Syntax-guided Contrastive Learning for Pre-trained Language Model

Shuai Zhang, Lijie Wang, Xinyan Xiao, Hua Wu
Baidu Inc, Beijing, China
{zhangshuai28, wanglijie, xiaoxinyan, wu_hua}@baidu.com

Abstract

Syntactic information has been proved to be useful for transformer-based pre-trained language models. Previous studies often rely on additional syntax-guided attention components to enhance the transformer, which require more parameters and additional syntactic parsing in downstream tasks. This increase in complexity severely limits the application of syntax-enhanced language model in a wide range of scenarios. In order to inject syntactic knowledge effectively and efficiently into pre-trained language models, we propose a novel syntax-guided contrastive learning method which does not change the transformer architecture. Based on constituency and dependency structures of syntax trees, we design phrase-guided and tree-guided contrastive objectives, and optimize them in the pre-training stage, so as to help the pre-trained language model to capture rich syntactic knowledge in its representations. Experimental results show that our contrastive method achieves consistent improvements in a variety of tasks, including grammatical error detection, entity tasks, structural probing and GLUE. Detailed analysis further verifies that the improvements come from the utilization of syntactic information, and the learned attention weights are more explainable in terms of linguistics.

1 Introduction

Pre-trained transformer-based neural language models (LMs), such as BERT (Devlin et al., 2019) and RoBERTa (Liu et al., 2019), have achieved remarkable results in a variety of NLP tasks (Wang et al., 2018). However, many studies have found that these LMs do not encode enough syntactic knowledge in their learned representations (Wang et al., 2019; Min et al., 2020; Wang et al., 2020). As it is widely acknowledged that structural information is very important for NLP (Strubell et al., 2018; Nguyen et al., 2019; Zhang et al., 2020), there is an increasing interest in improving pre-trained LMs by using syntactic information.

Most of these works enhance pre-trained LMs by adding syntax-driven attention components to the transformer (Li et al., 2020b; Xu et al., 2020; Bai et al., 2021). They use the added components to produce a syntax-aware representation, and inject this additional representation into the original one from the vanilla transformer, so as to get a final syntax-enhanced representation. Although these works did bring improvements, the additional syntax-aware layers obviously increase application inconvenience and computation complexity, as they need to parse the input text during testing and require more neural parameters. Moreover, the performance of such explicit method depends on the parsing quality of test data (Sachan et al., 2020). There are also some efforts on incorporating syntax-related objectives into the pre-training stage, such as syntax head prediction (Wang et al., 2020) and dependency distance prediction (Xu et al., 2020). However, these predictive pre-training tasks often fail to improve performance alone and need to work together with the additional attention components (Xu et al., 2020). Overall, it is still an open challenge to effectively and efficiently incorporate syntactic information into pre-trained LMs.

In order to address the above problems, we propose Syntax-guided Contrastive Language Model (***SynCLM***). Based on contrastive learning, SynCLM uses syntactic information to create contrastive positive and negative examples, and uses them to help the pre-trained LM to learn rich syntactic knowledge through contrastive learning method. SynCLM only adds contrastive objectives in the pre-training stage, ensuring an effective and efficient utilization of syntax.

Specifically, based on constituent and dependency structures of syntax trees, we propose *phrase-guided* and *tree-guided* contrastive objectives for pre-training, as shown in Figure 1. The constituent structure represents the grouping of words into phrases within an input according to constituency

Findings of the Association for Computational Linguistics: ACL 2022, pages 2430 - 2440

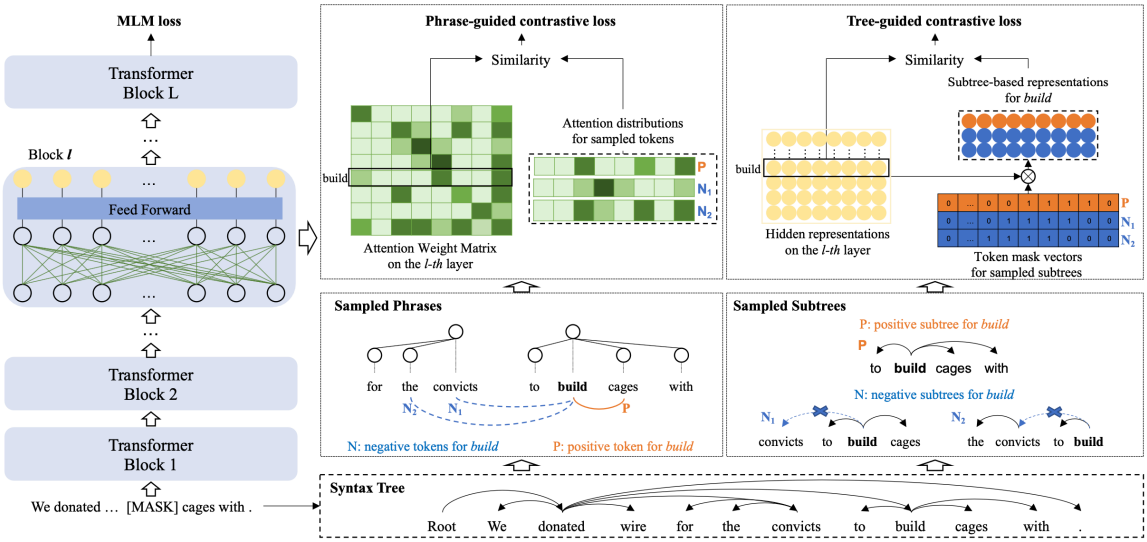

Figure 1: Overview of our pre-training framework. $\boldsymbol{P}$ and $\boldsymbol{N}_i$ represent the positive sample and the i-th negative sample, respectively. The phrase-guided contrastive objective is based on the constituent structure of inputs, focusing on using local syntactic information to guide the learning of attention distributions. The tree-guided objective is based on the dependency structure, using global syntactic information to enhance the hidden representations.

