COMBINER: INDUCTIVELY LEARNING TREE STRUCTURED ATTENTION IN TRANSFORMERS

Anonymous authors
Paper under double-blind review

ABSTRACT

Transformers employ dense attention mechanisms over text which can fail to capture or utilize the strong intrinsic structures present in natural language. This paper presents the Combiner model, a new Transformer architecture that learns tree-structured attention patterns inductively from language. Instead of dense or pre-specified structures, Combiner automatically learns tree-structured attention connections using a novel sparse residual attention mechanism. It first employs a sparsity-inducing gate that learns to prune attention connections in each network layer, so as to determine the nodes to be combined. Then the learned connections are propagated through layers using hierarchical attention blocks, which combine the sub-tree nodes in a bottom-up manner. Our experiments demonstrate the robust modeling performance of Combiner and usefulness of structures it learns in various information retrieval and unsupervised sentence parsing tasks. By leveraging search session structures, Combiner outperforms other pre-trained Transformers in generative query suggestion. Moreover, the learned tree structures align well with linguistic structures and improve the current state-of-the-art unsupervised constituency parsing by 8 average sentence-level F1.

1 INTRODUCTION

Pre-trained Transformers, such as BERT (Devlin et al., 2019), XNLI (Conneau et al., 2018), GPT-2 (Radford et al., 2019), XL-NET (Yang et al., 2019), and SpanBERT (Joshi et al., 2019), have changed the landscape of natural language representation, understanding, and generation. What makes them so powerful is their versatility and transferability. With fine-tuning, these models produce state-of-the-art results on many different downstream tasks (Wang et al., 2018; Rajpurkar et al., 2016; 2018). The core component of these models is the Transformer architecture (Vaswani et al., 2017). It includes multiple feed-forward layers with self-attention on top of each other, enabling the model to consume information from the entire text sequence. The self-attentive architectures are robust and efficient in training and can be stacked to create very deep and highly effective networks (Shoeybi et al., 2019).

The high effectiveness and the simplicity of the attention architectures in Transformers spurred many efforts trying to understand the underlying mechanisms. While the studies differ in their detailed findings, a common observation is that the learned attention connections are dense across tokens, making it challenging to interpret (Clark et al., 2019; Jain & Wallace, 2019; Brunner et al., 2019; Petroni et al., 2019; Bosselut et al., 2019). Also, their dense specification does not fully leverage the strong syntactic and semantic structures present in natural language, which makes the pre-trained models relying on a large number of parameters and may even weak their generalization ability.

One potential way to alleviate these shortcomings is to inform the neural network through structural priors. For example, one can integrate syntax trees via special position embeddings (Shiv & Quirk, 2019); use the convolutional structure to enforce sparse attention matrices (Child et al., 2019); or apply recurrent attentions to model long text sequences (Dai et al., 2019). As those structures include informative prior knowledge, integrating them has lead to more intuitive and effective Transformer networks in various tasks (Yang et al., 2019).

On the other hand, the intrinsic structures in data may vary task-by-task; pre-specifying those structures may not always be feasible. This paper presents another solution that uses a better structural

inductive bias to encourage the Transformer to learn the structures from data automatically. Specifically, we start from tree structures, which have strong linguistic grounding and rich history in discourse analysis (Marcus et al., 1993; Shen et al., 2019), and propose Combiner, a new Transformer that combines tokens into trees hierarchically across its layers using learned sparse attention connections. This is achieved by a novel Sparse Hierarchical Attention (SHA) mechanism, which, in each Combiner layer, uses a Sparse Attention Gate (SAG) to decide which tokens to connect and then uses a Hierarchical Attention Block (HAB) to propagate the connections across network layers. With multiple Combiner layers stacked together, SHA simulates a hierarchical process that combines tokens into trees using learned attention connections.

Learning the tree structures via the structural inductive bias from SHA brings advanced flexibility to Combiner. The trees are learned inductively from data without any supervision; neither pre-specified structures nor structural labels are required. The attention connections can form consecutive spans or have skips; a token can be merged with any number of other tokens in each Combiner layer, allowing a general set of tree shapes. Moreover, Combiner can easily be integrated with traditional Transformers to leverage pre-trained language models (Devlin et al., 2019).

Our experiments demonstrate Combiner's effectiveness in Search Session Understanding, a real-world application in information retrieval, and unsupervised constituency parsing, a core structured language modeling task. When integrated with $\text{BERT}_{\text{BASE}}$, Combiner outperforms $\text{BERT}_{\text{LARGE}}$ in Generative Query Suggestion accuracy (Sordoni et al., 2015) by 4 BLEU scores with fewer parameters, and significantly outperforms other BERT variations with pre-specified session structures. By combining the advantage of $\text{BERT}_{\text{BASE}}$ pre-training, Combiner achieves state-of-the-art performance on the Penn Tree Bank unsupervised constituency parsing task, improving the average sentence-level F1 score by over 15%.(Kim et al., 2019; Drozdov et al., 2019).

2 METHOD

Before presenting our approach, we briefly revisit the self-attention mechanism, the core component of Transformer (Vaswani et al., 2017).

The self-attention in Transformer fuses token representations from previous layers to create more contextualized representations. Formally, let H be the input token representations with one row vector for each of the n tokens. The attention mechanism starts by projecting the tokens into three subspaces: the query subspace Q, key subspace K, and value subspace V, with corresponding projection matrices W^Q, W^K, W^H. It then computes the $n \times n$ attention matrix M as

$$M = \text{softmax}\left(\frac{Q \cdot K^T}{\sqrt{d_k}}\right), \tag{1}$$

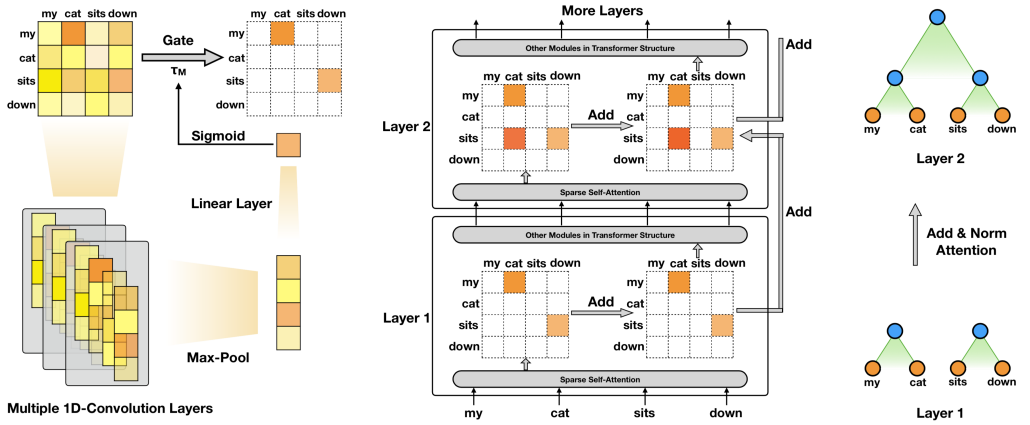
where d_k is a scaling factor. Then the attention fuses the value representations V of all tokens using the attention matrix M:

