Syntax-BERT: Improving Pre-trained Transformers with Syntax Trees

Jiangang Bai[1,*], **Yujing Wang**[1,2,†], **Yiren Chen**[1], **Yaming Yang**[2]
Jing Bai[2], **Jing Yu**[3], **Yunhai Tong**[1]
[1]Peking University, Beijing, China
[2]Microsoft Research Asia, Beijing, China
[3]Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
{pku_bjg,yujwang,yrchen92,yhtong}@pku.edu.cn
{yujwang,yayaming,jbai}@microsoft.com
yujing02@iie.ac.cn

Abstract

Pre-trained language models like BERT achieve superior performances in various NLP tasks without explicit consideration of syntactic information. Meanwhile, syntactic information has been proved to be crucial for the success of NLP applications. However, how to incorporate the syntax trees effectively and efficiently into pre-trained Transformers is still unsettled. In this paper, we address this problem by proposing a novel framework named Syntax-BERT. This framework works in a plug-and-play mode and is applicable to an arbitrary pre-trained checkpoint based on Transformer architecture. Experiments on various datasets of natural language understanding verify the effectiveness of syntax trees and achieve consistent improvement over multiple pre-trained models, including BERT, RoBERTa, and T5.

1 Introduction

Pre-trained language models like BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2020) and T5 (Raffel et al., 2019) become popular in recent years and achieve outstanding performances in various NLP benchmarks. These models often choose a Transformer architecture largely owing to its attractive scalability. Studies (Hewitt and Manning, 2019; Jawahar et al., 2019) have shown that a pre-trained transformer is able to capture certain syntactic information implicitly by learning from sufficient examples. However, there is still a big gap between the syntactic structures implicitly learned and the golden syntax trees created by human experts.

On the other hand, syntax tree is a useful prior for NLP-oriented neural networks (Kiperwasser and Ballesteros, 2018). For example, Tree-LSTM (Tai et al., 2015) extends the sequential architecture of LSTM to a tree-structured network. Linguistically-informed self-attention (LISA) (Strubell et al., 2018) proposes a multi-task learning framework for semantic role labeling, which incorporates syntactic knowledge into Transformer by training one attention head to be attended to its parent in a syntax tree. In addition, Nguyen et al. (2020) integrate tree-structured attention in Transformer with hierarchical accumulation guided by the syntax tree.

Although there are numerous works on syntax-enhanced LSTM and Transformer models, none of the previous works have addressed the usefulness of syntax-trees in the pre-training context. It is straight-forward to ask: *it is still helpful to leverage syntax trees explicitly in the pre-training context?* If the answer is yes, *can we ingest syntax trees into a pre-trained checkpoint efficiently without training from scratch for a specific downstream application?* This is an appealing feature in practice because pre-training from scratch is a huge waste of energy and time.

In this paper, we propose Syntax-BERT to tackle the raised questions. Unlike a standard BERT, which has a complete self-attention typology, we decompose the self-attention network into multiple sub-networks according to the tree structure. Each sub-network encapsulates one relationship from the syntax trees, including ancestor, offspring, and sibling relationships with different hops. All sub-networks share the same parameters with the pre-trained network, so they can be learned collaboratively and inherited directly from an existing checkpoint. To select the task-oriented relationships automatically, we further adopt a topical attention layer to calculate the relative importance of syntactic representations generated by different sub-networks. Finally, the customized representation is calculated by weighted summation of all

[*] The work was done when the author visited Microsoft Research Asia.

[†] Corresponding Author

Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics, pages 3011–3020
April 19 - 23, 2021. ©2021 Association for Computational Linguistics

sub-networks.

We conduct extensive experiments to verify the effectiveness of Syntax-BERT framework on various NLP tasks, including sentiment classification, natural language inference, and other tasks in the GLUE benchmark. Experimental results show that Syntax-BERT outperforms vanilla BERT models and LISA-enhanced models consistently with multiple model backbones, including BERT, RoBERTa, and T5. Specifically, it boosts the overall score of GLUE benchmark from 86.3 to 86.8 for T5-Large (Raffel et al., 2019) checkpoint, which is already trained on a huge amount of data. This improvement is convincing since only a few extra parameters are introduced to the model.

Our **major contributions** are as follows:

- To the best of our knowledge, Syntax-BERT is one of the first attempts to demonstrate the usefulness of syntax trees in pre-trained language models. It works efficiently in a plug-and-play fashion for an existing checkpoint without the need for pre-training from scratch.
- To integrate syntax trees into pre-trained Transformers, we propose a novel method that decomposes self-attention networks into different aspects and adopts topical attention for customized aggregation. As shown in the ablation study, this design benefits from syntactic structures effectively while retaining pre-trained knowledge to the largest extent.
- Syntax-BERT shows consistent improvement over multiple pre-trained backbone models with comparable model capacities. It can be combined with LISA to achieve further enhancement, indicating that these two algorithms are complementary to each other.