grammar (Ford and Fox, 2002). Inspired by recent studies (Mareček and Rosa, 2019; Kim et al., 2020) which prove that LM's attention heads exhibit syntactic structure akin to constituency grammar, in order to better recognize phrases from attentions, we propose the phrase-guided contrastive objective to enhance attention learning by maximizing the similarity of attention distributions between words in the same phrase. The dependency structure further encodes the binary head-dependent relations between words, and the root node aggregates the semantic information of the whole structure from all its descendant words. To make the root node attend to its descendant nodes, we propose the tree-guided contrastive objective to enhance word representations by maximizing the similarity between the representation obtained from all tokens and that obtained from syntactically related tokens. The two contrastive objectives are jointly optimized during pre-training, so as to inject syntactic knowledge into pre-trained LMs. In summary, our contributions are as following:

- We are the first to leverage the contrastive learning method to incorporate syntactic information into the pre-training stage. Our models can be directly applied to downstream tasks without introducing additional parameters and syntax parsing of inputs. In addition, our method is applicable to any arbitrary transformer-based pre-trained LM. Our code[1] will be released.
- Based on the constituency and dependency structure, we design two novel syntax-guided learning objectives to enhance the learning of attention weight distributions and hidden representations in the transformer.
- Extensive experiments show that our SynCLM achieves consistent improvements on tasks that are often used in related works, including grammatical error detection, entity-related tasks, structural probing task, and general evaluation tasks (GLUE). Detailed analysis verifies that the performance improvements come from the use of syntactic information, and the learned attention weights are more explainable in terms of linguistics.

2 Related Work

We first review studies on analyzing the linguistic knowledge learned by pre-trained LMs, and then we will introduce recent researches on incorporating linguistic knowledge into pre-trained LMs.

Linguistic Studies on Pre-trained LMs As pre-trained LMs (Devlin et al., 2019; Liu et al., 2019) continue to provide gains on NLP benchmarks, understanding what they have learned is very important, which can help us understand the reason

[1] https://github.com/PaddlePaddle/Research/tree/master/NLP/ACL2022-SynCLM

behind their success and their limitations. Many studies aim to unveil linguistic structures from the representations learned by pre-trained LMs (Jawahar et al., 2019; Wang et al., 2019). Some works demonstrate that pre-trained LMs have learned syntactic information. Hewitt and Manning (2019) indicate that syntax information is implicitly embedded in BERT by learning a linear transformation to predict the syntactic depth of each word based on its representation. Jawahar et al. (2019) and Tenney et al. (2019) show that BERT captures syntactic features at lower layers and loses some of learned syntactic information at higher layers. However, some works show that pre-trained LMs do not capture adequate syntactic knowledge. Wang et al. (2019) find that certain syntactic structures may not be embedded in BERT, as the dependency weights calculated by BERT seem to be inconsistent with human intuitions of hierarchical structures. Min et al. (2020) prove that BERT need to recruit syntactic representations from the generated syntactically informative examples to improve model performance on syntax-aware examples.

Based on these studies, we can find that pre-trained LMs often fail to encode enough syntactic information in their representations and get poor performance on syntax-aware data.

Syntax Enhanced Pre-trained LMs On the other hand, many works try to use syntax information to further improve models (Strubell et al., 2018; Nguyen et al., 2019; Zhang et al., 2020; Li et al., 2020b; Xu et al., 2020).

Task oriented works attempt to inject syntactic knowledge into the transformer (Strubell et al., 2018; Nguyen et al., 2019; Bugliarello and Okazaki, 2020; Zhang et al., 2020). In the semantic role labeling task, Strubell et al. (2018) restrict each token to attend to its syntactic parent in an attention head and improve the model performance. In the machine translation task, Nguyen et al. (2019) incorporate a tree-structured attention into the transformer for helping encode syntactic information. Bugliarello and Okazaki (2020) propose a syntax-aware self-attention mechanism to incorporate syntactic knowledge into the model. In the machine reading comprehension task, Zhang et al. (2020) use syntactic information to guide the self-attention to pay no attention to the dispensable words. These works mainly inject syntactic information into attention mechanisms, and obtain performance gains. However, they confine to a certain task.

Pre-training oriented works try to integrate syntactic information in a general way that can be applied to various NLP tasks. Inspired by the above researches, some studies (Xu et al., 2020; Li et al., 2020b; Bai et al., 2021) design various syntax-aware attention mechanisms. Despite different in detail, all of them use syntactic dependency relations to restrict the attention to important local regions. The syntax-aware attention can capture the information of important local regions according to syntactic structures, so as to obtain more benefits. Meanwhile, some works inject syntactic knowledge into pre-trained LMs via introducing new learning objectives, such as syntax head prediction (Wang et al., 2020) and dependency distance prediction (Xu et al., 2020). However, they need to work with additional syntax-guided attention methods (Xu et al., 2020).

Notably, most of these works incorporate an explicit syntax-guided component into models during testing. This increases the computational complexity and application difficulty of the model, which may limit the application of model in broader NLP tasks. In order to address these problems, we propose a novel contrastive pre-training framework to incorporate syntactic knowledge into pre-trained LMs, without introducing computational complexity in downstream tasks.

3 Methodology

In this section, we first describe the two new contrastive learning objectives in our SynCLM. Then we introduce our pre-training framework and implementation details.

