, Fei Xia, Fu-dong Chiou, and Marta Palmer. 2005. The penn chinese treebank: Phrase structure annotation of a large corpus. *Nat. Lang. Eng.*, 11(2):207–238.

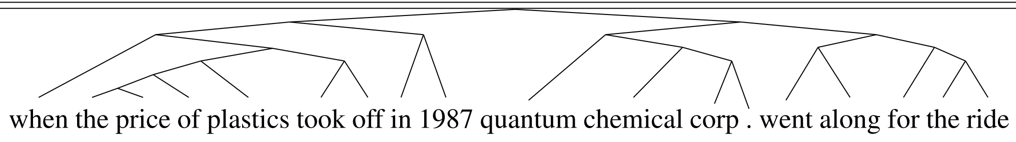
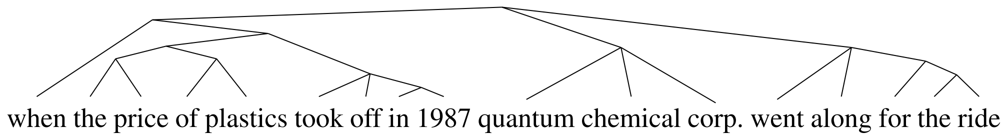
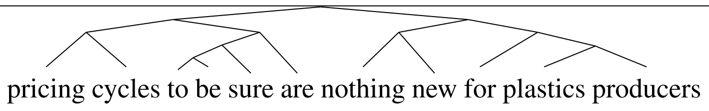
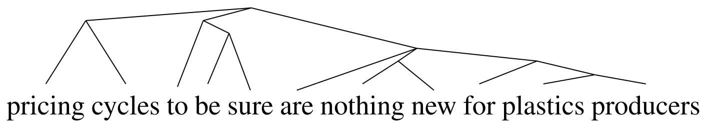
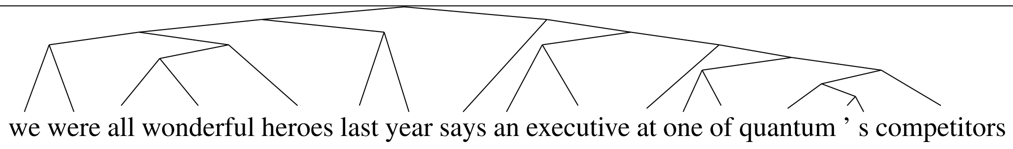
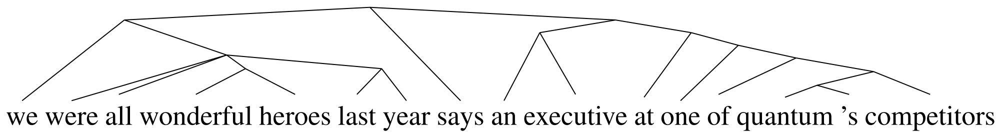
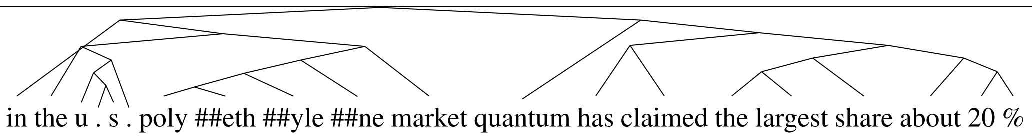
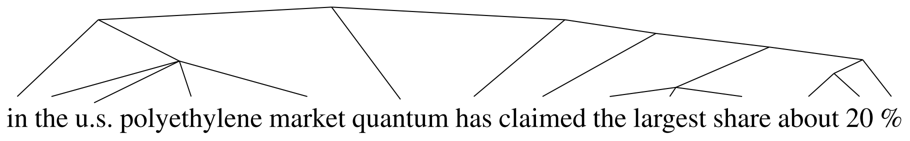
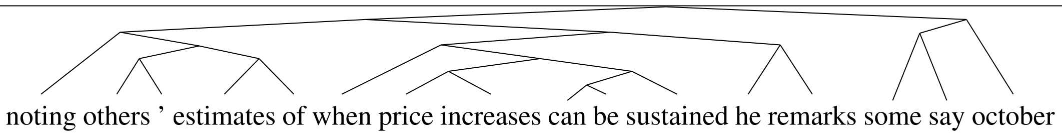
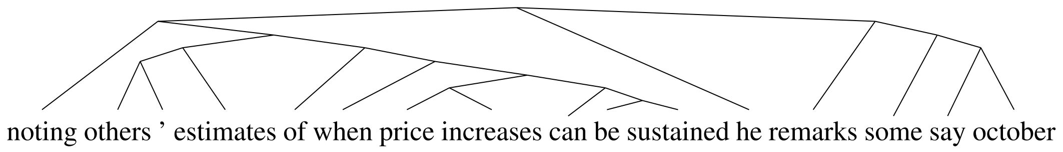
Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 5754–5764.

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

Daniel H Younger. 1967. Recognition and parsing of context-free languages in time n3. *Information and control*, 10(2):189–208.

Xiao-Dan Zhu, Parinaz Sobhani, and Hongyu Guo. 2015. Long short-term memory over recursive structures. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 1604–1612. JMLR.org.

A Appendix: Tree Examples

System	Tree
R2D2	 when the price of plastics took off in 1987 quantum chemical corp . went along for the ride
GOLD	 when the price of plastics took off in 1987 quantum chemical corp. went along for the ride
R2D2	 pricing cycles to be sure are nothing new for plastics producers
GOLD	 pricing cycles to be sure are nothing new for plastics producers
R2D2	 we were all wonderful heroes last year says an executive at one of quantum ' s competitors
GOLD	 we were all wonderful heroes last year says an executive at one of quantum 's competitors
R2D2	 in the u . s . poly ##eth ##yle ##ne market quantum has claimed the largest share about 20 %
GOLD	 in the u.s. polyethylene market quantum has claimed the largest share about 20 %
R2D2	 noting others ' estimates of when price increases can be sustained he remarks some say october
GOLD	 noting others ' estimates of when price increases can be sustained he remarks some say october