2 Related Work

2.1 Pre-trained language models

Recently, pre-trained language models have received significant attention from the natural language processing community. Many excellent pre-trained language models are proposed, such as BERT, RoBERTa and T5. Transformer (Vaswani et al., 2017) is a typical architecture for pre-training language models, which is based on the self-attention mechanism and is much more efficient than RNNs. BERT (Devlin et al., 2019) is a representative work that trains a large language model on the free text and then fine-tunes it on specific downstream tasks separately. BERT is pre-trained on two auxiliary pre-training tasks, Masked Language Model (MLM) and Next Sentence Prediction (NSP). RoBERTa (Liu et al., 2020) is an improved variant of BERT which utilizes dynamic masks. In RoBERTa, the NSP task is cancelled, but the full-sentence mechanism is considered. At the same time, the size of RoBERTa's training data ($\sim$160GB) is ten times the size of BERT's training data. Moreover, Raffel et al. (2019) explore the effectiveness of multiple transfer learning techniques and apply these insights at scale to create a new model T5 (Text to Text Transfer Transformer). With T5, they reform all NLP tasks into a unified text-to-text format where the input and output are always text strings. This is in contrast to BERT-style models that only output either a class label or an input span.

2.2 Syntax-aware models

Syntax is a crucial prior for NLP-oriented neural network models. Along this direction, a range of interesting approaches have been proposed, like Tree-LSTM (Tai et al., 2015), PECNN (Yang et al., 2016), SDP-LSTM (Xu et al., 2015), Supervised Treebank Conversion (Jiang et al., 2018), PRPN (Shen et al., 2018), and ON-LSTM (Shen et al., 2019).

Recent works also investigate syntactic knowledge in the context of Transformer, which are more related to this paper. For instance, Syntax-Infused Transformer (Sundararaman et al., 2019) feeds the extra syntactic features into the Transformer models explicitly, but it only considers simple syntactic features and does not provide a generic solution to incorporate tree-structured knowledge. Strubell et al. (2018) present a neural network model named LISA (Linguistically-Informed Self-Attention) that learns multi-head self-attention in a multi-task learning framework consisting of dependency parsing, part-of-speech tagging, predicate detection, and semantic role labeling. They also show that golden syntax trees can dramatically improve the performance of semantic role labeling. Moreover, Nguyen et al. (2020) propose a hierarchical accumulation approach to encode parse tree structures into self-attention mechanism. However, these approaches are designed for training a Transformer from scratch without benefiting from pre-trained checkpoints. Instead, our frame-

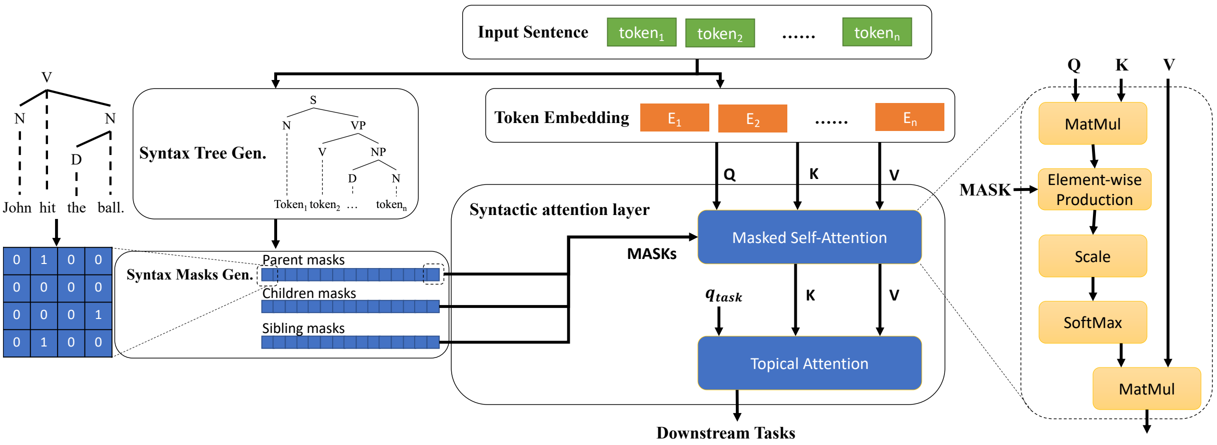

Figure 1: The Overall Architecture of Syntax-BERT. Note that the leftmost part shows an example of syntax tree and its corresponding parent syntax mask ($d = 1$).

work works in a plug-and-play mode and retains the pre-trained knowledge as much as possible for downstream applications. Concurrent to our work, Sachan et al. (2020) investigate popular strategies for incorporating dependency structures into pre-trained language models, revealing essential design decisions are necessary for strong performances. In addition, Hewitt and Manning (2019) design two sets of probes to determine whether the embedded space can be converted into syntactic information space through a linear transformation. It gives the evaluation metrics to examine how much syntactic information is included in a model.

3 Syntax-BERT

Syntax-BERT is a variant of pre-trained Transformer models, which changes the flow of information in a standard BERT network via a syntax-aware self-attention mechanism. First, the overall architecture of Syntax-BERT is presented in Section 3.1. Then, we introduce the construction of syntax trees and corresponding masks in Section 3.2. The details of syntactic attention layers will be described in Section 3.3.