3.1 Syntax-guide Contrastive Learning

In order to facilitate the learning of syntax-aware representations, we propose two learning tasks which use syntactic structures to guide the learning of attention distributions and hidden representations in the transformer. Here, we will first introduce the transformer architecture and the contrastive learning method as background. Then we will introduce our two contrastive learning objectives, and the construction of positive and negative samples, which is the main challenge of contrastive learning.

Transformer A *Transformer* (Vaswani et al., 2017) is a stack of self-attention layers where each layer (consisting of H heads) transforms the input unit into a continuous representation. Given

the input sentence S with n tokens, denoted as $\{t_1, t_2, ..., t_n\}$, we use $\mathbf{a}_i^{(l,h)}$ to represent the attention distribution of the i-th token by the h-th attention head on the l-th layer, where $1 \leq h \leq H$ and $1 \leq l \leq L$. We take the average of all heads' attention distributions on the l-th layer as the final distribution of the l-th layer, denoted as $\mathbf{a}_i^{(l,\bar{h})}$. Finally, we use $\mathbf{z}_i^l$ to represent the intermediate hidden representation of token i on the l-th layer.

Contrastive Learning Method Contrastive self-supervised learning (CSSL) (Wu et al., 2018; He et al., 2020) is a learning paradigm which aims to capture the intrinsic patterns and properties of input data without using human-provided labels. The basic idea of CSSL is to construct auxiliary tasks solely based on the input data, which is the key to CSSL, and force the network to learn meaningful representations by performing the auxiliary tasks well. The auxiliary tasks are learned by the contrastive learning loss. In this paper, we use InfoNCE function which is a variant of Noise Contrastive Estimation (NCE) (Gutmann and Hyvärinen, 2010) function for contrastive learning, as shown in Equation 1.

$$L_{cl} = -log \frac{exp(\frac{sim(q,q^+)}{\tau})}{exp(\frac{sim(q,q^+)}{\tau}) + \sum_{i=0}^{K} exp(\frac{sim(q,q_i^-)}{\tau})} \quad (1)$$

where q is the original sample; q^+ and q_i^- are the positive and the i-th negative samples, respectively; K is the number of negative samples. The $sim()$ function can be any similarity function, such as cosine, Jensen-Shannon Divergence (Endres and Schindelin, 2003) and Hellinger distance (Beran, 1977). τ called temperature coefficient is a hyper-parameter used in recent methods (Khosla et al., 2020; Yu et al., 2021).

Phrase-guided Contrastive Learning Objective Some phrases can be recognized by using the similarity of attention distributions over words (Mareček and Rosa, 2019; Kim et al., 2020). To further improve the recognition, we propose to use prior phrase structure information to further guide the learning of attention distributions by maximizing the similarity of attention distributions between words in the same phrase.

Given a sampled token t_i, we randomly select a token in the same phrase[2] as its positive example, and select K tokens outside the phrase as the contrastive negative examples. As shown in the sampled phrases of Figure 1, for the token "build", the token marked as *P* is the positive example, and tokens marked as *N* are negative examples. Then we use the contrastive learning loss (defined in Equation 1) for this learning task, and the corresponding $sim()$ function is defined as follows:

$$\begin{aligned} sim_{\text{phrase}} &= -JSD(\mathbf{a}_i^{(l,\bar{h})} \parallel \mathbf{a}_s^{(l,\bar{h})}) \\ &= -(D_{KL}(\mathbf{a}_i^{(l,\bar{h})} \parallel \mathbf{m}) + D_{KL}(\mathbf{a}_s^{(l,\bar{h})} \parallel \mathbf{m}))/2 \\ where \quad \mathbf{m} &= (\mathbf{a}_i^{(l,\bar{h})} + \mathbf{a}_s^{(l,\bar{h})})/2 \end{aligned} \quad (2)$$

where JSD is short for Jensen-Shannon Divergence (Endres and Schindelin, 2003), and D_{KL} for Kullback-Leibler Divergence (KLD) (Kullback and Leibler, 1951). The index s indicates a sampled example of token t_i, which may be positive or negative. Please note that there are many calculation choices for the $sim()$ function, such as cosine, JSD and KLD. In our early-stage preliminary experiments, we have experimented with JSD and KLD, and the former performs slightly better and thus is adopted in our framework.

Tree-guided Contrastive Learning Objective The idea that the root of a syntax tree should pay more attention to its descendant nodes has been proved to be effective in attention-based models by existing syntax-aware attention mechanisms (Xu et al., 2020; Li et al., 2020b; Bai et al., 2021). Therefore, we propose a tree-guided contrastive learning objective to maximize the similarity between the global representation based on all input tokens and the syntax-aware representation based on syntactically related tokens.

Given a sampled token t_i, we derive its subtree from the entire dependency tree. As described by the sampled subtrees in Figure 1, the subtree of token "build" consists of all tokens dominated by token "build", and "build" is the root of the subtree. We use it as the positive tree, denoted as T^+. Then we randomly replace no more than three tokens in T^+ with adjacent tokens to get the negative tree T^-, and ensure that there is at least one same token in T^+ and T^-, as shown by the other two subtrees in Figure 1. According to the above conclusion, the representation based on the tokens in the positive subtree should be closer to the original representation given by the pre-trained LM. We also use the contrastive learning loss in Equation 1 to optimize this learning objective, and the $sim()$ function is

[2] In our experiments, the sampled phrase has no more than two hierarchical layers, that is to say, the height of its corresponding subtree is no more than 2.

defined as follows:

$$sim_{\texttt{tree}} = cosine(\mathbf{z}_i^l, \sum_{t_j \in T_s} e_{ij}\mathbf{z}_j^l)$$
$$where \quad e_{ij} = \frac{exp(\mathbf{z}_i^l \cdot \mathbf{z}_j^l)}{\sum_{t_k \in T_s} exp(\mathbf{z}_i^l \cdot \mathbf{z}_k^l)} \tag{3}$$

where T_s represents a sampled subtree of token t_i, which may be positive or negative. And z_i^l represents the intermediate hidden representation of token i on the l-th layer.

