STRUCTFORMER: JOINT UNSUPERVISED INDUCTION OF DEPENDENCY AND CONSTITUENCY STRUCTURE FROM MASKED LANGUAGE MODELING

Anonymous authors
Paper under double-blind review

ABSTRACT

There are two major classes of natural language grammars — the dependency grammar that models one-to-one correspondences between words and the constituency grammar that models the assembly of one or several corresponded words. While previous unsupervised parsing methods mostly focus on only inducing one class of grammars, we introduce a novel model, StructFormer, that can induce dependency and constituency structure at the same time. To achieve this, we propose a new self-attention mechanism with novel hierarchical and dependency constraints. Experimental results show that our model can achieve strong results on unsupervised constituency parsing, unsupervised dependency parsing and masked language modeling at the same time. To the best of our knowledge, StructFormer is the first neural network model that can directly induce both dependency and constituency structures from raw data in an end-to-end unsupervised fashion.

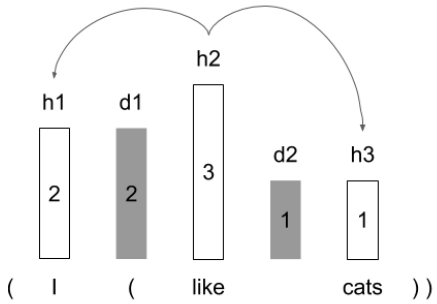
1 INTRODUCTION

Human languages have a rich latent structure. This structure is multifaceted, with the two major classes of grammar being dependency and constituency structures. While there have been an exciting breath of recent work that are targeted at learning this structure in a data-driven unsupervised fashion, they have been primarily focused on a single aspect of this latent structure, e.g., constituency parsing alone (Shen et al., 2018c; Wang et al., 2019). To the best of our knowledge, there have been no prior model or work that is able to induce, in an unsupervised fashion, more than one grammar structure at once.

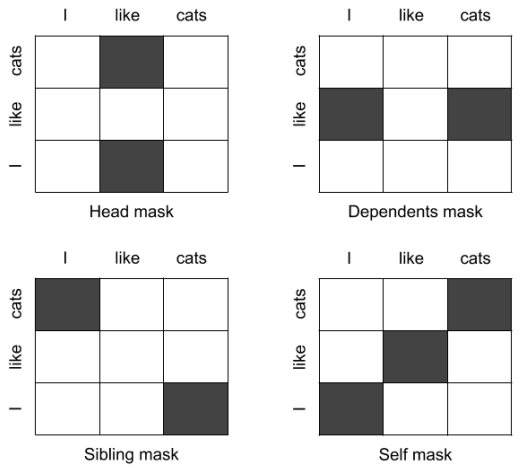
The core principle behind recent neural methods that induce structure from data is simple - provide an inductive bias that is conducive for structure to emerge as a byproduct of some self-supervised training, e.g., language modeling. To this end, a wide range of models have been proposed that are able to successfully learn grammar structures (Shen et al., 2018a;c; Wang et al., 2019). However, most of these works focus on learning constituency structures alone.

In this paper, we make two important technical contributions. First, we introduce the first neural model that is able to induce dependency structures from raw data in an end-to-end unsupervised fashion. Most of existing approaches induce dependency structures from other syntactic information like gold POS tags (Klein & Manning, 2004; Cohen & Smith, 2009; Jiang et al., 2016). Previous works, that have trained from words alone, often requires additional information, like pre-trained word clustering (Spitkovsky et al., 2011), pre-trained word embedding (He et al., 2018), acoustic cues (Pate & Goldwater, 2013), or annotated data from related languages (Cohen et al., 2011). Second, we introduce the first neural model that is able to induce **both** dependency structure and constituency structure at the same time. Specifically, our approach aims to unify latent structure induction of different types of grammar within the same framework.

We introduce a new inductive bias for Transformer models that enables the self-attention mechanism to induce a directed dependency graph in a fully unsupervised manner. To avoid the need of grammar labels during training, we introduce a Joint Distance Parser (JDP). The key idea behind the proposed JDP is that it predicts a sequence of Syntactic Distances $\mathbf{d}$ (Shen et al., 2018b) and a sequence of



(a) An example of Syntactic Distances $\mathbf{d}$ (grey bars) and Syntactic Heights $\mathbf{h}$ (white bars). Following the Algorithm.1, we condition on $\mathbf{d}$ to recursively split the input sentence into smaller constituents `(I (like cats))`. Then we assign the token with the largest $\mathbf{h}$ inside each constituent as the head of the constituent. In this example, `like` is the head of constituent `(like cats)` and the sentence `(I like cats)`.