3.1 Architecture

As mentioned earlier, one limitation of vanilla Transformer is that it simply uses a fully-connected topology of tokens in the pre-trained self-attention layer. Although the self-attention mechanism automatically calculates a relevance score for each token pair, it still suffers from optimization and over-fitting problems, especially when the training data is limited. Some previous works have tried to induce syntactic structure explicitly into self-attention. For instance, in Linguistically-Informed Self-Attention (LISA) (Strubell et al., 2018), syntax tree is incorporated by training one attention head to be attended to the parent of each token. However, other structural features such as siblings and children are discarded in the model. Moreover, it can not distinguish the usefulness of multiple syntactic features while largely retain the knowledge from a pre-trained checkpoint.

Syntax-BERT is designed to incorporate grammatical and syntactic knowledge as prior in the self-attention layer and support fine-grained adaptation for different downstream tasks. Specifically, it generates a bunch of sub-networks based on sparse masks reflecting different relationships and distances of tokens in a syntax tree. Intuitively, the tokens inside a sub-network often semantically related to each other, resulting in a topical representation. Therefore, we can adopt a topical attention layer to aggregate task-oriented representations from different sub-networks.

The overall architecture of Syntax-BERT is illustrated in Figure 1. As shown in the left part of this figure, we generate syntax masks for the input sentence in two steps. First, the input sentence is converted into the corresponding tree structure by a syntax parser. Second, we extract a bunch of syntax-related masks according to different features incorporated in the syntax tree. Next, the sentence is embedded similar to a standard BERT (token + positional + field embedding) and served as input to the self-attention layer. Each self-attention layer in the Syntax-BERT is composed of two kinds of attention modules, namely *masked self-attention* and *topical attention*. In a *masked* *self-attention* module, we apply syntactic masks to the fully-connected topology, generating topical sub-networks that share parameters with each other. Furthermore, the representations from different sub-networks are aggregated through a *topical attention* module so that the task-related knowledge can be distilled to the final representation vector.

3.2 Masks induced by syntax tree

Generically, a syntax tree is an ordered, rooted tree that represents the syntactic structure of a sentence according to some context-free grammar. It can be defined abstractly as $T = \{R, \mathcal{N}, \mathcal{E}\}$, where R is the root of syntax tree, $\mathcal{N}$ and $\mathcal{E}$ stands for node set and edge set respectively. The most commonly-used syntax trees are constituency trees (Chen and Manning, 2014) and dependency trees (Zhu et al., 2013), and we use both of them in our experiments unless notified.

To utilize the knowledge in a syntax tree effectively, we introduce syntax-based sub-network typologies in the self-attention layer to guide the model. Each sub-network shares the same model parameters with the global pre-trained self-attention layer, while each sub-network reflects a specific aspect of the syntax tree. This procedure can be easily implemented by multiple masks applied to the complete graph topology.

Without loss of generality, we design three categories of masks reflecting different aspects of a tree structure, namely *parent mask*, *child mask*, and *sibling mask*. For a pairwise inference task that contains a pair of sentences as input, we also apply another mask, i.e., *pairwise mask*, to capture the inter-sentence attention. Moreover, the distances between nodes (tokens) in a tree incorporate semantic relatedness. Starting from a node A, along the edges of a syntax tree, the minimum number of edges required to reach another node B can be regarded as the distance between A and B, written as $dist(A, B)$. We create fine-grained masks according to the distance between two nodes to enable customized aggregation of task-oriented knowledge.

Mathematically, a mask can be denoted by $M \in \{0,1\}^{n\times n}$, where $M_{i,j} \in \{0,1\}$ denotes if there is a connection from token i to token j, and n is the number of tokens in the current sentence.

In the **parent mask** with certain distance d, we have $M^p_{i,j,d} = 1$ if and only if the node i is the parent or ancestor of node j, at the same time $dist(i,j) = d$. Otherwise, the value will be set as zero.

In the **child mask** with certain distance d, we have $M^c_{i,j,d} = 1$ if and only if the node i is the child or offspring of node j, at the same time $dist(i,j) = d$. In other words, node j is the parent or ancestor of node i.

In the **sibling mask** with certain distance d, we have $M^s_{i,j,d} = 1$ if and only if we can find their lowest common ancestor and $dist(i,j) = d$. Note that if two nodes are in the same sentence, we can always find the lowest common ancestor, but the value should be zero if the corresponding nodes come from different sentences (in pairwise inference tasks).

The **pairwise mask** captures the interaction of multiple sentences in a pairwise inference task. We have $M^{pair}_{i,j} = 1$ if and only if both node i and j are from different sentences. we do not consider the distances in-between as the nodes are from different trees.