$$H' = M \cdot V^T. \tag{2}$$

Each row in H' is a more contextualized representation of the corresponding token using information from potentially all other tokens' representations. After the attention operation, additional feed-forward, dropout and normalization layers may be used. In practice, especially when pre-trained, the learned attention matrices M are often dense with rather scattered attention scores between tokens. This raises many questions about whether Transformers learn meaningful language properties (Clark et al., 2019; Jain & Wallace, 2019; Brunner et al., 2019; Petroni et al., 2019).

2.1 THE COMBINER TRANSFORMER

Instead of the dense attention connections in the vanilla Transformer, Combiner uses a sparse attention mechanism to encourage the network to learn tree-structured attention patterns, which may better explore the intrinsic structures in language. We propose a new Sparse Hierarchical Attention (SHA) mechanism. It includes two components, as shown in Fig. 1. First, SHA uses a *Sparse Attention Gate* (SAG) to enable a Combiner layer to learn sparse attentions. Then it uses a *Hierarchical Attention Block* (HAB) to propagate the learned attention connections through layers. The two together help Combiner learn tree-structured attentions inductively from data.



(a) Sparse Attention Gate (b) Hierarchical Attention Block (c) Formed Tree

Figure 1: Combiner Architecture. The Sparse Attention Gate truncates entries in the self-attention matrix (a). The new sparser attention matrix is passed on via the Hierarchical Attention Block (b). Repeating this process merges nodes hierarchically into trees through layers (c).

2.1.1 Sparse Attention Gate

The Sparse Attention Gate enforces sparsity via a learned, input-dependent gate threshold τ:

$$\hat{M} = \text{relu}(M - \tau_M). \tag{3}$$

Put differently, we truncate all the attention weights in M that are below the learned threshold τ_M to create a sparser matrix $\hat{M}$. Intuitively, a non-zero score in $\hat{M}$, (i.e., $\hat{M}_{ij} > 0$) corresponds to an association between token i and j learned by the Combiner layer.

We learn this gating function from the current attention matrix M as follows:

$$\tau_M = \text{sigmoid}(\text{linear}(\text{max-pool}(\text{CNN}(M)))). \tag{4}$$

In other words, the threshold is learned through a standard convolution neural network with a subsequent max-pooling, linear, and sigmoid layer, producing a scalar between 0 and 1 as the gate for the attention matrix M. The convolution is performed across the rows of M, the normalized attention scores from each token with respect to all other tokens.

2.1.2 Hierarchical Attention Block

To create valid tree structures, tokens combined in a lower layer need to remain connected in all the higher layers. The Hierarchical Attention Block enables this by propagating the attention connectivity patterns from lower layers to higher layers, i.e., for layer l,

$$\tilde{M}^l = \text{softmax}(\hat{M}^l + \tilde{M}^{l-1}), \tag{5}$$

where $\hat{M}^l$ is the output of the sparse attention gate from Equation (3), and $\tilde{M}^0 = \mathbf{0}$. The softmax operation re-normalizes the attention maps.

Equation (5) ensures that the connections $\tilde{M}^{l-1}$ from the previous layer are inherited by the current layer as any non-zero entry $\tilde{M}^{l-1}_{ij} > 0$ guarantees that $\tilde{M}^l_{ij} > 0$.

2.1.3 Overall Architecture

The overall architecture of Combiner is similar to standard Transformers, with the dense attention mechanism replaced by our Sparse Hierarchical Attention mechanism. Specifically, the l-th Combiner layer takes the input H^{l-1}, conducts the same three projections (Q, K, V), and calculates the fused representation:

$$\hat{H}^l = \tilde{M}^l \cdot (V^l)^T, \tag{6}$$

which uses the sparse hierarchical attention matrix $\tilde{M}^l$ from Eqn. (5). After that, we create a combined representation for all nodes that have been merged in the current layer $\tilde{M}^l$:

$$\tilde{H}^l = \text{L}_1\text{-Norm}(\text{Boolean}(\tilde{M}^l)) \cdot \hat{H}^l. \tag{7}$$

Boolean is an element-wise operation and is 1 if the corresponding element in $\tilde{M}^l$ is non-zero. The L_1-Norm normalizes each row to sum to one. Eqn. (7) combines the connected tokens in current Combiner layer (tree level) by averaging and sharing their representations. After that, standard feed-forward and layer norm layers are used, and the Combiner layer is stacked multiple times, following BERT (Devlin et al., 2019).

Starting from the token embeddings, each Combiner layer learns sparse attention connections between tokens and mean-pools the representation of connected tokens into combined representations. This combining operation is repeated layer by layer and merges more and more tokens, thus enables the neural network to learn tree-structured attention patterns from data automatically. Intuitively, the sparsity-inducing and information merging objectives in SAG and HAB incentivize the Combiner to consume less model capacity to represent the input texts; they force the model to compress information in a more efficient and structured way. This is related to the minimum description length principle (Rissanen, 1978), a formalization of Occam's razor in which the best hypothesis (a model and its parameters) for a given set of data is the one that leads to the best compression of the data.

The combiner can be trained/fine-tuned the same as vanilla Transformers, for example, using the Masked LM loss (Devlin et al., 2019). No additional labels or hyper-parameters are required.

3 Experimental Setup

Search Session Understanding. Understanding user sessions is a crucial task in information retrieval and has many applications in search engines, such as session-based search and query suggestion (Croft et al., 2010). A search session is defined as a sequence of queries issued by a user in a short time frame, expressing a single or a sequence of information needs. Search session understanding is an interesting task because search sessions are rich in non-trivial higher-level structures that arise from users switching tasks, reformulating queries, or exploring related topics (Sordoni et al., 2015; Wang et al., 2013).

Our experiments evaluate a model's session understanding ability regarding two tasks: *Masked Query Prediction* and *Generative Query Suggestion*. The Masked Query Prediction task is akin to the Masked LM task (Devlin et al., 2019), but masks entire queries in the session for training and testing. The Generative Query Suggestion task provides the model the entire session except for the last query, and asks the model to suggest the last query using the previous context. It is an important feature for commercial search engines.