3.2 Syntax-guided Pre-training Framework

We then add the two contrastive learning objectives into traditional pre-training, so as to enhance vanilla pre-trained LM. The final loss for the pre-training is the summation of the training loss for masked language model (MLM) (Devlin et al., 2019) and two new proposed tasks, as shown below.

$$L = L_{\texttt{MLM}} + L_{\texttt{phrase}} + L_{\texttt{tree}}$$

Data for Pre-training We use BERT's pre-training data (Devlin et al., 2019) as our model's pre-training data, including documents from English Wikipedia and BookCorpus (Zhu et al., 2015). Then we use the pre-processing and BPE (Byte-Pair Encoding) tokenization from RoBERTa (Liu et al., 2019) to process the training data. The maximum length of input sequence is set to 512.

To obtain syntactic structures for each sentence, we adopt a well-trained parsing model - Stanza[3] to automatically generate a syntax tree for each sentence. Because the pre-trained LM takes subwords as the input unit, for the word u, we take its first subword as the root, and add edges connecting non-first subwords to the first subword. Since syntactic information is pre-processed in advance, syntax parsing only needs to be performed once in the entire process. In our work, it takes about one day to parse the pre-training data with 20 P40 GPUs. Then, syntactic information is used as the additional input in the pre-training stage.

Implementation Details To accelerate the training process, we initialize parameters from RoBERTa models[4] released by Liu et al. (2019). We use RoBERTa-base and RoBERTa-large to initialize our base and large models respectively. RoBERTa-base contains 12 layers, each of which has 12 heads and 768 hidden states. And RoBERTa-large contains 24 layers, each of which has 16 heads and 1024 hidden states. We set l as the last hidden layer in Equation 2 and Equation 3. And the number of negative examples is set to 3. As our pre-trained LMs do not introduce additional parameters, the parameter sizes of our base and large models are the same as those of RoBERTa models.

We pre-train our models with 16 32G NVIDIA V100 GPUs. The base model takes about four days and the large model takes about seven days. During the training process, in order to choose a well pre-trained model, we evaluate the intermediate model per 10K steps, and terminate the training when the performance alteration (i.e., Perplexity of LMs) is below a certain threshold for five sequential evaluations. In the base setting, the batch size is 512, and the total steps are 300,000, 24,000 of which is the warm up steps. For the large model, the batch size is 256, and the total steps are 350,000, 30,000 of which is for warming up.

Dataset	Train	Test	Class	Metric
CoLA	8,551	1,063	2	MCC
BLiMP	0	40,000	*	Acc
FCE	28,731	2,720	2/*	Acc/$F_{0.5}$
CoNLL-2003	14,041	3,453	*	F1
OpenEntity	1,988	1,988	9	F1
SST-2	63,749	1,821	2	Acc
MRPC	3,668	1,725	2	Acc/F1
QQP	363,871	390,695	2	Acc/F1
STS-B	5,749	1,379	*	Pea./Spr.
MNLI	392,702	9,796	3	Acc
QNLI	104,743	5,463	2	Acc
RTE	2,490	3,000	2	Acc

Table 1: Statistics of datasets used in our work. "*" represents for the non-classification tasks. "Acc" is short for "accuracy". "Pea." and "Spr." are abbreviations for "Pearson" and "Spearman correlation" respectively.

4 Experiments

First, we verify the effectiveness of SynCLM on several syntax-aware tasks, including grammatical error detection task (Section 4.1) and entity tasks (Section 4.2), which are often used for testing syntax pre-training models. Then, we test the effectiveness of SynCLM on more general tasks by using GLUE benchmark (Section 4.3). At last, detailed analysis is conducted to show the impact of incorporating syntactic knowledge (Section 4.4).

Please note that ↑ in our reported results means statistically significant improvement over the baseline with $\texttt{p-value} < 0.05$. Besides, for fair comparison, we report continue training results

[3] https://github.com/stanfordnlp/stanza

[4] https://github.com/pytorch/fairseq/tree/master/examples/roberta

Models	**BLiMP** 1P/2P Acc	**CoLA** MCC	**FCE** Acc/$F_{0.5}$
BERT-large (Devlin)	-/-	63.9*	-/57.3*
BiLSTM-Joint (Rei)	-/-	-	80.1/52.1
SLA-large (Li20)	-/-	64.5	-/58.0
GPT-2 large (Rad19)	78.0/81.6	-	-/-
RoBERTa-base (Liu)	74.9/78.5	63.6	83.3/68.6
+ continuous	75.0/79.6	63.8	83.5/68.6
+ PHRASE	75.5/81.2	64.5	83.9/69.0
+ TREE	76.4/80.6	64.9	84.2/68.9
SynCLM-base	**77.3**$^\uparrow$**/81.0**$^\uparrow$	**65.3**$^\uparrow$	**84.3**$^\uparrow$**69.2**$^\uparrow$
RoBERTa-base + SLA	-/-	64.2	83.2/68.3
+ PHRASE	-/-	65.1	83.7/67.3
+ TREE	-/-	65.8	84.3/68.4
SynCLM-base + SLA	-/-	**66.3**$^\uparrow$	83.6/**68.7**
RoBERTa-large (Liu)	77.3/79.4	68.0	85.3/72.2
SynCLM^g	**79.5**$^\uparrow$**/81.1**$^\uparrow$	**69.3**$^\uparrow$	**86.1**$^\uparrow$**/72.4**

Table 2: Results on GED datasets. Results with "$\star$" are taken from Li et al. (2020b). Reported results of CoLA are a median over 5 runs, and those of FCE are the average over 5 runs. For BLiMP, we report accuracies for "one prefix" (1P) (Linzen et al., 2016) and "two prefix" (2P) (Wilcox et al., 2019).