3.3 Syntactic attention layers

A block of Syntax-BERT contains two kinds of attention modules: *masked self-attention* and *topical attention*. The operations in a *masked self-attention* are similar to a standard self-attention except that we have sparse network connections as defined in the masks. The *masked self-attention* can be formulated as an element-wise multiplication of dot-product attention and its corresponding mask:

$$\begin{aligned}
&MaskAtt(Q,K,V,M) = \sigma(\frac{QK^\top \odot M}{\sqrt{d}})V \\
&A_{i,j} = MaskAtt(HW_i^Q, HW_i^K, HW_i^V, M_j) \\
&H_j = (A_{1,j} \oplus A_{2,j} \oplus ... \oplus A_{k,j})W^O, j \in 1, ..., m
\end{aligned} \tag{1}$$

where Q, K, V represent for the matrix of query, key and value respectively, which can be calculated by the input representation H. M represents for the matrix of syntax mask and $\odot$ denotes an operator for element-wise production; σ stands for softmax operator; $A_{i,j}$ denotes the attention-based representation obtained by the i^{th} head and j^{th} sub-network; W_i^Q, W_i^K and W_i^V represent for the parameters for linear projections; M_j denotes the mask for the j^{th}sub-network; and H_j denotes the corresponding output representation.

The output representations from different sub-networks embody knowledge from different syntactic and semantic aspects. Therefore, we leverage

Task	#Train	#Dev	#Test	#Class
SST-1	8,544	1,101	2,210	5
SST-2	6,920	873	1,822	2
SNLI	549,367	9,842	9,824	3
MNLI	392,703	9,816/9,833	9,797/9,848	3
CoLA	8,551	1,042	1,064	2
MRPC	3,669	409	1,726	2
STS-B	5,750	1,501	1,380	*
QQP	363,871	40,432	390,965	2
QNLI	104,744	5,464	5,464	2
RTE	2,491	278	3,001	2
WNLI	636	72	147	2

Table 1: Dataset Statistics: the character '/' seperate MNLI-m and MNLI-mm, '*' represents for the regression task.

another attention layer, named *topical attention* to perform a fine-grained aggregation of these representations. The most distinct part of a *topical attention* is that q_{task} is a trainable query vector for task-specific embedding. Thus, the *topical attention* layer is able to emphasize task-oriented knowledge captured by numerous sub-networks.

$$\begin{aligned} &TopicAtt(q_{task}, K, V) = \sigma(\frac{q_{task}K^{\top}}{\sqrt{d}})V \\ &H^{O} = TopicAtt(q_{task}, HW^{K}, HW^{V}) \end{aligned} \qquad (2)$$

where d denotes the size of hidden dimension, $q_{task} \in R^{1 \times d}$ is a task-related learnable query embedding vector; σ stands for the softmax operator; $H = (H_1, H_2, ..., H_m)^{\top} \in R^{m \times d}$ is the output representation collected by multiple sub-networks; W^K and W^V are parameters in the feed-forward operations; and H^O stands for the final text representation.

4 Experiments

First, we run experiments on the Stanford Sentiment Treebank (SST) dataset (Socher et al., 2013) in Section 4.1, which is designed to study the syntactic and semantic compositionality of sentiment classification. Second, in Section 4.2, we evaluate the performance of Syntax-BERT on two natural language inference datasets: SNLI and MNLI. Then, more empirical results on the GLUE benchmark and a comprehensive ablation study will be presented in Section 4.3 and 4.4 respectively. At last, we present the analysis of the structural probes in Section 4.5.

The statistics of all datasets adopted in this paper are summarized in Table 1. For each dataset, we optimize the hyper-parameters of Syntax-BERT through grid search on the validation data. Detailed settings can be found in the appendix. In our experiments, we set the maximum value of $dist(A, B)$ in a syntax tree as 15 and use both dependency and constituency trees unless specified. Thus, we have totally 90 ($15 \times 3 \times 2$) sub-networks for single-sentence tasks and 92 ($(15 \times 3 + 1) \times 2$) sub-networks for pairwise inference tasks. We adopt Transformer (Vaswani et al., 2017), BERT-Base, BERT-Large (Devlin et al., 2019), RoBERTa-Base, RoBERTa-Large (Liu et al., 2020) and T5-Large (Raffel et al., 2019) as backbone models and perform syntax-aware fine-tuning on them. We also compare with LISA (Linguistically-Informed Self-Attention) (Strubell et al., 2018), a state-of-the-art method that incorporates linguistic knowledge into self-attention operations. Specifically, LISA (Strubell et al., 2018) adopt an additional attention head to learn the syntactic dependency in the tree structure, and the parameters of this additional head are initialized randomly.

4.1 Stanford Sentiment Treebank

The SST dataset contains more than 10,000 sentences collected from movie reviews from the *rottentomatoes.com* website. The corresponding constituency trees for review sentences are contained in the dataset, where each intermediate node in a tree represents a phrase. All phrases are labeled to one of five fine-grained categories of sentiment polarity. SST-2 is a binary classification task. We follow a common setting that utilizes all phrases with lengths larger than 3 as training samples, and only full sentences will be used in the validation and testing phase. The hyper parameters for each model are selected by grid search and listed in the appendix. We compare Syntax-BERT with vanilla baselines and LISA-enhanced models. The results are listed in Table 2. As shown in the table, our model achieves 4.8 and 4.9 absolute points improvements respectively against the vanilla Transformer with comparable parameter size. By combining our framework with LISA, the results can be further boosted obviously. This indicates that our mechanism is somewhat complementary to LISA. LISA captures the syntactic information through an additional attention head, whereas our framework incorporates syntactic dependencies into original pre-trained attention heads and increases the sparsity of the network. We can see that *Syntax-Transformer + LISA* performs the best among all settings, and similar trends are demonstrated on the BERT-Base and BERT-Large checkpoints.