(b) Four base masks given by dependency relations. The head mask allows each token to attend on its head. The dependent mask allows each token to attend on its dependents. The sibling mask allows each token to attend on other dependents of its head. The self mask allows each token to attend on itself. Each attention head will receive a different weighted sum of these masks as its attention mask.

Figure 1: An example of our joint parsing and masking mechanism. The parsing network first predicts the syntactic distance $\mathbf{d}$ and syntactic height $\mathbf{h}$ to induce the latent structure of the input sentence `I like cats`. Then the dependency masking mechanism generates four different base masks according to $\mathbf{d}$ and $\mathbf{h}$.

Syntactic Heights $\mathbf{h}$ (Luo et al., 2019) to build dependency and constituency trees at the same time. We provide an illustration in Fig.1a.

Based on the syntactic distances ($\mathbf{d}$) and syntactic heights ($\mathbf{h}$), we provide a differentiable masking mechanism for multi-head self-attention layers. Concretely, each attention head can selectively attend on the token itself, its head, its dependents, or its siblings in the predicted dependency structure, through a weighted sum of masks shown in Fig.1b. Thus, the parser can be trained in an unsupervised manner as a component of a masked language model.

Incorporating the new parsing mechanism, the differentiable mask, and the Transformer architecture, we introduce a new model named StructFormer. The proposed model can perform unsupervised dependency and constituency parsing at the same time, and can leverage the parsing results to achieve strong performance on masked language model tasks.

2 Related Work

Previous work on unsupervised dependency parsing is primarily based on the dependency model with valence (DMV) (Klein & Manning, 2004) and its extension (Daumé III, 2009; Gillenwater et al., 2010). To effectively learn the DMV model for better parsing accuracy, a variety of inductive biases and handcrafted features, such as correlations between parameters of grammar rules involving different part-of-speech (POS) tags, have been proposed to incorporate prior information into learning. The most recent progress is the neural DMV model (Jiang et al., 2016), which uses a neural network model to predict the grammar rule probabilities based on the distributed representation of POS tags. However, most previous unsupervised dependency parsing algorithms require the gold POS tags as input, which are labeled by humans and can be potentially difficult (or prohibitively expensive) to obtain for large corpora. Spitkovsky et al. (2011) proposed to overcome this problem with unsupervised word clustering that can dynamically assign tags to each word considering its context.

Unsupervised constituency parsing has recently received more attention. PRPN (Shen et al., 2018a) and ON-LSTM (Shen et al., 2018c) induce tree structure by introducing an inductive bias to recurrent neural networks. PRPN proposes a parsing network to compute the syntactic distance of all word pairs, and a reading network utilizes the syntactic structure to attend to relevant memories. ON-LSTM allows hidden neurons to learn long-term or short-term information by a novel gating mechanism and activation function. In URNNG (Kim et al., 2019b), amortized variational inference was applied between a recurrent neural network grammar (RNNG) (Dyer et al., 2016) decoder and a tree structure inference network, which encourages the decoder to generate reasonable tree structures. DIORA (Drozdov et al., 2019) proposed using inside-outside dynamic programming to compose latent representations from all possible binary trees. The representations of inside and outside passes from the same sentences are optimized to be close to each other. Compound PCFG (Kim et al., 2019a) achieves grammar induction by maximizing the marginal likelihood of the sentences which are generated by a probabilistic context-free grammar (PCFG) in a corpus. Tree Transformer (Wang et al., 2019) adds extra locality constraints to the Transformer encoder's self-attention to encourage the attention heads to follow a tree structure such that each token can only attend on nearby neighbors in lower layers and gradually extend the attention field to further tokens when climbing to higher layers.

Though large scale pre-trained models have dominated most natural language processing tasks, some recent work indicates that neural network models can see accurarcy gains by leveraging syntactic information rather then ignoring it (Marcheggiani & Titov, 2017; Strubell et al., 2018). Strubell et al. (2018) introduces syntactically-informed self-attention that force one attention head to attend on the syntactic governor of input token. Omote et al. (2019) and Deguchi et al. (2019) argue that dependency-informed self-attention can improve Transformer's performance on machine translation. Kuncoro et al. (2020) shows that syntactic biases help large scale pre-trained models, like BERT, to achieve better language understanding.