(*continuous*) of RoBERTa[5].

The statistics of datasets adopted in this paper are summarized in Table 1. For datasets of GLUE, we use metrics reported in Devlin et al. (2019). For other datasets, we use popular metrics provided by dataset authors and other researchers.

4.1 Grammatical Error Detection (GED)

GED task aims to evaluate the grammatical acceptability of a given sentence. We use three popular public datasets, i.e., CoLA (Warstadt et al., 2019), BLiMP (Warstadt et al., 2020), and FCE (Yannakoudakis et al., 2011), to evaluate our models. For CoLA, we use Matthews Correlation Coefficient (MCC) (Matthews, 1975) as the evaluation metric. For BLiMP, we evaluate models using the overall accuracy on all input pairs, namely the proportion of pairs whose acceptable sentence is assigned a higher probability. On FCE, following Rei and Søgaard (2019), we take it as a binary classification task and a sequence labeling task, and use accuracy and $F_{0.5}$ to evaluate them respectively.

Baselines Rei and Søgaard (2019) combine objectives at different granularities (i.e., sentence and token) to learn better representations. Li et al. (2020b) use dependency distance matrix to obtain a syntax-aware local attention (SLA) and achieve SOTA results on FCE. We also report the results of BERT, RoBERTa and GPT-2 (Radford et al.), where GPT-2 reports SOTA results on BLiMP.

Main Results From Table 2, it can be seen that SynCLM achieves consistent gains over RoBERTa on all three datasets: 2.0% higher average accuracy on BLiMP, 1.3% higher MCC on CoLA, and 0.8% higher accuracy on FCE. The results show that syntactic prior information helps SynCLM to perform much better on GED task. We believe this is because the grammatical acceptability of a sentence strongly rely on its syntactic structure. As illustrated by the first example in Figure 2, which checks the morphological number agreement of the sentence, the morphological number of the word "*eat*" should be consistent with that of its subject "*John*". And the dependency syntax illustrates the subject-verb relation between them.

The tree-guided method performs better than the phrase-guided method on most metrics, as the tree structure gives the head-dependent relations between words more directly and more explicitly. Moreover, combining the two methods can achieve more gains.

Merging SLA We also test whether SynCLM can be further improved with previous syntax-enhanced attention mechanisms for fine-tuning. We implement SLA (Li et al., 2020b) in the fine-tuning stage, and show the results in the third part of Table 2. It can be seen that merging SLA and SynCLM can achieve more gains on CoLA and FCE, which means that SynCLM can be further improved by using syntax information during fine-tuning.

4.2 Entity Tasks

We evaluate SynCLM on two entity related tasks: named entity recognition (NER) and entity typing (ENT), which aim to recognize entities and predict entity types respectively. We use CoNLL-2003 (Sang and De Meulder, 2003) for NER task and OpenEntity (Choi et al., 2018) for ENT task.

Baselines BERT-MRC (Li et al., 2020a) formulates NER task as a machine reading comprehension task to handle both flat and nested NER tasks. KEPLER (Wang et al., 2021) infuses knowledge into pre-trained models and jointly learns knowledge embeddings and language representations. SEPREM (Xu et al., 2020) injects syntax information into pre-trained LMs by introducing two learning tasks and a syntax-aware attention layer. LUKE (Yamada et al., 2020) uses a large amount of entity-annotated corpus and an entity-aware self-

[5]Due to the limitation of space and computing resources, we only give continue training results of base models.

Models	QQP	MRPC	STS	SST	CoLA	RTE	MNLI-m	QNLI
BERT-large (Devlin et al., 2019)	91.3/-	88.0/-	90.0/-	93.2	60.6	70.4	86.6	92.3
XLNet-large (Yang et al., 2019)	92.3/-	90.8/-	92.5/-	97.0	69.0	85.9	90.8	94.9
SLA-large (Li et al., 2020b)	-/-	-/-	-/-	94.3	64.5	-	-	-
RoBERTa-base (Liu et al., 2019)	91.6/88.9*	90.1/92.7*	90.9/90.7*	94.8	63.6	78.7	87.6	92.8
+ continuous	91.6/88.8	90.2/92.8	90.2/90.1	94.9	63.8	79.1	87.2	92.8
+ PHRASE	91.7/88.9	90.5/93.0	90.3/90.2	95.2	64.5	79.8	87.0	92.9
+ TREE	91.7/88.9	91.2/93.6	90.6/90.4	95.1	64.9	80.1	87.4	93.0
SynCLM-base	91.7/88.9	**91.4**$^\uparrow$**/93.7**$^\uparrow$	90.8/90.6	**95.1**$^\uparrow$	**65.3**$^\uparrow$	**80.1**$^\uparrow$	87.2	**93.0**
RoBERTa-large (Liu et al., 2019)	92.1/89.5*	90.7/93.2*	92.2/92.1*	96.4	68.0	86.3*	90.2	94.7
SynCLM-large	**92.3/89.7**	**91.2**$^\uparrow$**/93.6**$^\uparrow$	92.0/91.8	**96.7**$^\uparrow$	**69.3**$^\uparrow$	**87.4**$^\uparrow$	**90.5**	**94.8**

Table 3: Performance on dev sets of GLUE tasks. The results of BERT and RoBERTa are from Liu et al. (2019). Results with * are from our re-implementations, as some metrics are not given by Liu et al. (2019). For each task, we run model for 5 times with different random initialization seeds, and report the median result.