Datasets. This task uses search sessions sampled from the logs of a commercial search engine. Following standard session processing, we sampled 1.3 million sessions in En-US market with three queries or more. The sessions are split into train/validation/test as 1.1M/100K/100K.

Evaluation Metrics. The Mask Query Prediction uses token-level accuracy and Jaccard similarity, as well as word-level BLEU-2 scores to evaluate the model prediction ability. The Query Suggestion performances are evaluated by word-level BLEU-2 only. We chose length two in BLEU, which corresponds to the average length of web queries (Croft et al., 2010).

Baselines. Baselines include the standard frequency-based ADJ (Sordoni et al., 2015), which suggests the query that most frequently appears after the current query, counted over one month worth of search logs, and four BERT baselines. The first two BERTs are standard $\text{BERT}_{\text{BASE}}$ and $\text{BERT}_{\text{LARGE}}$ (Devlin et al., 2019). The last two, $\text{BERT}_{\text{WIDE}}$ and $\text{BERT}_{\text{HIER}}$, are developed by us.

$\text{BERT}_{\text{WIDE}}$ maintains two attention mechanisms, one initialized from $\text{BERT}_{\text{BASE}}$ and one trained from scratch, and combines the two attention mechanisms similar to Combiner. The difference is that both attention mechanisms in $\text{BERT}_{\text{WIDE}}$ are standard dense attentions, while Combiner uses SHA to learn structured attentions inductively. It is a specifically designed baseline to assess the effectiveness and the ability of Combiner's inductive structure learning.

$\text{BERT}_{\text{HIER}}$ is inspired by the hierarchical Transformer (Liu & Lapata, 2019). It starts from $\text{BERT}_{\text{BASE}}$ and adds explicit hierarchical attention structures on top of individual queries.

Table 1: Query Prediction and Suggestion Results. Percentages are relative performance compared to previous SOTA, $\text{BERT}_{\text{LARGE}}$. Best results are marked in **bold**.

		Mask Query Prediction			Query Suggestion	
		Token-Level		Word-Level	Word-Level	
Model	**# Para**	**Accuracy**	**Jaccard**	**BLEU-2**	**BLEU-2**	
ADJ (One Month Log) (Sordoni et al., 2015)	–	–	–	–	0.391	-15.7%
$\text{BERT}_{\text{BASE}}$ + Mask LM	110M	0.696	0.702	0.300	0.386	-16.8%
$\text{BERT}_{\text{WIDE}}$ + Mask LM	152M	0.703	0.708	0.301	0.392	-15.5%
$\text{BERT}_{\text{BASE}}$ + Mask Query	110M	0.741	0.754	0.325	0.464	–
$\text{BERT}_{\text{WIDE}}$ + Mask Query	152M	0.744	0.760	0.328	0.478	+3.0%
$\text{BERT}_{\text{HIER}}$ + Mask Query	156M	0.760	0.778	0.336	0.517	+11.4%
$\text{BERT}_{\text{LARGE}}$ + Mask Query	340M	0.765	0.794	0.341	0.535	+15.3%
Combiner	167M	**0.798**	**0.823**	**0.369**	**0.573**	+23.5%

Table 2: Unsupervised Constituency Parsing Results. Percentages are relative performance compared to the best performing ON-LSTM architecture. Contemporary methods are marked with *. The best results in each column are marked in **bold**. Mean, Signma, and Max are from five independent runs.

	Sentence Level F1				Accuracy on by Tag			
Model	**Mean (Sigma)**		**Max**		**ADJP**	**NP**	**PP**	**INTJ**
Balanced Trees	24.5 (0.0)	-48.8%	24.5	-50.4%	22.1	20.2	9.3	55.9
Left Branching	9.0 (0.0)	-81.1%	9.0	-81.8%	–	–	–	–
Right Branching	39.8 (0.0)	-16.6%	39.8	-19.4%	–	–	–	–
PRPN-UP (Shen et al., 2018)	38.3 (0.5)	-19.7%	39.8	-19.4%	28.7	65.5	32.7	**0.0**
PRPN-LM (Shen et al., 2018)	35.0 (5.4)	-26.6%	42.8	-13.4%	37.8	59.7	61.5	**100.0**
ON-LSTM 1st-layer (Shen et al., 2019)	20.0 (2.8)	-58.1%	24.0	-51.4%	38.1	23.8	18.3	**100.0**
ON-LSTM 2nd-layer (Shen et al., 2019)	47.7 (1.5)	–	49.4	–	46.2	61.4	55.4	0.0
ON-LSTM 3rd-layer (Shen et al., 2019)	36.6 (3.3)	-23.3%	40.4	-18.2%	44.8	57.5	47.2	0.0
DIORA (Drozdov et al., 2019)	55.7 (8.5)	+16.8%	56.2	+13.8%	n.a.	n.a.	n.a.	n.a.
Compound PCFG (Kim et al., 2019)	55.2 (n.a.)	+15.7%	60.1	+21.7%	n.a.	n.a.	n.a.	n.a.
Tree Transformer (Best) (Wang et al., 2019)*	50.5 (n.a.)	+5.9%	52.0	+5.3%	24.7	67.6	52.3	n.a.
Combiner	**64.1** (0.89)	+34.4%	**65.1**	+31.8%	**53.8**	**68.1**	**66.3**	58.0

All four BERT models start from the same pre-trained model and then are fine-tuned on the same data as Combiner. For the fine-tuning step, we explore two different objectives: standard Mask-LM (Devlin et al., 2019) and Mask-Query. The latter randomly masks an entire query in each session as a training target, inspired by MASS (Song et al., 2019) and SpanBERT (Joshi et al., 2019). All methods use the same hyper-parameters and training strategies as Combiner; the only difference lies in their respective architectures.

Unsupervised Constituency Parsing. As a core NLP task, unsupervised constituency parsing allows us to evaluate the quality and meaningfulness of the learned tree structures in Combiner. We adopt the same dataset and evaluation setup as previous research (Htut et al., 2018; Shen et al., 2019) and evaluate our models on benchmark Penn Tree Bank (PTB) dataset. Methods requiring large scale language model training are first continuously trained on the Wikitext-103 dataset (Merity et al., 2017) and then fine-tuned (using only the unlabeled text) on PTB.

Baselines. We include state-of-the-art baselines representing a wide set of approaches, including rule-based tree methods, Parsing-Reading-Predict Networks (PRPN) (Shen et al., 2018), Ordered Neurons (Shen et al., 2019), DIORA (Drozdov et al., 2019), Compound Probabilistic Context-Free Grammars (PCFG, current SOTA) (Kim et al., 2019), and Tree Transformer (Wang et al., 2019). All the evaluation are kept consistent and their published results are compared.