3 Joint Dependency and Constituency Parsing

In this section, we will introduce a new parsing algorithm, Joint Distance Parser (JDP), that can jointly produce dependency and constituency trees at the same time. The JDP predicts two sets of values: Syntactic Distance $\mathbf{d}$ that can be used to reconstruct the constituency structure, and Syntactic Height $\mathbf{h}$ that indicate the head of each constituent.

Syntactic Distance is proposed in Shen et al. (2018b) for constituency parsing. Given a constituency tree $\mathbf{T}$ that contains a set of leaves $(w_0, ..., w_n)$, $\tilde{d}_i$ is the height of the lowest common ancestor between consecutive words x_i and x_{i+1}. Syntactic distances $\mathbf{d} = (d_0, ..., d_{n-1})$ are defined as a sequence of $n-1$ real scalars that share the same rank as $(\tilde{d}_0, ..., \tilde{d}_{n-1})$.

In other words, each syntactic distance x_i is associated with a split point $(i, i+1)$ in the sentence and specify the order in which the sentence will be split into smaller components. Thus, a sequence of syntactic distances $\mathbf{d}$ can unambiguously rebuild the unlabeled constituency tree through a deterministic parsing algorithm (Shen et al., 2018b).

Syntactic Height is proposed in Luo et al. (2019), where the syntactic height is used to capture the distance to the root node in a dependency tree. A word with high syntactic height means it is close to the root node. In this paper, to match the definition of syntactic distance, we redefine syntactic height as:

Definition 3.1. Let $\mathbf{D}$ be a dependency tree for sentence $(w_0, ..., w_n)$. The height of a token w_i in $\mathbf{D}$ is $\tilde{h}_i$. The syntactic heights of $\mathbf{D}$ can be any sequence of n real scalars $\mathbf{h} = (h_0, ..., h_n)$ that share the same rank as $(\tilde{h}_0, ..., \tilde{h}_{n-1})$.

Although the syntactic height is defined based on the dependency structure, it is unable to rebuild the original dependency structure just by itself since there is no information about whether a token should be attached to the left side or the right side. However, given a constituent, we can identify the head of this constituent by finding the word with the largest syntactic height, because one token can only be attached to the token with the largest syntactic height in the constituent. Thus, we can use syntactic height to convert a constituency tree to a dependency tree, or use both syntactic distance

Algorithm 1 The joint dependency and constituency parsing algorithm. The inputs are a sequence of words $\mathbf{w}$, syntactic distances $\mathbf{d}$, syntactic heights $\mathbf{h}$, and a positive real number gap.

function BUILDTREE($\mathbf{w}, \mathbf{d}, \mathbf{h}$, gap)
  if $\mathbf{w} = [w_i]$ **then**
    node $\Leftarrow w_i$, head $\Leftarrow w_i$, height $\Leftarrow h_i$
  else
    $d_{\max} \Leftarrow \max(\mathbf{d})$, s $\Leftarrow 0$
    for $d_i \in \mathbf{d}$ **do**
      if $d_i > d_{\max} - \text{gap}$ or $i = \text{len}(\mathbf{w})$ **then**
        $\text{child}_i, \text{head}_i, \text{height}_i \Leftarrow$ BuildTree($\mathbf{d}_{s:i}, \mathbf{w}_{s:i}, \mathbf{h}_{s:i}$)
        $s = i + 1$
      end if
    end for
    node $\Leftarrow [\text{child}_1, ..., \text{child}_n]$, head $\Leftarrow \text{head}_k$, height $\Leftarrow \text{height}_k$
    $k \Leftarrow \text{argmax}_i(\text{height}_i)$
    for $\text{head}_i \in \{\text{head}_i\}/\text{head}_k$ **do**
      AddDependency($\text{head}_i \leftarrow \text{head}_k$)
    end for
  end if
  return node, head, height
end function

and syntactic height to jointly do constituency and dependency parsing at the same time. The joint parsing algorithm is described in Algorithm 1. The basic idea is to build the constituency structure in a top-down recursive manner and identify the head of each constituent through the syntactic height.