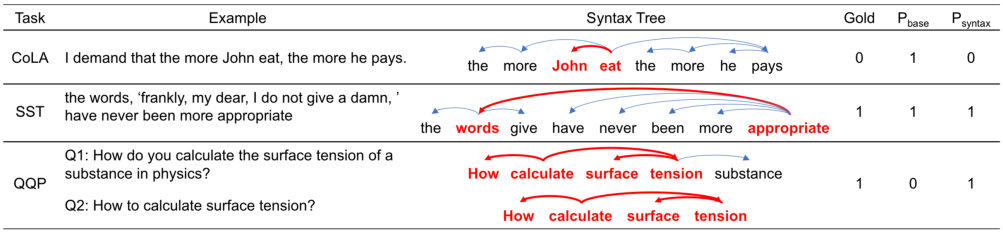

Figure 2: Case study. The third column shows the simplified syntax tree of each example. $P_{\texttt{base}}$ represents the label predicted by RoBERTa-base model, and $P_{\texttt{syntax}}$ represents the label predicted by SynCLM-base.

Models	**CoNLL-2003** P / R / F1	**OpenEntity** P / R / F1
BERT-MRC (Li)	92.3 / 94.6 / 93.0	- / - / -
SLA-large (Li20)	92.3 / 93.4 / 92.9	- / - / -
KEPLER (Wang)	- / - / -	77.2 / 74.2 / 75.7
SEPREM (Xu20)	- / - / -	80.1 / 77.1 / 79.1
LUKE (Yamada)	- / - / 94.3	79.9 / 76.6 / 78.2
SEPREM-base	84.0 / 92.9 /88.2	**76.7** / 73.5 / 75.1
RoBERTa-base	92.4 / 92.9 / 92.6	75.7 / 74.6 / 75.1
+ PHRASE	92.9 / 93.0 / **93.0**	75.9 / 74.9 / 75.4
+ TREE	92.7 / 92.9 / 92.8	75.6 / 75.2 / 75.4
SynCLM-base	**93.0**$^\uparrow$ / **93.0**$^\uparrow$ / **93.0**$^\uparrow$	**76.6** / 74.6 / **75.6**$^\uparrow$
+ SLA	92.1 / **94.1** / **93.1**$^\uparrow$	**76.7** / 74.6 / **75.7**$^\uparrow$
RoBERTa-large	93.0 / 93.5 / 93.2	76.3 / 76.1 / 76.2
SynCLM-large	**93.4**$^\uparrow$ / **93.8**$^\uparrow$ / **93.6**$^\uparrow$	**76.8** / 76.1 / **76.4**

Table 4: Results (average of 5 runs) on entity tasks. We report continue training results for RoBERTa-base.

attention mechanism to learn pre-trained contextualized representations for words and entities, and obtains SOTA results on five entity-related datasets, including CoNLL-2003 and OpenEntity.

Main Results Table 4 shows the performances of SOTA models and our models on CoNLL-2003 and OpenEntity. On the NER task, SynCLM-large improves F1 score by 0.4% compared with RoBERTa-large. Meanwhile, phrase-guided method consistently outperforms tree-guided method both in the base and large model. We think this is because the goal of phrase-guided method matches pretty well with the goal of NER. On the ENT task, SynCLM obtains 0.9% and 0.5% precision improvements under the settings of base-size and large-size, respectively.

Comparison with SEPREM and SLA Compared with SEPREM on ENT, SynCLM achieves a smaller improvement. We suspect the reason is two-fold. First, in SynCLM, syntactic information is incorporated only in the pre-training stage. Second, the pre-training data used in SEPREM is about ten times larger than ours. In order to verify the above hypotheses, based on RoBERTa-base, we implement the two pre-training tasks of SEPREM, namely dependency head prediction and dependency distance prediction, and train them on the pre-training data used in our work, resulting in *SEPREM-base* in Table 4. We observe that SEPREM trained on small-scale data does not perform well. Meanwhile, we merge SLA in the fine-tuning stage, resulting in *SynCLM-base + SLA*. The result verifies that SynCLM can be further improved by using syntactic information during fine-tuning stage.

4.3 GLUE Benchmark

Besides, we evaluate SynCLM on the GLUE benchmark (Wang et al., 2018), a collection of diverse

datasets for evaluating natural language understanding 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. For each dataset, we fine-tune the pre-trained model separately, using only the corresponding single-task training data (i.e., without multi-task training). Our fine-tuning procedure follows the original RoBERTa paper. We consider a limited hyperparameter sweep for each task, with batch sizes $\in \{16, 32\}$ and learning rates $\in \{1e-5, 2e-5, 3e-5\}$, with a linear warm up for the first 6% of steps followed by a linear decay to 0. We fine-tune for 10 epochs. The rest of the hyperparameters remain the same as during pre-training.

Experimental Results As shown in Table 3, our models outperform baseline models in most tasks, and achieve more significant gains in tasks with small training datasets, such as CoLA, RTE, MRPC. The performance on CoLA is discussed in Section 4.1. On RTE, compared with baseline models, SynCLM obtains significant gains of 1.4% and 1.1% in base size and large size, respectively. Similarly, it brings accuracy improvement of 1.3% for base model and 0.5% for large model on MRPC. Moreover, incorporating syntactic knowledge into base models brings greater improvements in some datasets. From the results of all downstream tasks, it can be seen that syntactic information is more useful when task's training data is small or the computation power is limited. We think that more training data in the fine-tuning stage will lead to greater loss of syntactic knowledge encoded in the last layer's hidden representations, as last two layers encode task-specific features and undergo the largest changes (Kovaleva et al., 2019).

