On the utility of enhancing BERT syntactic bias with Token Reordering **Pretraining**

Yassir El Mesbahi†∗ Atif Mahmud²†∗ Abbas Ghaddar¹♠∗ Mehdi Rezagholizadeh¹ Philippe Langlais³ Prasanna Parthasarathi¹ ¹ Huawei Noah's Ark Lab

² David R. Cheriton School of Computer Science, University of Waterloo

³ RALI/DIRO, Université de Montréal, Canada

abbas.ghaddar@huawei.com

Abstract

Self-supervised Language Modelling (LM) objectives —like BERT masked LM— have become the default choice for pretraining language models. TOken Reordering (TOR) pretraining objectives, beyond *token prediction*[1](#page-0-0) , have not been extensively studied yet. In this work, we explore challenges that underlie the development and usefulness of such objectives on downstream language tasks. In particular, we design a novel TOR pretraining objective which predicts whether two tokens are adjacent or not given a *partial bag-of-tokens* input. In addition, we investigate the usefulness of Graph Isomorphism Network (GIN), when placed on top of the BERT encoder, in order to enhance the overall model ability to leverage topological signal from the encoded representations. We compare language understanding abilities of TOR to the one of MLM on word-order sensitive (e.g. Dependency Parsing) and insensitive (e.g. text classification) tasks in both full training and few-shot settings. Our results indicate that TOR is competitive to MLM on the GLUE language understanding benchmark, and slightly superior on syntax-dependent datasets, especially in the few-shot setting.

1 Introduction

Pretraining with self-supervised language modelling objectives [\(Devlin et al.,](#page-9-0) [2019;](#page-9-0) [Radford et al.,](#page-10-0) [2019;](#page-10-0) [Yang et al.,](#page-11-0) [2019;](#page-11-0) [Clark et al.,](#page-8-0) [2019;](#page-8-0) [Song](#page-11-1) [et al.,](#page-11-1) [2019\)](#page-11-1) has become indispensable for stateof-the-art performances on Natural Language Understanding (NLU) benchmarks [\(Rajpurkar et al.,](#page-10-1) [2018;](#page-10-1) [Wang et al.,](#page-11-2) [2018,](#page-11-2) [2019a;](#page-11-3) [Hu et al.,](#page-9-1) [2020\)](#page-9-1). Identifying the mechanisms those models use for task solving gained prominence [\(Tenney et al.,](#page-11-4) [2019;](#page-11-4) [Goldberg,](#page-9-2) [2019;](#page-9-2) [Kulmizev and Nivre,](#page-9-3) [2021;](#page-9-3) [Kazemnejad et al.,](#page-9-4) [2023\)](#page-9-4). Such works attempted to shed light on whether Pretrained Language Models (PLMs) [\(Liu et al.,](#page-9-5) [2019;](#page-9-5) [Brown et al.,](#page-8-1) [2020a;](#page-8-1) [Conneau et al.,](#page-8-2) [2020;](#page-8-2) [Raffel et al.,](#page-10-2) [2019\)](#page-10-2) learn to encode language through appropriate inductive biases that align with the human understanding of syntax in languages. Models not demonstrating this behavior suggest that existing pretraining objectives (like MLM [\(Devlin et al.,](#page-9-0) [2019\)](#page-9-0) and its variants) may not be sufficient at encoding the essential aspects of syntax that potentially guide language understanding [\(Sinha et al.,](#page-10-3) [2021a](#page-10-3)[,b;](#page-10-4) [Alajrami and Aletras,](#page-8-3) [2022\)](#page-8-3).

Figure 1: Illustration of input and target of the MLM (left) and TOR (right) pretraining objectives. Green solid and yellow dotted boxes indicate token and position indexes respectively. $x_{[M]}$ and $p_{[M]}$ indicate a randomly masked token and position respectively, while transparent targets are ignored during loss calculation. The target of TOR is a matrix that point to neighbor token at distance k (+1 in this example).

Order of tokens being an essential artifact to capture syntactic cues, we propose TOken Reordering (TOR), a novel self-supervised task that boosts the awareness to word-order in models. Figure [1](#page-0-1) shows the difference between MLM [\(Devlin et al.,](#page-9-0) [2019\)](#page-9-0) and TOR objectives, where in pretraining with MLM some input tokens are masked and the model is tasked with predicting the masked tokens.

Equal contribution. Listing order is random

[†]Work done while at Huawei Noah's Ark Lab.

[♠]Corresponding author.

¹The term point to objectives that project the last layer representation to vocabulary space in order to output tokens (e.g. MLM, casual LM, or the one of T5).

In TOR, token-order information is removed^{[2](#page-1-0)} from the input sequence, and a model is tasked to predict the neighbor token-to-token positional relations. We further investigate the utility of a novel structure-aware architecture that consists in end-toend pretraining of a Graph Isomorphism Network (GIN) model [\(Xu et al.,](#page-11-5) [2018\)](#page-11-5) placed on top of the BERT encoder [\(Devlin et al.,](#page-9-0) [2019\)](#page-9-0).