4 STRUCTFORMER

In this section, we present the StructFormer model. Fig.2a shows the architecture of StructFormer, which includes a parser network and a Transformer module. The parser network predicts $\mathbf{d}$ and $\mathbf{h}$, then passes them to a set of differentiable functions to generate attention masks that reflect the latent dependency relations. The Transformer module takes these masks and the sentence as input and computes a contextual embedding for each position. The StructFormer can be trained in an end-to-end fashion on a Masked Language Model task. In this setting, the gradient back propagates through the differentiable attention mask into the parser.

4.1 PARSING NETWORK

As shown in Fig.2b, the parsing network takes word embeddings as input and feeds them into several residual convolution layers:

$$s_{l,i} = \text{Conv}(s_{l-1,i-W}, s_{l-1,i-W+1}, ..., s_{l-1,i+W}) + s_{l-1,i} \tag{1}$$

where $s_{l,i}$ is the output of l-th layer at i-th position, $s_{0,i}$ is the input embedding of token w_i, and $2W + 1$ is the convolution kernel size.

Given the output of the convolution stack $s_{N,i}$, we parameterize the syntactic distance $\mathbf{d}$ as:

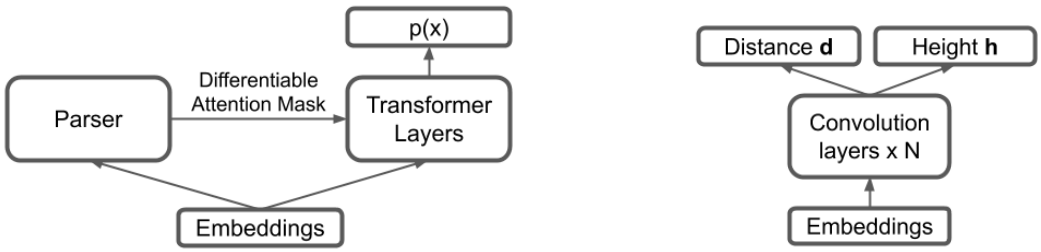
$$d_i = \mathbf{W}_1^d \tanh\left(\mathbf{W}_2^d \begin{bmatrix} s_{N,i} \\ s_{N,i+1} \end{bmatrix} + b_2^d\right) + b_1^d \tag{2}$$

where d_i is the contextualized distance for the i-th split point between token w_i and w_{i+1}. The syntactic height $\mathbf{h}$ is parameterized in a similar way:

$$h_i = \mathbf{W}_1^h \tanh\left(\mathbf{W}_2^h s_{N,i} + b_2^h\right) + b_1^h \tag{3}$$

4.2 DEPENDENCY MASKING

Given $\mathbf{d}$ and $\mathbf{h}$, we now explain how to compute a soft attention mask that represents the dependency relations. We first parameterize the probability that the token w_j is in the smallest constituent $\mathbf{C}(i)$



(a) Model Architecture

(b) Parsing Network

Figure 2: The Architecture of StructFormer. The parser takes shared word embeddings as input, outputs syntactic distances $\mathbf{d}$, syntactic heights $\mathbf{h}$, and self-attention masks that reflects the dependency relations between tokens. The transformer layers take word embeddings and self-attention masks as input, output the probability distribution for missing tokens.

that contains both w_i and its head $w_{\text{head}(i)}$:

$$\beta_{i,k} = \sigma((h_i - d_k)/\tau + b) \tag{4}$$

$$p_j^{\mathbf{C}(i)} = \begin{cases} \prod_{j \leq k < i} \beta_{i,k} & j < i \\ 1, & j = i \\ \prod_{i \leq k < j} \beta_{i,k} & j > i \end{cases} \tag{5}$$

where σ is the sigmoid function, τ is a temperature parameter, b is the bias, $\beta_{i,k}$ is the probability that w_i is higher than the boundary between w_k and w_{k+1}, and $p_{i,j}^{\text{ct}}$ is the probability that w_i is higher then all boundary between w_i and w_j.