Fine-Tuning Combiner from Pre-Trained $\text{BERT}_{\text{BASE}}$. We integrate the pre-trained $\text{BERT}_{\text{BASE}}$ into Combiner and continuously train from it. Specifically, we maintain two attention paths, a structured one from Combiner and another one from pre-trained $\text{BERT}_{\text{BASE}}$. The two act in parallel in each layer and are concatenated in the attention output: $H^l_{\text{Fusion}} = [H^l_{\text{BERT}} \circ H^l_{\text{Combiner}}]$. Note that this leads to a similar number of parameters with $\text{BERT}_{\text{WIDE}}$ but much fewer than $\text{BERT}_{\text{LARGE}}$.

Table 3: Performances of different sparse attention strategies: fixed gating threshold, top-k pooling, and sparse attention gates. Best results are marked in **bold**.

	Mask Query Prediction			Query Suggestion	Unsupervised Parsing	
Model	**Token-Level**		**Word-Level**	**Word-Level**	**Sentence-Level F1**	
	Accuracy	**Jaccard**	**BLEU-2**	**BLEU-2**	**Mean (Sigma)**	**Max**
Fixed Threshold (0.50)	0.753	0.771	0.338	0.496	56.9 (1.46)	57.4
Fixed Threshold (0.55)	0.757	0.778	0.341	0.504	57.6 (1.28)	58.7
Top-2 Pooling Per Token	0.772	0.790	0.347	0.522	59.2 (1.22)	60.6
Top-3 Pooling Per Token	0.769	0.783	0.344	0.509	56.6 (1.88)	58.2
One Linear Layer	0.787	0.803	0.355	0.544	61.8 (1.02)	62.9
One Layer CNN	0.791	0.808	0.361	0.557	62.2 (0.95)	63.4
Two Layer CNN	0.794	0.815	0.363	0.568	63.0 (1.75)	64.3
Three Layer CNN	**0.798**	**0.823**	**0.369**	**0.573**	**64.1** (0.89)	**65.1**

The BERT part is initialized from the released $\text{BERT}_{\text{BASE}}$ weights and the Combiner part is trained from scratch. Thus Combiner parameters are fine-tuned (or continuous trained) the same as all our BERT baselines. We leave pre-training for future work as it requires significantly more computational resources. In the Search Session Understanding tasks, Combiner is trained using 1.2M training sessions using the Masked Query. In the parsing task, it is trained on WikiText and then fine-tuned on the training set of PTB, using Masked LM.

We provide more details on experiment settings and model details in the Appendix.

4 Results and Analysis

We are now present our results before analyzing the induced attention patterns in more detail.

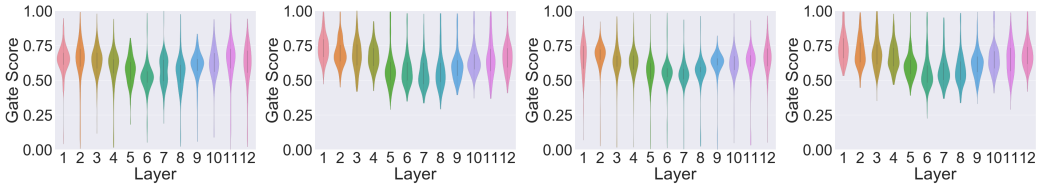
4.1 Overall Results

Search Session Understanding. As shown in Table 1, Combiner outperforms other state-of-the-art methods as well as our baselines on all evaluation metrics. Notably, on the real-world task, Query Suggestion, Combiner improves over $\text{BERT}_{\text{LARGE}}$ by 7.1% although the latter has a much larger set of parameters. It also outperforms ADJ, the IR-style frequency-based baseline, by 46%. Even though it relies on simple frequencies, ADJ leverages the wisdom of crowd from one-month commercial search log and is a very strong baseline (Sordoni et al., 2015).

Among the three $\text{BERT}_{\text{BASE}}$ versions, $\text{BERT}_{\text{WIDE}}$ uses an additional set of dense attention modules and performs only 3% better than $\text{BERT}_{\text{BASE}}$, though using doubled parameter space. $\text{BERT}_{\text{HIER}}$ is able to leverage structural information in the session data and outperforms $\text{BERT}_{\text{BASE}}$. However, its structures have to be manually defined and often are not optimal; it merely performs on par with $\text{BERT}_{\text{LARGE}}$ which simply uses stronger pre-training. In contrast, Combiner learns structures inductively from data and outperforms $\text{BERT}_{\text{LARGE}}$, using only half parameters.

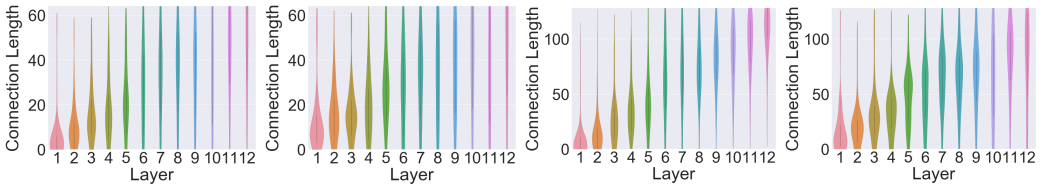
Structured Language Modeling. As the results in Table 2 demonstrate, Combiner outperforms all other baselines by large margins on all metrics except for accuracy on INTJ tags. Compared to ON-LSTM which also uses an inductive bias to guide the RNN language model to learn tree structures, Combiner is able to leverage pre-training effectively, performing 30%+ better on sentence-level F1 scores. Compared to the current SOTA method Compound PCFG, which uses reinforcement learning to construct parsing trees explicitly, Combiner performs 9 points better on average in terms of absolute F1 while having much lower variance across different runs.

Compared with the parallel work Tree Transformer (Wang et al., 2019), Combiner performs significantly better, improving its F1 score by 25%+, despite that Tree Transformer was specifically designed for constituency parsing. We attribute this to the fact that Combiner uses a more flexible inductive bias, softer constraints, and effectively makes use of the pre-trained model parameters. Our experiments also demonstrate that Combiner was able to do this effectively on both IR and NLP tasks.



(a) Combiner on Search. (b) Top-2 on Search. (c) Combiner on Text. (d) Top-2 on Text.

Figure 2: Distribution of the learned SAG thresholds vs. Top-2 pooling on Search sessions and WikiText. X-axes represent the network layers and Y-axes indicate the induced thresholds.