Model	SST-1	SST-2
Transformer	48.4	86.2
LISA-Transformer	52.2	89.1
Syntax-Transformer (Ours)	52.7	90.1
Syntax-Transformer + LISA (Ours)	**53.2**	**91.1**
BERT-Base	53.7	93.5
LISA-BERT-Base	54.2	93.7
Syntax-BERT-Base (Ours)	54.4	94.0
Syntax-BERT-Base + LISA (Ours)	**54.5**	**94.4**
BERT-Large	54.8	94.9
LISA-BERT-Large	55.0	95.9
Syntax-BERT-Large (Ours)	55.3	96.1
Syntax-BERT-Large + LISA (Ours)	**55.5**	**96.4**

Table 2: Comparison with SOTA models on SST dataset.

Model	SNLI	MNLI
Transformer	84.9	71.4
LISA-Transformer	86.1	73.7
Syntax-Transformer (Ours)	86.8	74.1
Syntax-Transformer + LISA (Ours)	**87.0**	**74.5**
BERT-Base	87.0	84.3
LISA-BERT-base	87.4	84.7
Syntax-BERT-Base (Ours)	87.7	**84.9**
Syntax-BERT-Base + LISA (Ours)	**87.8**	**84.9**
BERT-Large	88.4	86.8
LISA-BERT-Large	88.8	86.8
Syntax-BERT-Large (Ours)	88.9	86.7
Syntax-BERT-Large + LISA (Ours)	**89.0**	**87.0**

Table 3: Comparison with SOTA models on NLI datasets.

4.2 Natural Language Inference

The Natural Language Inference (NLI) task requires a model to identify the semantic relationship (entailment, contradiction, or neutral) between a premise sentence and the corresponding hypothesis sentence. In our experiments, we use two datasets for evaluation, namely SNLI (Bowman et al., 2015) and MNLI (Williams et al., 2018). We utilize the Stanford parser (Klein and Manning, 2003) to generate constituency and dependency trees for the input sentences. The MNLI dataset has two separate sets for evaluation (matched set and mismatched set), and we report the average evaluation score of these two sets.

The test accuracies on SNLI and MNLI datasets are shown in Table 3. The syntactic prior information helps the Transformer to perform much better on the NLI tasks. The accuracies on the SNLI and MNLI datasets have been improved by 1.9 and 2.7, respectively, by applying our framework to a vanilla Transformer. The LISA-enhanced transformer can also outperform vanilla transformer on NLI tasks, but the accuracy improvement is not as large as Syntax-Transformer. When the backbone model is BERT-Base or BERT-Large, consistent conclusions can be drawn from the experimental results. It is worth noting that the syntax-enhanced models for BERT-large do not show much gain based on the vanilla counterparts. This may because BERT-Large already captures sufficient knowledge for NLI tasks in the pre-training phase.

4.3 GLUE Benchmark

The GLUE benchmark (Wang et al., 2019) offers a collection of tools for evaluating the performance of models. It contains single-sentence classification tasks (CoLA and SST-2), similarity and paraphrase tasks (MRPC, QQP, and STS-B), as well as pairwise inference tasks (MNLI, RTE, and QNLI). We use the default train/dev/test split. The hyper-parameters are chosen based on the validation set (refer to the appendix for details). After the model is trained, we make predictions on the test data and send the results to GLUE online evaluation service[1] to obtain final evaluation scores.

The evaluation scores on all datasets in GLUE benchmark are illustrated in Table 4. The performances of BERT-Base, BERT-Large, RoBERTa-Base, RoBERTa-Large, and T5-Large are reproduced using the official checkpoint provided by respective authors. We only use self-contained constituency trees for the SST-2 dataset while other datasets are processed by Stanford parser[2] to extract both dependency trees and constituency trees. For a fair comparison, all results of baseline models are reproduced by our own, which are close to the reported results.