We then compute the probability that w_j is a head of w_i. Since $\mathbf{C}$ is the smallest constituent that contains both w_i and $w_{\text{head}(i)}$, $w_{\text{head}(i)}$ should be the head of this constituent. Otherwise, there will be a smaller constituent that uses $w_{\text{head}(i)}$ as a head and contains w_i as well. To identify the head $\text{H}(i)$ of $\mathbf{C}(i)$, we use softmax to compute the distribution of maximum syntactic height:

$$\alpha_{i,j} = \sigma\left((h_j - h_i)/\tau + b\right) \tag{6}$$

$$p(i \leftarrow j) = \begin{cases} \frac{\alpha_{i,j} p_j^{\mathbf{C}(i)} \exp(h_j/\tau)}{\sum_{k \neq i} \alpha_{i,k} p_k^{\mathbf{C}(i)} \exp(h_k/\tau)} & j \neq i \\ 0 & j = i \end{cases} \tag{7}$$

The probability that w_j is a dependent of w_i is:

$$p(j \rightarrow i) = p(i \leftarrow j) \tag{8}$$

And the probability that w_i and w_j share the same head is:

$$p_j^{\mathbf{S}(i)} = \sum_{k \neq i, k \neq j} p(i \leftarrow k) p(i \leftarrow k) \tag{9}$$

Since we only allow words that have dependency or sibling relations to attend on each other, the attention mask $m^{l,k}$ for head k of transformer layer l is formulated as:

$$m_{i,j}^{l,k} = w_{l,k}^{\text{head}} p(i \leftarrow j) + w_{l,k}^{\text{dep}} p(i \rightarrow j) + w_{l,k}^{\text{sib}} p_j^{\mathbf{S}(i)} + w_{l,k}^{\text{eye}} \mathbf{I}_{ij} \tag{10}$$

where $w_{l,k}^{\text{head}}$, $w_{l,k}^{\text{dep}}$, and $w_{l,k}^{\text{sib}}$ are weights for the respective relations mask, and $w_{l,k}^{\text{eye}}$ is the weight for the identity matrix I that allows the token to attend on itself.

5 Experiments

To evaluate the performance of StructFormer, we train it as a Masked Language Model on BLLIP datasets (Hu et al., 2020), then evaluate the trained model as a unsupervised constituency and dependency parser on WSJ testset.

Methods	UF1
RANDOM	21.6
LBRANCH	9.0
RBRANCH	39.8
Pre-trained LMs	
BERT-large*	34.2
GPT2*	37.1
XLNet-base*	40.1
Unsupervised Parsing Models	
ON-LSTM (Shen et al., 2018c)	47.7
Tree-T (Wang et al., 2019)	49.5
URNNG (Kim et al., 2019b)	52.4
C-PCFG (Kim et al., 2019a)	55.2
StructFormer	50.2

(a) Constituency Parsing Results. * results are from Kim et al. (2020).

Methods	UAS
Inducing from Raw Data	
LBRANCH/RHEAD	33.3
RANDOM	3.7
RBRANCH/LHEAD	4.7
StructFormer	41.0
From Gold POS Tags	
DMV* (Klein & Manning, 2004)	39.4
Shared LN* (Cohen & Smith, 2009)	41.4
From Gold POS Tags, and other informations	
Neural E-DMV* (Jiang et al., 2016)	52.5
TSG-DMV* (Blunsom & Cohn, 2010)	53.1
UR-A E-DMV* (Tu & Honavar, 2012)	57.0

(b) Dependency Parsing Results on WSJ testset. * models use gold POS tags and other informations as input, while StructFormer directly takes raw data as input. Hence, their results is not comparable with our model.

Table 1: The unsupervised parsing performance of different models.

Model	**XS**	**SM**	**MD**
Transformer	93.90	19.92	14.31
StructFormer	57.28	18.70	13.70

Table 2: Masked Language Model perplexities on BLLIP datasets of different models.

Our implementation of StructFormer is close to the original Transformer encoder (Vaswani et al., 2017), except that we put the layer normalization in front of each layer, similar to the T5 model (Raffel et al., 2019). We found that this modification allows the model to converges faster. For all experiments, we set the number of layers $L = 8$, the embedding size and hidden size to be $d_{model} = 512$, the number of self-attention heads $h = 8$, the feed-forward size $d_{ff} = 2048$, dropout rate as 0.1, and the number of convolution layers in the parsing network as $L_p = 3$.