(a) Combiner on Search. (b) Top-2 on Search. (c) Combiner on Text. (d) Top-2 on Text.

Figure 3: Distribution of attention distances of Combiner and Top-2 pooling trained on Search sessions and WikiText. X-axes mark the network layers and Y-axes mark the connection distances.

4.2 Effectiveness of Sparse Attention Strategies

This experiment compares different strategies that can be used to induce sparse attention structures. More specifically, we replace the SAG in Eqn. (3) by two different unsupervised strategies: *Fixed Threshold* which uses manually defined cut-off in the attention connections, and *Top-k Pooling* which only keeps the top k attention connections for each token. In addition, we also experiment with different neural networks to learn the sparse attention gate, i.e., Eqn. (4).

The results on both tasks are shown in Table 3. Among the alternative strategies, Top-2 pooling performs the best on all metrics. Using a fixed threshold does not lead to good results and performs worse than $\text{BERT}_{\text{LARGE}}$. This indicates that there are substantial differences between layers and also datasets, making it impossible to accommodate with a manually-set cut-off. Not surprisingly, our fully data-driven method SAG performs best and the performance increases with more complex gating networks. With its crucial role in Combiner, it would be interesting to explore more sophisticated sparse mechanisms in future research.

4.3 Inductive Structure Learning Analysis

This experiment analyzes Combiner's inductive structure learning behaviors. More analyses and examples of the learned structures can be found in Appendix.

The **distributions of gate scores** learned in Combiner's SAG and Top-2 pooling are illustrated in Figure 2. It shows that Combiner inductively prefers sparse attention: most of the learned thresholds are above 0.5, which prunes the softmax normalized attention to at most one connections per token, i.e., to binary trees, although there is no hard constraint in Combiner enforcing the binary structure. The learned gate scores also distribute differently in SAG and Top-2. SAG has more concentrated thresholds, evidenced by the concentration towards the mean in each layer. At the same, SAG also has a more flexible range as the gates can be anywhere between 0-1 in SAG but not in Top-2. Another observation is that Combiner's gates are more informed by the underlying data: The scores have a higher variance on Search sessions which are noisier and have more diverse structures than Wikipedia Text.

We further investigate how far apart the merged tokens are. We define the *attention distance* to be the absolute difference in position between two connected tokens in the learned sparse attention. The **distributions of attention distances** in the trees learned by Combiner and Top-2 are shown in Figure 3. As expected, the attention distances start small in the lower layers, merging nearby

and closely related tokens, and grow to align longer-term dependencies in higher layers. The lower attention layers, through favoring continuous spans, do not necessarily connect adjacent tokens. In fact, even in the first layer, many attentions connect tokens not adjacent to each other (length > 1). Combiners also learn long-term dependencies in the top layers. Many of the last layers' attentions span across queries or sentences.

We provide more analyses of the inductively learned structures in Appendix.

5 Related Work

Augmenting models with appropriate structured inductive biases lies is at the heart of deep learning: Convolutional structures, recurrent structures, and long-term-short-term dependency structures are standard components of deep neural networks (Goodfellow et al., 2016). Integrating these fixed structures into neural network designs has achieved great successes in many areas, for example, linguistic trees in text sequence models (Tai et al., 2015), document hierarchical structures in document encoders (Liu & Lapata, 2019; Yang et al., 2016), search/dialog hierarchies in conversation models (Sordoni et al., 2015; Serban et al., 2016), and graph structures in graph neural networks (Veličković et al., 2018; Wu et al., 2019). Instead of fixed, pre-specified structures, ON-LSTM is more related to this work as it also inductively learns tree structures using a structured gating mechanism (Shen et al., 2019).

Integrating pre-specified structures are also effective in Transformers. The recurrent structures in text segments significantly help Transformer-XL to model long-term dependencies (Dai et al., 2019) and lead to better pre-trained contextual representations (Yang et al., 2019). Explicitly modeling n-gram spans when training also effectively improves Transformers in many NLP tasks (Song et al., 2019; Joshi et al., 2019). Using pre-specified sparse patterns, such as fixed strides (Child et al., 2019) or log-spaced strides (Li et al., 2019), effectively reduce the computation cost and lengthen the attention spans of Transformers. Linguistic parsing trees can also be integrated into a Transformer's position encodings (Shiv & Quirk, 2019).

There has also been recent work on learning sparse or structured attention patterns in Transformers. For example, Sukhbaatar et al. (2019) leverages soft masks to learn dynamic attention spans which can differ from token to token. Correia et al. (2019) use α-entmax transformations to learn sparse attention weights which allow the attentions to skip tokens rather than being restricted to spans. Combiner's sparse attention gate has a similar goal as the α-entmax mechanism, but take a much simpler form; also its sparsity is entirely data-driven, introducing no further hyper-parameters.

Another closely related approach is Tree Transformer (Wang et al., 2019), which uses a constituent prior to constrain the tokens to only self-attend within the same constituent and to form constituent trees. Combiner differs as it allows more flexible attention structures with no explicit constraints on where each token can attend to. This provides more flexibility as the underlying structures in many scenarios do not have to form a well-specified tree, for example, search and dialogue sessions often include "skip connections" as two topics may intertwine. Combiner's more flexible form of structure inductive bias also effectively integrates the power of language model pre-training, achieving much stronger performance in unsupervised constituency parsing without explicit priors or constraints (Wang et al., 2019).

6 Conclusion

Our novel architecture Combiner integrates powerful structural inductive bias learning with the strong generalization properties of pre-trained Transformers. Combiner automatically leverages hierarchical intrinsic structures in the data, without the need for prior knowledge nor explicit structural constraints. In doing so, it achieves substantially better prediction and generation accuracy with fewer parameters than previous pre-trained Transformer-based models, yielding new state-of-the-art results in both generative query suggestion and unsupervised constituency parsing. These results show the promising potential of inductive structure learning in deep neural networks and we envision many future explorations in this direction.

REFERENCES

Antoine Bosselut, Hannah Rashkin, Maarten Sap, Chaitanya Malaviya, Asli Celikyilmaz, and Yejin Choi. Comet: Commonsense transformers for automatic knowledge graph construction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 4762–4779, 2019.

Gino Brunner, Yang Liu, Damián Pascual, Oliver Richter, and Roger Wattenhofer. On the validity of self-attention as explanation in transformer models. *arXiv preprint arXiv:1908.04211*, 2019.

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. What does BERT look at? an analysis of BERT's attention. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pp. 276–286, 2019.