Besides, SynCLM achieves larger improvements on single-sentence tasks, but does not always perform well on sentence-pair tasks. We think this is because the cross-sentence interactions are more important for sentence-pair tasks. How to use syntactic information effectively in the sentence-pair tasks is a problem we plan to explore in the future.

Finally, we can conclude that SynCLM is still effective in general tasks, especially in tasks with small training data.

Models	UUAS	Spr.
BERT-large (Devlin)	82.5	0.86
Syntax-BERT-large (Bai21)	83.4	0.90
Syntax-RoBERTa-large (Bai21)	84.6	0.93
RoBERTa-base (Liu)	81.2	0.85
SynCLM-base	**84.9**↑	0.87

Table 5: The results of structural probing task.

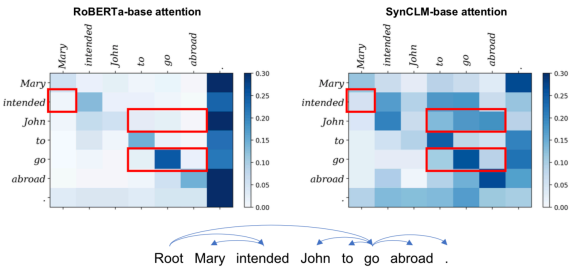

Figure 3: Visualization of attention weight scores. This case is from CoLA and has grammatical error. The red rectangle indicates higher scores in our model but lower scores in the baseline model.

4.4 Analysis

Structural Probing Tasks To check whether the representation learned by SynCLM captures syntactic knowledge effectively, following Hewitt and Manning (2019), we construct a syntax tree of a sentence with linear transformation learned for the embedding space. If the syntax tree is better constructed, the model is considered having learned more syntactic information. We use the pre-trained LM based on phrase-guided method to capture the Stanford Dependencies formalism (De Marneffe et al., 2006). Similarly, we use the *undirected attachment score* (UUAS) denoting the percentage of correctly placed undirected edges as the main metric. We also report *spearman correlation* (Spr.) between predicted and the actual distance between each token pair in a sentence. The results are shown in Table 5. It can be seen that our base model obtains SOTA results on UUAS, indicating that our method can enhance the model capability of capturing syntactic structures.

Case Study Figure 2 gives three examples to illustrate the effectiveness of incorporating syntactic information. These examples show that SynCLM can capture syntactic information and make correct predictions based on the obtained information. To give further insight into how syntactic knowledge affects prediction, we highlight the main syntax structures that affect prediction. Here, we take the third case for detailed analysis. The core tokens in

the syntax trees of the two sentences are the same, so the model predicts they have the same semantics. Through more data analysis, we find that SynCLM enhances the attention weight between syntactically related words, thus increasing the importance of non-leaf tokens in model prediction. Please note that this feature also leads to wrong predictions. In the future we will attend to the problem of how to integrate syntactic and semantic information in model prediction.

Attention Visualization In order to verify the impact of syntactic information in the attention mechanisms of the pre-trained LM, we plot attention weights of baseline models and our models in Figure 3. We mainly focus on the interactions of tokens, except for [CLS] and [SEP]. Then the attention weights are averaged over all heads and layers. This visualization demonstrates the effectiveness of injecting syntactic information into self-attention. From Figure 3, we can see that higher attention weight between directly syntactically related tokens in our model. For example, our model assigns strong attentions from the token "John" to "go" and "abroad", while the baseline model assigns lower attentions for these correlated tokens.

5 Conclusion

To the best of our knowledge, this is the first work of leveraging contrasting learning to inject syntax knowledge into pre-trained LMs. Motivated by the properties of constituent and dependency structures of syntax, we design phrase-guided and tree-guided learning objectives to guide the learning of attention distributions and hidden representations in the transformer. Through extensive experiments, we show that SynCLM consistently improves a wide range of tasks, from GED, entity tasks to GLEU, which confirms the advantage of our syntax-guided contrastive learning. Detailed analysis also shows that SynCLM does incorporate rich syntax knowledge and learn explainable attention weights.

Acknowledgements

We are very grateful to our anonymous reviewers for their helpful feedback on this work. We would like to thank Ying Chen for examination and revision in paper writing; Can Gao for the help on model training. This work was supported in part by the National Key R&D Program of China under Grant 2020YFB1406701.

References

Jiangang Bai, Yujing Wang, Yiren Chen, Yaming Yang, Jing Bai, Jing Yu, and Yunhai Tong. 2021. Syntax-bert: Improving pre-trained transformers with syntax trees. arXiv preprint arXiv:2103.04350.

Rudolf Beran. 1977. Minimum hellinger distance estimates for parametric models. The annals of Statistics, pages 445–463.

Emanuele Bugliarello and Naoaki Okazaki. 2020. Enhancing machine translation with dependency-aware self-attention. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 1618–1627.

Eunsol Choi, Omer Levy, Yejin Choi, and Luke Zettlemoyer. 2018. Ultra-fine entity typing. In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 87–96.

Marie-Catherine De Marneffe, Bill MacCartney, Christopher D Manning, et al. 2006. Generating typed dependency parses from phrase structure parses. In Lrec, volume 6, pages 449–454.

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

Dominik Maria Endres and Johannes E Schindelin. 2003. A new metric for probability distributions. IEEE Transactions on Information theory, 49(7):1858–1860.