5.1 Masked Language Model

BLLIP is a large Penn Treebank-style parsed corpus of approximately 24 million sentences. We train and evaluate StructFormer on three splits of BLLIP: BLLIP-XS (40k sentences, 1M tokens), BLLIP-SM (200K sentences, 5M tokens), and BLLIP-MD (600K sentences, 14M tokens). They are obtained by randomly sampling sections from BLLIP 1987-89 Corpus Release 1. All models are tested on a shared held-out test set (20k sentences, 500k tokens). Following the settings provided in (Hu et al., 2020), we use subword-level vocabulary extracted from the GPT-2 pre-trained model rather than the BLLIP training corpora.

5.2 Unsupervised Constituency Parsing

The unsupervised constituency parsing task compares the latent tree structure induced by the model with those annotated by human experts. We use the Alogrithm 1 to predict the constituency trees from $\mathbf{d}$ and $\mathbf{h}$ predicted by StructFormer. Following the experiment settings proposed in Kim et al. (2019a), we use corpus-level Unlabeled F1 score to evaluate our model on WSJ test set, which is the section 23 of Wall Street Journal corpus. The WSJ test set contains 2416 human expert labeled sentences. Punctuation is ignored during the evaluation. For both constituency parsing and dependency parsing, we set $\text{gap} = 0.1$.

Methods	MLM	Constituency	Stanford Dependency		Conll Dependency	
	PPL	UF1	UAS	UUAS	UAS	UUAS
eye+deps+head+sibling	18.70	48.7	39.0	51.6	32.9	50.5
eye+deps+head	19.10	50.2	40.0	53.0	42.9	56.7
eye+deps	20.68	41.89	8.0	29.4	10.5	33.7

Table 3: The performance of StructFormer with different combinations of attention masks.

Dataset	Size	MLM	Constituency	Stanford Dependency		Conll Dependency	
	#tokens	PPL	UF1	UAS	UUAS	UAS	UUAS
BLLIP-XS	1M	56.11	45.26	23.8	42.1	31.7	48.1
BLLIP-SM	5M	19.10	50.2	40.0	53.0	42.9	56.7
BLLIP-MD	14M	13.70	49.5	41.0	53.9	38.9	54.6

Table 4: The performance of StructFormer on different datasets.

5.3 Unsupervised Dependency Parsing

The unsupervised dependency parsing evaluation compares the induced dependency relations with those in the gold dependency tree. The most common metric is Unlabeled Attachment Score, which measures the percentage that a token is correctly attached to its head in the reference tree. Another widely used metric for unsupervised dependency parsing measures the percentage that the reference undirected connections are recovered by the induced tree. For the WSJ dataset, reference dependency trees are converted from its human annotated constituency trees. However, there are two different sets of rules for the conversion: the Stanford dependencies and the CoNLL dependencies. The Stanford dependencies are used as reference dependencies in previous unsupervised parsing papers. However, we noticed that our model often output dependency structures that are closer to the CoNLL dependencies. Therefore, we report UAS and UUAS for both Stanford and CoNLL dependencies. Following the setting of previous papers (Jiang et al., 2016), we ignored the punctuation during evaluation.

5.4 Experimental Results

The masked language model results are shown in Table 2. StructFormer consistently outperform our Transformer baseline. This result aligns with previous observations that linguistic informed self-attention can help Transformers achieve stronger performance. We also observe that StructFormer converges much faster than the standard Transformer model.

Table 1a and Table 1b shows our model achieves competitive performance on both dependency parsing and constituency parsing. URNNG (Kim et al., 2019b) and C-PCFG (Kim et al., 2019a) are more dedicated unsupervised parsing algorithm, with their strong linguistic constraints, like a finite set of production rules. They both tuned their hyperparameters on the parsing performance of validation set. It worth noting that all baseline dependency parsing models induce structure from golden POS tags and other linguistic features which are **not** directly comparable with our approach. Since POS tags are known to contain significant amounts of information for infering the dependency structure (McDonald et al., 2011). Directly inducing dependency grammar from raw data, like StructFormer, can be a more difficult task. Structformer achieves 41 F1 on unsupervised dependency parsing, a very competitive score on just raw data.