Alexis Conneau, Ruty Rinott, Guillaume Lample, Adina Williams, Samuel Bowman, Holger Schwenk, and Veselin Stoyanov. Xnli: Evaluating cross-lingual sentence representations. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 2475–2485, 2018.

Gonçalo M Correia, Vlad Niculae, and André FT Martins. Adaptively sparse transformers. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2019.

W Bruce Croft, Donald Metzler, and Trevor Strohman. *Search engines: Information retrieval in practice*, volume 520. Addison-Wesley Reading, 2010.

Zihang Dai, Zhilin Yang, Yiming Yang, William W Cohen, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 2978–2988, 2019.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 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 (NAACL-HLT)*, pp. 4171–4186, 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 (NAACL-HLT)*, pp. 1129–1141, 2019.

Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT Press, 2016.

Phu Mon Htut, Kyunghyun Cho, and Samuel Bowman. Grammar induction with neural language models: An unusual replication. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 4998–5003, 2018.

Sarthak Jain and Byron C Wallace. Attention is not explanation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pp. 3543–3556, 2019.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S Weld, Luke Zettlemoyer, and Omer Levy. Spanbert: Improving pre-training by representing and predicting spans. *arXiv preprint arXiv:1907.10529*, 2019.

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 (ACL)*, pp. 2369–2385, 2019.

Shiyang Li, Xiaoyong Jin, Yao Xuan, Xiyou Zhou, Wenhu Chen, Yu-Xiang Wang, and Xifeng Yan. Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. In *Proceedings of the 33rd Conference on Neural Information Processing Systems (NeurIPS)*, 2019.

Yang Liu and Mirella Lapata. Hierarchical transformers for multi-document summarization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 5070–5081, 2019.

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

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In *Proceedings of the 5th International Conference on Learning Representations (ICLR)*, 2017.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in PyTorch. In *NeurIPS Autodiff Workshop*, 2017.

Fabio Petroni, Tim Rocktäschel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, Alexander H Miller, and Sebastian Riedel. Language models as knowledge bases? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2019.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8), 2019.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 2383–2392, 2016.

Pranav Rajpurkar, Robin Jia, and Percy Liang. Know what you don't know: Unanswerable questions for squad. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 784–789, 2018.

Jorma Rissanen. Modeling by shortest data description. *Automatica*, 14(5):465–471, 1978.

Iulian V Serban, Alessandro Sordoni, Yoshua Bengio, Aaron Courville, and Joelle Pineau. Building end-to-end dialogue systems using generative hierarchical neural network models. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI)*, 2016.

Yikang Shen, Zhouhan Lin, Chin-Wei Huang, and Aaron Courville. Neural language modeling by jointly learning syntax and lexicon. In *Proceedings of the 6th International Conference on Learning Representations (ICLR)*, 2018.

Yikang Shen, Shawn Tan, Alessandro Sordoni, and Aaron Courville. Ordered neurons: Integrating tree structures into recurrent neural networks. In *Proceedings of the 7th International Conference on Learning Representations (ICLR)*, 2019.

Vighnesh Leonardo Shiv and Chris Quirk. Novel positional encodings to enable tree-based transformers. In *Proceedings of the 33rd Conference on Neural Information Processing Systems (NeurIPS)*, 2019.

Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. Megatron-lm: Training multi-billion parameter language models using model parallelism. *arXiv preprint arXiv:1909.08053*, 2019.

Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. Mass: Masked sequence to sequence pre-training for language generation. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, pp. 5926–5936, 2019.

Alessandro Sordoni, Yoshua Bengio, Hossein Vahabi, Christina Lioma, Jakob Grue Simonsen, and Jian-Yun Nie. A hierarchical recurrent encoder-decoder for generative context-aware query suggestion. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management (CIKM)*, pp. 553–562, 2015.

Sainbayar Sukhbaatar, Edouard Grave, Piotr Bojanowski, and Armand Joulin. Adaptive attention span in transformers. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 331–335, 2019.

Kai Sheng Tai, Richard Socher, and Christopher D Manning. 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 (ACL-IJCNLP)*, pp. 1556–1566, 2015.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proceedings of the 31st Conference on Neural Information Processing Systems (NeurIPS)*, pp. 5998–6008, 2017.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. In *Proceedings of the 6th International Conference on Learning Representations (ICLR)*, 2018.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 6th International Conference on Learning Representations (ICLR)*, 2018.

Hongning Wang, Yang Song, Ming-Wei Chang, Xiaodong He, Ryen W White, and Wei Chu. Learning to extract cross-session search tasks. In *Proceedings of the 22nd international conference on World Wide Web (WWW)*, pp. 1353–1364. ACM, 2013.

Yau-Shian 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 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2019.

Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S Yu. A comprehensive survey on graph neural networks. *arXiv preprint arXiv:1901.00596*, 2019.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. *arXiv preprint arXiv:1906.08237*, 2019.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pp. 1480–1489, 2016.

A APPENDIX

A.1 IMPLEMENTATION DETAILS

We implement Combiner in PyTorch (Paszke et al., 2017) by leveraging pytorch-transformers library[1]. All the hyper-parameters in the Transformer part, *e.g.*, hidden dimensions, number of layers, training hyper-parameters, and feed-forward layers, are kept the same with $\text{BERT}_{\text{BASE}}$, in order to be consistent with prior research (Devlin et al., 2019). To make a fair comparison, we implement a 12-layer Combiner model, where each layer has 12 attention heads and 768 hidden units. The only additional parameter is in SAG (Eqn. 4) which we use a three layer 1D CNN for each Combiner layer, convoluted on the rows of the attention matrix. The generative decoder uses standard Transformer decoder following the follow the implementation in Song et al. (2019), which is kept the same in all methods.

We max-pool the multi-head attention in the Combiner side into one attention head, to force it learn one tree structure. Learning multiple diverse tree structures is left for future exploration. To utilize the pre-trained parameters from $\text{BERT}_{\text{BASE}}$, we concatenate the intermediate hidden embeddings of $\text{BERT}_{\text{BASE}}$ and Combiner, and then project them into the 768-D embeddings. In this way, we can load the pre-trained $\text{BERT}_{\text{BASE}}$ parameters and jointly train the whole parameters on our corpus.

For training on the search session dataset, we directly use the uncased $\text{BERT}_{\text{BASE}}$ tokenizer on the raw data. We set the max sequence length as 128 and batch size as 64 for search session data. The input is the whole session, which contains multiple ($>$3) queries, and queries are separated by the [SEP] token. For training on the text datasets, we use the uncased $\text{BERT}_{\text{BASE}}$ tokenizer on the pre-processed corpus and keep the [UNK] tokens for a fair comparison. The max sequence length is then set as 512 and batch size is 64. We use a sliding windows of 512 tokens to construct the training batches. Furthermore, the Apex library[2] is used for mixed precision (fp16) training. The training process usually takes 1-3 days depending on different data sizes.