As shown in the table, syntax-enhanced models always outperform corresponding baseline models. Most notably, Syntax-RoBERTa-Base achieves an average GLUE score of 82.1, lifting 1.3 scores from a standard RoBERTa-Base with the same setting. This is impressive as only a few extra parameters are introduced to the baseline model. Particularly, the improvements on CoLA and SST-2 datasets are fairly large, showing the generalization capability of Syntax-BERT and Syntax-RoBERTa on smaller datasets. Even on T5-Large, which is trained on more data and holds more advanced performances, our approach still outperforms the base model marginally (statistically significant under 4.3 p-value using paired t-test). We can see that more training data will improve the generalization

[1] https://gluebenchmark.com

[2] https://nlp.stanford.edu/software/lex-parser.shtml

Model	Avg	CoLA	SST-2	MRPC	STS-B	QQP	MNLI-m/-mm	QNLI	RTE	WNLI
Transformer	66.1	31.3	83.9	81.7/68.6	73.6/70.2	65.6/84.4	72.3/71.4	80.3	58.0	65.1
Syntax-Transformer (Ours)	**68.8**	**36.6**	**86.4**	**81.8/69.0**	**74.0/72.3**	**65.5/84.9**	**72.5/71.2**	**81.0**	**56.7**	65.1
BERT-Base	77.4	51.7	93.5	87.2/82.1	86.7/85.4	71.1/89.0	84.3/83.7	90.4	67.2	65.1
Syntax-BERT-Base (Ours)	**78.5**	**54.1**	**94.0**	**89.2/86.0**	**88.1/86.7**	**72.0/89.6**	**84.9/84.6**	**91.1**	**68.9**	65.1
BERT-Large	80.5	60.5	94.9	89.3/85.4	87.6/86.5	72.1/89.3	**86.8/85.9**	92.7	70.1	65.1
Syntax-BERT-Large (Ours)	**81.8**	**61.9**	**96.1**	**92.0/88.9**	**89.6/88.5**	**72.4/89.5**	86.7/86.6	**92.8**	**74.7**	65.1
RoBERTa-Base	80.8	57.1	95.4	90.8/89.3	88.0/87.4	72.5/89.6	86.3/86.2	92.2	73.8	65.1
Syntax-RoBERTa-Base (Ours)	**82.1**	**63.3**	**96.1**	**91.4/88.5**	**89.9/88.3**	**73.5/88.5**	**87.8/85.7**	**94.3**	**81.2**	65.1
RoBERTa-Large	83.9	63.8	96.3	91.0/89.4	72.9/90.2	72.7/90.1	89.5/89.7	94.2	84.2	65.1
Syntax-RoBERTa-Large (Ours)	**84.7**	**64.3**	**96.9**	**92.5/90.1**	**91.6/91.4**	**73.1/89.8**	**90.2/90.0**	**94.5**	**85.0**	65.1
T5-Large	86.3	61.1	96.1	92.2/88.7	90.0/89.2	74.1/89.9	89.7/89.6	94.8	87.0	65.1
Syntax-T5-Large (Ours)	**86.8**	**62.9**	**97.2**	**92.7/90.6**	**91.3/90.7**	**74.3/90.1**	**91.2/90.5**	**95.2**	**89.6**	65.1

Table 4: Comparison with state-of-the-art models without pre-training on GLUE benchmark.

Model	SST-2	CoLA	STS-B
BERT-Large	94.9	60.5	87.6/86.5
Syntax-BERT-Large	**96.1**	**61.9**	**89.6/88.5**
w/o topical attention	95.1	61.6	88.4/87.3
w/o syntax trees	95.0	60.5	88.0/87.1
w/o dependency trees	95.6	61.4	88.7/88.1
w/o constituency trees	95.9	61.4	87.6/86.8
w/o parent masks	95.5	60.9	88.7/87.2
w/o child masks	95.3	61.2	88.3/86.8
w/o sibling masks	95.8	61.5	89.0/88.1
w/o pairwise masks	-	-	88.8/87.9

Table 5: Ablation study

Model	UUAS	Spr.
BERT-Base (Devlin et al., 2019)	79.8	0.85
Syntax-BERT-Base	**81.1**	**0.88**
BERT-Large (Devlin et al., 2019)	82.5	0.86
Syntax-BERT-Large	**83.4**	**0.90**
RoBERTa-Large (Liu et al., 2020)	83.2	0.88
Syntax-RoBERTa-Large	**84.6**	**0.93**

Table 6: The results of using Structural Probe to test whether different models contain syntactic information or not. UUAS denotes undirected attachment score, and Spr. denotes Spearman correlation.

capability of the model and compensate for the lack of syntax priors. On the other hand, syntactic information is useful in most cases, especially when training data or computation power is limited.

4.4 Ablation Study

For a comprehensive understanding of the model design, we conduct ablation study with the following settings. (1) *without topical attention*: the topical attention layer is removed, and a simple summation layer is replaced instead; (2) *without syntax tree:* all the syntactic masks generated by the syntax trees are replaced by randomly generated masks, while the parameter size of the model remains unchanged; (3) *without constituency/dependency tree*: only one kind of syntax tree is used in the model; (4) *without parent / child / sibling / pairwise masks*: the corresponding masks are removed in the implementation.

As shown in Table 5, all datasets benefit from the usage of syntactic information. Generally, parent/child masks are of more importance than the sibling masks. Moreover, the topical attention layer is crucial to the performance of Syntax-BERT model, indicating the advantage of decomposing self-attention into different sub-networks and performing fine-grained aggregation. In addition, the pairwise mask is important on STS-B dataset and shows the benefit of cross-sentence attention.