Since our model uses a mixture of relation masks for each self-attention head, we also studied how different combinations of relation masks effect the performance of our model. Table 3 shows that the model can achieve the best MLM perplexity while using all four masks. The model can achieve the best parsing performance when only using the eye mask, dependents mask, and the head mask. Our hypothesis is that sibling attention are helpful for predicting the missing token. Because it is easier to infer the missing component while the head and sibling information can be directly attended, instead of only have the head information. But when it comes to latent structure induction, the sibling mask can lead the model to assign the incorrect head to the constituent. For example, given

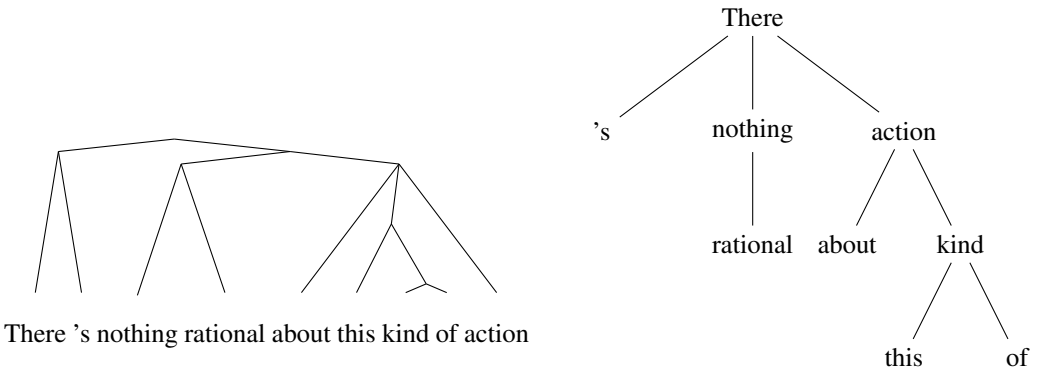


Figure 3: Parsing example from StructFormer

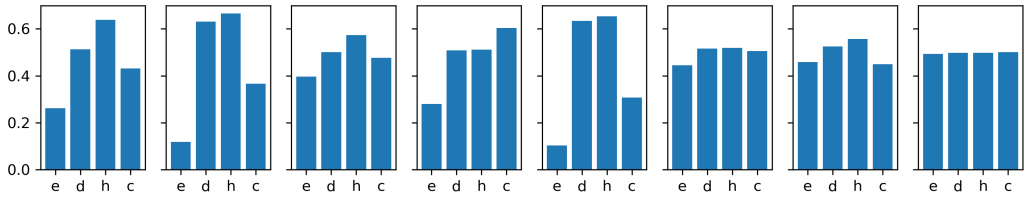


Figure 4: Self-attention mask weights for 8 heads in the first layer of StructFormer. `e` represents the identity matrix **I**. `d` represents the dependency matrix. `h` represents the head matrix. `s` represents the sibling matrix.

the constituent `(a b c)`, the model can choose any token as the head of constituent, and all three tokens will always been fully connected. If `a` is the head, without the sibling mask `c` can only attend to `a`, but with the sibling mask it can also attend on `b`, and vice versa. Thus, inside each constituent, the choice of head can be arbitrary. Figure 4 shows that the model learns to assign different mixture relations mask to different attention heads.

Table 4 shows the performance of our model on datasets of different sizes. It's interesting to notice that the masked language model perplexity consistently improves while the dataset size increases. However, the parsing performance does not improve as significantly. This may suggest that while we increase the training set for pre-trained language models, we can add more common sense and other world knowledge to the model. However, the amount of linguistic knowledge that the model can acquire is limited by the model's architecture. This observation agrees with the findings presented in Hu et al. (2020).

6 Conclusion

In this paper, we introduce Joint Distance Parser (JDP), a novel dependency and constituency joint parsing algorithm. Based on JDP, we propose StructFormer, a new unsupervised parsing algorithm that does unsupervised dependency and constituency parsing at the same time. We also introduced a new masking mechanism for self-attention that allows each attention head to focus on a specific mixture of dependencies. This brings Transformers closer to modeling a directed dependency graph. The experiments show premising results that StructFormer can induce meaningful dependency and constituency structures and achieve better performance on masked language model task. This research provides a new path to build more linguistic bias into pre-trained language model.

References

Phil Blunsom and Trevor Cohn. Unsupervised induction of tree substitution grammars for dependency parsing. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pp. 1204–1213, 2010.

Shay B Cohen and Noah A Smith. Shared logistic normal distributions for soft parameter tying in unsupervised grammar induction. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pp. 74–82, 2009.

Shay B Cohen, Dipanjan Das, and Noah A Smith. Unsupervised structure prediction with non-parallel multilingual guidance. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pp. 50–61, 2011.