A.2 EXPERIMENTAL DETAILS ABOUT UNSUPERVISED CONSTITUENCY PARSING

We follow the standard preprocessing steps by lower-casing all the tokens and discarding punctuation, and use the standard splits: 2-21 for fine-tuning, 22 for validation and 23 for test. All the structural learning methods are trained in an unsupervised fashion before being evaluated against the ground-truth labels on PTB. The evaluation follows the same setup from Shen et al. (2019) and use sentence-level unsupervised F_1 as well as accuracy by tag as evaluation metrics. We ignore punctuation and discard trivial spans during the evaluation (Shen et al., 2019; Kim et al., 2019).

A.3 PERPLEXITY RESULTS

The perplexity results on the PTB dataset are demonstrated in Table 4 for further reference. The Transformer-XL (Dai et al., 2019) still shows the best perplexity on the PTB dataset, even better than the pre-trained $\text{BERT}_{\text{LARGE}}$. The most significant reason would be Transformer-XL considers longer token spans during training, while leads to better generalization on PTB, while Combiner and BERTs are restricted by single text segment lengths.. As for our combiner models with different settings, the full model with the three-layer CNN gate shows the best perplexity. The observation is consistent with other experimental results, which indicates that by modeling latent language structures as inductive bias, Combiner learns better language modeling.

A.4 CROSS QUERY FRACTIONS

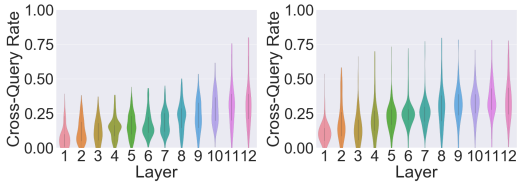
The distributions of cross-query connection rates in search sessions by Combiner and Top-2 are shown in Figure 4. Since the queries in each search session are separated by the [SEP] token, the cross-query connection rates show how many attention connections are cross these [SEP] tokens at each layer. The results show that Combiner model learns a better latent hierarchical structure than Top-2 model. The combiner prefers more intra-sentence structure at bottom layers with less

[1] https://github.com/huggingface/pytorch-transformers

[2] https://github.com/NVIDIA/apex

Table 4: The experimental results of perplexity on the PTB dataset.

Model	Perplexity
Ordered Neurons (Shen et al., 2019)	56.17
Compound PCFG (Kim et al., 2019)	83.7
Transformer-XL (Dai et al., 2019)	**54.52**
$\text{BERT}_{\text{BASE}}$ (Devlin et al., 2019)	56.84
$\text{BERT}_{\text{WIDE}}$	56.65
$\text{BERT}_{\text{LARGE}}$ (Devlin et al., 2019)	54.97
Combiner w/ Fixed Threshold (0.50)	56.32
Combiner w/ Fixed Threshold (0.55)	55.78
Combiner w/ Top-2 Pooling Per Token	55.94
Combiner w/ Top-3 Pooling Per Token	56.30
Combiner w/ One Linear Layer	55.43
Combiner w/ One Layer CNN	55.31
Combiner w/ Two Layer CNN	55.02
Combiner w/ Three Layer CNN	54.86



(a) Combiner on Search. (b) Top-2 on Search.

Figure 4: Distribution of cross-query connection rates of Combiner vs. Top-2 pooling on search sessions. X-axes represent the network layers and Y-axes indicate the cross-query connection rates.

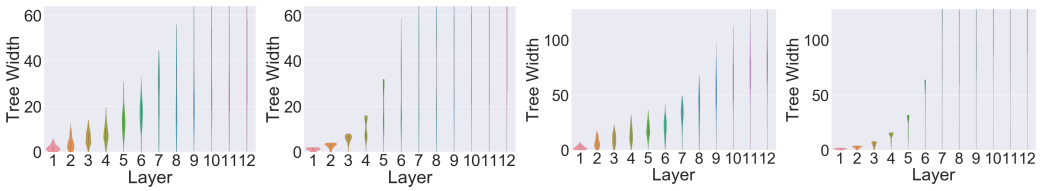
cross-query connection. When comes to higher layers, the combiner starts to learn more inter-query structure about the composition of one session.

A.5 Width of Learned Tree Structures

We show the width distributions of the learned tree structure in Figure 5. We can see the Combiner forms trees with more diverse widths at each layer, which indicates that the learned gates provide more flexibility when forming the latent tree structures. As for the Top-2 sparsity strategy, it has strong constraints on the number of connected tokens and the max width of formed trees, which can limit the performance of downstream tasks.

A.6 Case Study

We show some cases regarding the learned latent structures for search sessions and WikiText in Figure 6 and 7 separately. The qualitative results further demonstrate the effectiveness of Combiner on learning meaningful structures in an unsupervised fashion.



(a) Combiner on Session. (b) Top-2 on Session. (c) Combiner on Text. (d) Top-2 on Text.

Figure 5: Distribution of the max widths of connected trees of Combiner and Top-2 pooling on search sessions and WikiText.