Cecilia E Ford and Barbara A Fox. 2002. Constituency and the grammar. The language of turn and sequence, page 14.

Michael Gutmann and Aapo Hyvärinen. 2010. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In Proceedings of the thirteenth international conference on artificial intelligence and statistics, pages 297–304. JMLR Workshop and Conference Proceedings.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. 2020. Momentum contrast for unsupervised visual representation learning. In 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 9726–9735. IEEE Computer Society.

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

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

Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. 2020. Supervised contrastive learning. Advances in Neural Information Processing Systems, 33.

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

Olga Kovaleva, Alexey Romanov, Anna Rogers, and Anna Rumshisky. 2019. Revealing the dark secrets of bert. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pages 4365–4374.

Solomon Kullback and Richard A Leibler. 1951. On information and sufficiency. The annals of mathematical statistics, 22(1):79–86.

Xiaoya Li, Jingrong Feng, Yuxian Meng, Qinghong Han, Fei Wu, and Jiwei Li. 2020a. A unified mrc framework for named entity recognition. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 5849–5859.

Zhongli Li, Qingyu Zhou, Chao Li, Ke Xu, and Yunbo Cao. 2020b. Improving bert with syntax-aware local attention. arXiv preprint arXiv:2012.15150.

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. 2016. Assessing the ability of lstms to learn syntax-sensitive dependencies. Transactions of the Association for Computational Linguistics, 4:521–535.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. arXiv preprint arXiv:1907.11692.

David Mareček and Rudolf Rosa. 2019. From balustrades to pierre vinken: Looking for syntax in transformer self-attentions. In Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP, pages 263–275.

Brian W Matthews. 1975. Comparison of the predicted and observed secondary structure of t4 phage lysozyme. Biochimica et Biophysica Acta (BBA)-Protein Structure, 405(2):442–451.

Junghyun Min, R Thomas McCoy, Dipanjan Das, Emily Pitler, and Tal Linzen. 2020. Syntactic data augmentation increases robustness to inference heuristics. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 2339–2352.

Xuan-Phi Nguyen, Shafiq Joty, Steven Hoi, and Richard Socher. 2019. Tree-structured attention with hierarchical accumulation. In International Conference on Learning Representations.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners.

Marek Rei and Anders Søgaard. 2019. Jointly learning to label sentences and tokens. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 33, pages 6916–6923.

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.

Erik Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003, pages 142–147.

Emma Strubell, Patrick Verga, Daniel Andor, David Weiss, and Andrew McCallum. 2018. Linguistically-informed self-attention for semantic role labeling. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 5027–5038.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019. Bert rediscovers the classical nlp pipeline. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 4593–4601.

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 Advances in neural information processing systems, pages 5998–6008.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. In Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP, pages 353–355.

Ruize Wang, Duyu Tang, Nan Duan, Zhongyu Wei, Xuanjing Huang, Cuihong Cao, Daxin Jiang, Ming Zhou, et al. 2020. K-adapter: Infusing knowledge into pre-trained models with adapters. arXiv preprint arXiv:2002.01808.

Xiaozhi Wang, Tianyu Gao, Zhaocheng Zhu, Zhengyan Zhang, Zhiyuan Liu, Juanzi Li, and Jian Tang. 2021. Kepler: A unified model for knowledge embedding and pre-trained language representation.

Transactions of the Association for Computational Linguistics, 9:176–194.

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

Alex Warstadt, Alicia Parrish, Haokun Liu, Anhad Mohananey, Wei Peng, Sheng-Fu Wang, and Samuel R Bowman. 2020. Blimp: The benchmark of linguistic minimal pairs for english. Transactions of the Association for Computational Linguistics, 8:377–392.

Alex Warstadt, Amanpreet Singh, and Samuel R Bowman. 2019. Neural network acceptability judgments. Transactions of the Association for Computational Linguistics, 7:625–641.

Ethan Wilcox, Peng Qian, Richard Futrell, Miguel Ballesteros, and Roger Levy. 2019. Structural supervision improves learning of non-local grammatical dependencies. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pages 3302–3312.

Zhirong Wu, Yuanjun Xiong, X Yu Stella, and Dahua Lin. 2018. Unsupervised feature learning via non-parametric instance discrimination. In 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 3733–3742. IEEE.

Zenan Xu, Daya Guo, Duyu Tang, Qinliang Su, Linjun Shou, Ming Gong, Wanjun Zhong, Xiaojun Quan, Nan Duan, and Daxin Jiang. 2020. Syntax-enhanced pre-trained model. arXiv preprint arXiv:2012.14116.

Ikuya Yamada, Akari Asai, Hiroyuki Shindo, Hideaki Takeda, and Yuji Matsumoto. 2020. Luke: Deep contextualized entity representations with entity-aware self-attention. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 6442–6454.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. Advances in neural information processing systems, 32.

Helen Yannakoudakis, Ted Briscoe, and Ben Medlock. 2011. A new dataset and method for automatically grading esol texts. In Proceedings of the 49th annual meeting of the association for computational linguistics: human language technologies, pages 180–189.

Yue Yu, Simiao Zuo, Haoming Jiang, Wendi Ren, Tuo Zhao, and Chao Zhang. 2021. Fine-tuning pre-trained language model with weak supervision: A contrastive-regularized self-training approach. In Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 1063–1077.

Zhuosheng Zhang, Yuwei Wu, Junru Zhou, Sufeng Duan, Hai Zhao, and Rui Wang. 2020. Sg-net: Syntax-guided machine reading comprehension. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 34, pages 9636–9643.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In Proceedings of the IEEE international conference on computer vision, pages 19–27.