Hal Daumé III. Unsupervised search-based structured prediction. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pp. 209–216, 2009.

Hiroyuki Deguchi, Akihiro Tamura, and Takashi Ninomiya. Dependency-based self-attention for transformer nmt. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2019)*, pp. 239–246, 2019.

Andrew Drozdov, Patrick Verga, Mohit Yadav, Mohit Iyyer, and Andrew McCallum. 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)*, pp. 1129–1141, 2019.

Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A Smith. Recurrent neural network grammars. In *Proceedings of NAACL-HLT*, pp. 199–209, 2016.

Jennifer Gillenwater, Kuzman Ganchev, João Graça, Fernando Pereira, and Ben Taskar. Sparsity in dependency grammar induction. *ACL 2010*, pp. 194, 2010.

Junxian He, Graham Neubig, and Taylor Berg-Kirkpatrick. Unsupervised learning of syntactic structure with invertible neural projections. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 1292–1302, 2018.

Jennifer Hu, Jon Gauthier, Peng Qian, Ethan Wilcox, and Roger P Levy. A systematic assessment of syntactic generalization in neural language models. *arXiv preprint arXiv:2005.03692*, 2020.

Yong Jiang, Wenjuan Han, Kewei Tu, et al. Unsupervised neural dependency parsing. Association for Computational Linguistics (ACL), 2016.

Taeuk Kim, Jihun Choi, Daniel Edmiston, and Sang-goo Lee. Are pre-trained language models aware of phrases? simple but strong baselines for grammar induction. *arXiv preprint arXiv:2002.00737*, 2020.

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

Yoon Kim, Alexander M Rush, Lei Yu, Adhiguna Kuncoro, Chris Dyer, and Gábor Melis. 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)*, pp. 1105–1117, 2019b.

Dan Klein and Christopher D Manning. Corpus-based induction of syntactic structure: Models of dependency and constituency. In *Proceedings of the 42nd annual meeting of the association for computational linguistics (ACL-04)*, pp. 478–485, 2004.

Adhiguna Kuncoro, Lingpeng Kong, Daniel Fried, Dani Yogatama, Laura Rimell, Chris Dyer, and Phil Blunsom. Syntactic structure distillation pretraining for bidirectional encoders. *arXiv preprint arXiv:2005.13482*, 2020.

Hongyin Luo, Lan Jiang, Yonatan Belinkov, and James Glass. Improving neural language models by segmenting, attending, and predicting the future. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 1483–1493, 2019.

Diego Marcheggiani and Ivan Titov. Encoding sentences with graph convolutional networks for semantic role labeling. *arXiv preprint arXiv:1703.04826*, 2017.

Ryan McDonald, Slav Petrov, and Keith B Hall. Multi-source transfer of delexicalized dependency parsers. 2011.

Yutaro Omote, Akihiro Tamura, and Takashi Ninomiya. Dependency-based relative positional encoding for transformer nmt. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2019)*, pp. 854–861, 2019.

John K Pate and Sharon Goldwater. Unsupervised dependency parsing with acoustic cues. *Transactions of the Association for Computational Linguistics*, 1:63–74, 2013.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*, 2019.

Yikang Shen, Zhouhan Lin, Chin-wei Huang, and Aaron Courville. Neural language modeling by jointly learning syntax and lexicon. In *International Conference on Learning Representations*, 2018a.

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

Yikang Shen, Shawn Tan, Alessandro Sordoni, and Aaron Courville. Ordered neurons: Integrating tree structures into recurrent neural networks. In *International Conference on Learning Representations*, 2018c.

Valentin I Spitkovsky, Hiyan Alshawi, Angel Chang, and Dan Jurafsky. Unsupervised dependency parsing without gold part-of-speech tags. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pp. 1281–1290, 2011.

Emma Strubell, Patrick Verga, Daniel Andor, David Weiss, and Andrew McCallum. Linguistically-informed self-attention for semantic role labeling. *arXiv preprint arXiv:1804.08199*, 2018.

Kewei Tu and Vasant Honavar. Unambiguity regularization for unsupervised learning of probabilistic grammars. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pp. 1324–1334, 2012.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pp. 5998–6008, 2017.

Yaushian Wang, Hung-Yi Lee, and Yun-Nung Chen. Tree transformer: Integrating tree structures into self-attention. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 1060–1070, 2019.