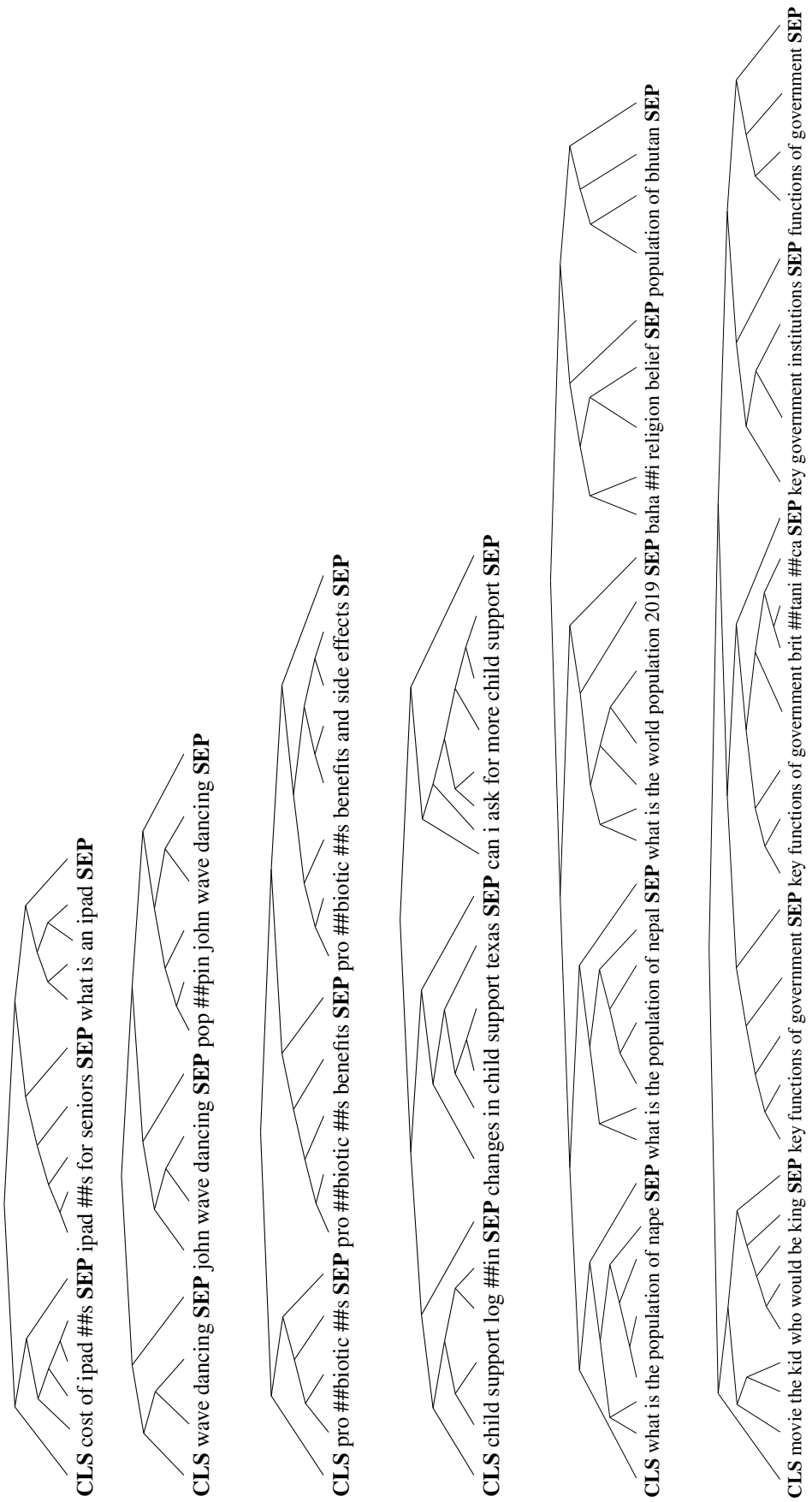


Figure 6: The latent structures learned by our Combiner model on the search session dataset.

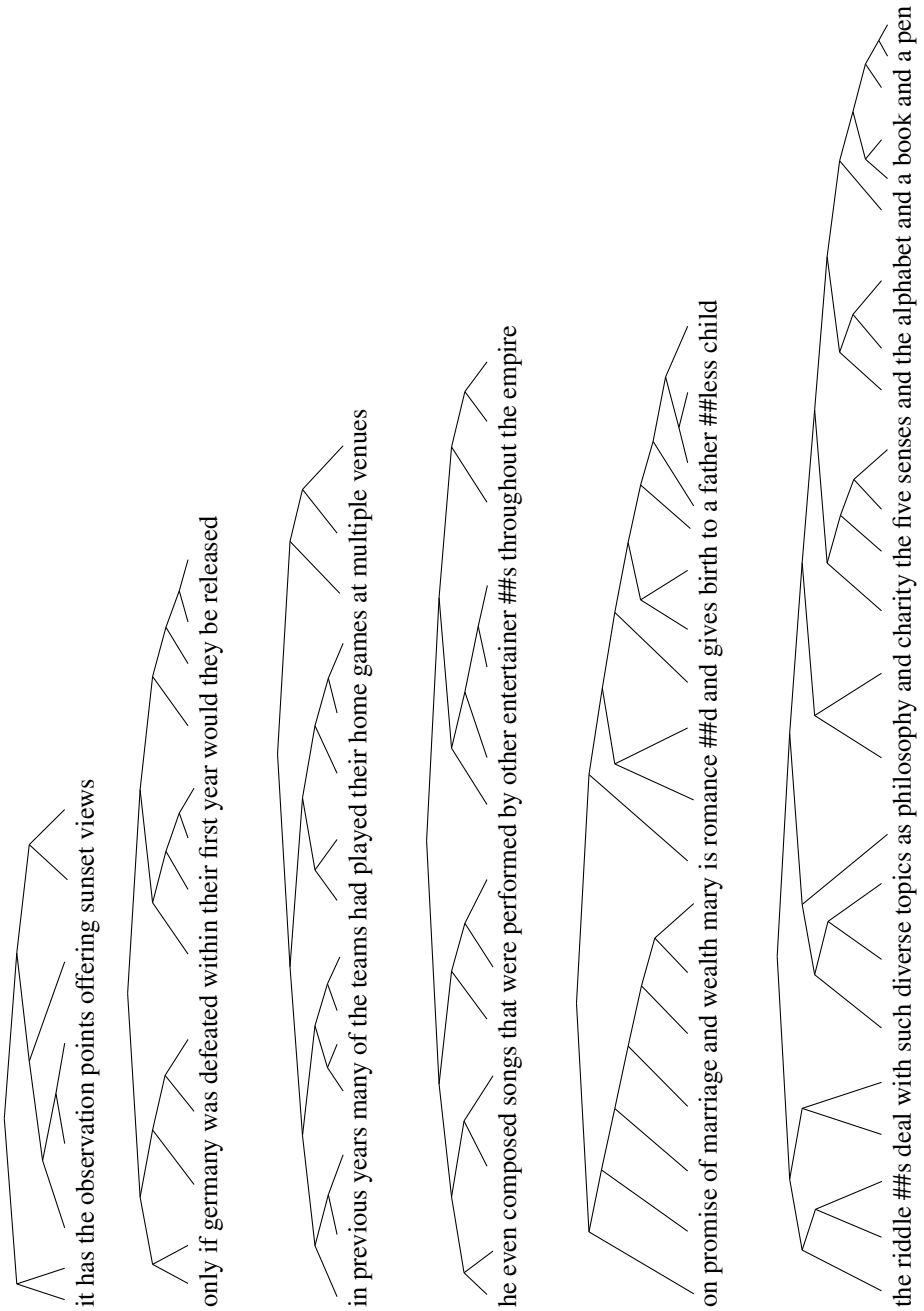


Figure 7: The latent structures learned by our Combiner model on the wikitext dataset.