4.5 Structural Probe

Our method ingests syntax trees into the model architecture directly. To examine if the representation learned by the model also captures syntactic knowledge effectively, we follow Hewitt and Manning (2019) to reconstruct a syntax tree of the entire sentence with linear transformation learned for the embedding space. If the syntax tree can be better reconstructed, the model is viewed to learn more syntactic information. We evaluate the tree on *undirected attachment score* – the percent of undirected edges placed correctly, and *Spearman correlation* between predicted and the actual distance between each word pair in a sentence. We probe models for their ability to capture the Stanford Dependencies formalism (de Marneffe et al., 2006). As shown in Table 6, for both metrics, the syntax-aware models get better scores than corresponding baseline models, indicating that Syntax-BERT is able to incorporate more syntax information than its vanilla counterparts.

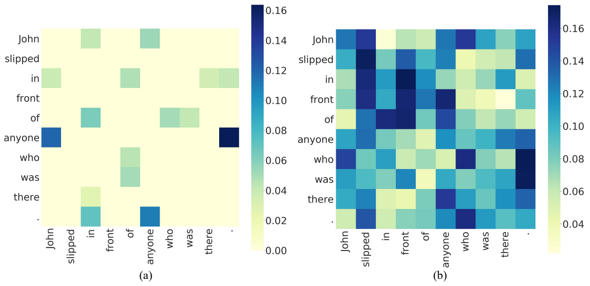

Figure 2: For an example sentence input, (a) The self-Attention scores of Syntax-Transformer corresponding to the sibling mask with $dist = 3$. (b) The self-attention scores of a vanilla Transformer.

5 Discussion

5.1 Complexity analysis

First, we choose BERT-Base as the base model to analyze the space complexity. As reported in (Devlin et al., 2019), the number of trainable parameters in BERT-Base is about 110 million. Following (Strubell et al., 2018), LISA-BERT-Base replaces one attention head in BERT-Base with a bi-affine attention head. Such an operation only adds a trainable matrix — the bi-affine transformation matrix — in each layer, which brings about 0.6 million extra parameters. Syntax-BERT-Base introduces a topical attention layer, which contains 1.0 million parameters in total for the BERT-Base version, while other parameters are inherited from vanilla BERT. Therefore, both LISA and Syntax-BERT add few parameters to the model and do not affect its original space complexity.

We now analyze the time complexity of Syntax-BERT. Assume the number of tokens in each sentence is N. First, constructing syntactic trees for each sentence and extract masking matrices can be prepossessed in the training phase or finish in $O(N^2)$ in the online inference phase. The time complexity of the embedding lookup layer is $O(N)$. Then, the attention score is calculated by $QK^\top \odot M$ with complexity $O(D_Q N^2)$, where D_Q is the dimension of Q. Assume we have M sub-networks. The complexity of masked self-attention is $O(MD_Q N^2)$. In the topical attention, the calculation process is very similar to traditional self-attention, only replacing Q with a task-related vector. So it does not change the time complexity of BERT. Finally, to get output representation, subsequent softmax and scalar-vector multiplication hold $O(D_V N)$ complexity, where D_V is the dimension of V for the topical attention. As such, the overall time complexity of Syntax-BERT is $O(N) + O(MD_Q N^2) + O(D_V N) = O(MD_Q N^2)$. When M is small, the model has the same time complexity as vanilla BERT. Moreover, as the sub-networks are usually very sparse, the time complexity can be further improved to $O(MD_Q E)$ by a sparse implementation. Here $E \ll N^2$ denotes the average number of edges in a sub-network.

5.2 Case Study

We select the sentence "John slipped in front of anyone who was there" in the CoLA dataset for case study. The task is to examine if a sentence conforms to English grammar. This sentence should be classified as negative since we use *everyone* instead of *anyone*. Syntax-Transformer classifies it correctly, but the vanilla transformer gives the wrong answer.

As visualized in Figure 2(a), the relationship between word pair *("anyone", ".")* has been highlighted in one of the sub-networks, and the corresponding topical attention score for this sub-network in Syntax-Transformer is also very high. This shows a good explainability of Syntax-Transformer by correctly identifying the error term "anyone", following a rule that "anyone" is seldom matched with the punctuation ".". However, a vanilla Transformer shows less meaningful self-attention scores, as illustrated in Figure 2(b). We give a briefing here, and please refer to the appendix for a complete description.

6 Conclusion

In this paper, we present Syntax-BERT, one of the first attempts to incorporate inductive bias of syntax trees to pre-trained Transformer models like BERT. The proposed framework can be easily plugged into an arbitrary pre-trained checkpoint, which underlines the most relevant syntactic knowledge automatically for each downstream task. We evaluate Syntax-BERT on various model backbones, including BERT, RoBERTa, and T5. The empirical results verify the effectiveness of this framework and the usefulness of syntax trees. In the future, we would like to investigate the performance of Syntax-BERT by applying it directly to the large-scale pre-training phase. Moreover, we are aiming to exploit more syntactic and semantic knowledge, including relation types from a dependency parser and concepts from a knowledge graph.

References

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *EMNLP*, pages 632–642.

Danqi Chen and Christopher D. Manning. 2014. A fast and accurate dependency parser using neural networks. In *EMNLP*.

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

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*, pages 4171–4186.

Qipeng Guo, Xipeng Qiu, Pengfei Liu, Yunfan Shao, Xiangyang Xue, and Zheng Zhang. 2019. Star-transformer. In *NAACL-HLT*, pages 1315–1325.

Nezihe Merve Gürel, Hansheng Ren, Yujing Wang, Hui Xue, Yaming Yang, and Ce Zhang. 2019. An anatomy of graph neural networks going deep via the lens of mutual information: Exponential decay vs. full preservation. *arXiv preprint arXiv:1910.04499*.

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

Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. 2019. What does BERT learn about the structure of language? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3651–3657, Florence, Italy. Association for Computational Linguistics.

Xinzhou Jiang, Zhenghua Li, Bo Zhang, Min Zhang, Sheng Li, and Luo Si. 2018. Supervised treebank conversion: Data and approaches. In *ACL (Volume 1: Long Papers)*, pages 2706–2716.

Eliyahu Kiperwasser and Miguel Ballesteros. 2018. Scheduled multi-task learning: From syntax to translation. *Transactions of the Association for Computational Linguistics*, 6:225–240.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *ACL*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Roberta: A robustly optimized bert pretraining approach.

Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *LREC*.

Xuan-Phi Nguyen, Shafiq Joty, Steven Hoi, and Richard Socher. 2020. Tree-structured attention with hierarchical accumulation. In *ICLR*.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, pages 1532–1543.

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

Devendra Singh Sachan, Yuhao Zhang, Peng Qi, and William Hamilton. 2020. Do syntax trees help pre-trained transformers extract information? *arXiv preprint arXiv:2008.09084*.

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

Yikang Shen, Shawn Tan, Alessandro Sordoni, and Aaron Courville. 2019. Ordered neurons: Integrating tree structures into recurrent neural networks. In *ICLR*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *EMNLP*, pages 1631–1642.

Emma Strubell, Pat Verga, Daniel Andor, David Weiss, and Andrew McCallum. 2018. Linguistically-informed self-attention for semantic role labeling. In *EMNLP*.

Dhanasekar Sundararaman, Vivek Subramanian, Guoyin Wang, Shijing Si, Dinghan Shen, Dong Wang, and Lawrence Carin. 2019. Syntax-infused transformer and bert models for machine translation and natural language understanding. *arXiv preprint arXiv:1911.06156*.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *ACL*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NeurIPS*, pages 5998–6008.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *ICLR*.

Adina Williams, Nikita Nangia, and Samuel R. Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *NAACL-HLT*, pages 1112–1122.

Yan Xu, Lili Mou, Ge Li, Yunchuan Chen, Hao Peng, and Zhi Jin. 2015. Classifying relations via long short term memory networks along shortest dependency paths. In *EMNLP*, pages 1785–1794.

Yunlun Yang, Yunhai Tong, Shulei Ma, and Zhi-Hong Deng. 2016. A position encoding convolutional neural network based on dependency tree for relation classification. In *EMNLP*, pages 65–74.

Muhua Zhu, Yue Zhang, Wenliang Chen, Min Zhang, and Jingbo Zhu. 2013. Fast and accurate shift-reduce constituent parsing. In *ACL*.

A Detailed settings

Here we provide detailed settings for reproduction. The open-source code will be released when this paper is officially published.

A.1 Stanford Sentiment Treebank

For raw Transformers, the number of layers is set as 12 and hidden dimension for each intermediate layer is set as 512. The probability of dropout is 0.1, and the hidden dimension of the final fully-connected layer is 2000. The word embedding vectors are initialized by GloVe (glove.840B.300d[3]) (Pennington et al., 2014) and fine-tuned during training. We use Adam optimizer with an initial learning rate 1e-4.

A.2 Natural Language Inference

For raw Transformers, we set layer number as 12, the hidden dimension of intermediate layers as 512, dropout ratio as 0.15, and the dimension of fully connected layer before Softmax activation as 2000. Learning rate is initialized as 5e-4, and Adam optimizer is used along with exponential learning rate decay of 0.9.

B Connection to GNN

A Transformer layer can be viewed as a special kind of Graph Neural Network (GNN), where each node represents for a word and all nodes construct a complete graph. To improve training speed and generalization ability, there are some previous works that advocate sparse architectures. For instance, Sparse Transformer (Child et al., 2019) separates the full self-attention operation across several steps of attention for image classification. Star-Transformer (Guo et al., 2019) sparsifies the architecture by shaping the fully-connected network into a star-shaped structure consisting of ring connections and radical connections. In the architecture of Syntax-BERT, we also introduce sparsity to the complete graph network by decomposing it into multiple sub-networks. The most salient part of our approach is that the inductive bias is designed by syntax tree, which is a crucial prior for NLP tasks. In addition, as shown previously in Table 5, a random decomposition of the network also result in moderate performance enhancement. Similar phenomena is also reported in the image classification scenario with Graph Convolutional Network (GCN) (Gürel et al., 2019).

[3] https://nlp.stanford.edu/projects/glove/