As some NLU tasks may not *always* require strong syntactic understanding [\(Glavaš and Vulic´,](#page-9-6) [2021;](#page-9-6) [Kulmizev and Nivre,](#page-9-3) [2021;](#page-9-3) [Haidar et al.,](#page-9-7) [2021\)](#page-9-7), we conduct a thorough empirical analysis on both word-order insensitive tasks from the GLUE [\(Wang et al.,](#page-11-2) [2018\)](#page-11-2) benchmark, as well as syntax-sensitive ones, namely Dependency Parsing (DP) [\(Kübler et al.,](#page-9-8) [2009\)](#page-9-8).

Our study shows that learning representations with an order reconstruction objective is highly effective only when the input sequence is partially (compared to fully) shuffled. Second, pretraining with TOR leads to competitive performances on order insensitive tasks compared with MLM, and superior performance on order sensitive ones especially in the few-shot setting. Third, BERT trained with TOR shows better sensitivity to absence of word-order information than BERT-MLM, thereby being a potential method to alleviate some of the concerns raised on PLM's syntax understanding. Yet, we find that with enough labelled data, TOR have hardly any additional value, which is consistent with other task-specific objectives [\(Ram et al.,](#page-10-5) [2021;](#page-10-5) [Jia et al.,](#page-9-9) [2022\)](#page-9-9).

2 Related Work

Language Modelling objectives such as BERT's masked language modelling [\(Devlin et al.,](#page-9-0) [2019\)](#page-9-0), XL-NET's permutation language modelling [\(Yang](#page-11-0) [et al.,](#page-11-0) [2019\)](#page-11-0), GPT next word prediction [\(Rad](#page-10-6)[ford et al.,](#page-10-6) [2018\)](#page-10-6), as well as auto-regressive sequence denoising ones of BART [\(Lewis et al.,](#page-9-10) [2019\)](#page-9-10) and MASS [\(Song et al.,](#page-11-1) [2019\)](#page-11-1) are popular self-supervised representation learning routines used in NLU tasks. Learning contextual word representations is grounded in linguistics [\(Culbertson](#page-8-4) [and Adger,](#page-8-4) [2014;](#page-8-4) [Futrell et al.,](#page-9-11) [2020\)](#page-9-11) and psycholinguistics [\(Hale,](#page-9-12) [2017;](#page-9-12) [Mollica et al.,](#page-10-7) [2020\)](#page-10-7) literature that supports that the natural order of words helps humans better capturing semantic information. [Mollica et al.](#page-10-7) [\(2020\)](#page-10-7) in their studies with humans found that local ordering of words

²Through the removal of spatial (positional) information.

when preserved eased comprehension when small perturbations affected word-order in the input text.

Despite large data and sophisticated inductive biases, PLMs seem to not quite understand the language like humans do [\(O'Connor and Andreas,](#page-10-8) [2021\)](#page-10-8). Recent studies [\(Sinha et al.,](#page-10-4) [2021b;](#page-10-4) [Gupta](#page-9-13) [et al.,](#page-9-13) [2021;](#page-9-13) [Pham et al.,](#page-10-9) [2020\)](#page-10-9) show that large language models are insensitive to word-order. These works measure the sensitivity of PLMs to task performance when a language model is pretrained [\(Sinha et al.,](#page-10-3) [2021a;](#page-10-3) [Alajrami and Aletras,](#page-8-3) [2022\)](#page-8-3) or fine-tuned [\(Sinha et al.,](#page-10-4) [2021b;](#page-10-4) [Hessel and](#page-9-14) [Schofield,](#page-9-14) [2021\)](#page-9-14) with text sequences with deleted or shuffled tokens. Notably, [\(Abdou et al.,](#page-8-5) [2022;](#page-8-5) [Clouâtre et al.,](#page-8-6) [2022\)](#page-8-6) demonstrate that PLMs are insensitive to word-order information suggesting further that language modeling objectives alone may not be sufficient to encode the essential aspects of syntactic abstraction of language understanding.

Exploring alternative pretraining objectives, such as linguistically (e.g. character, part of speech) informed [\(Yamaguchi et al.,](#page-11-6) [2021\)](#page-11-6), task-specific (e.g. question answering) [\(Ram et al.,](#page-10-5) [2021;](#page-10-5) [Jia](#page-9-9) [et al.,](#page-9-9) [2022\)](#page-9-9), and word-order aware ones [\(Raffel](#page-10-2) [et al.,](#page-10-2) [2019;](#page-10-2) [Wang et al.,](#page-11-7) [2019b\)](#page-11-7) has been gaining attention lately. With that, exploring inductive biases that better capture such objectives too has been gaining attention. Among such inductive biases, Graph Neural Network (GNN) [\(Scarselli](#page-10-10) [et al.,](#page-10-10) [2008\)](#page-10-10) has become popular due to their conventional use of structure prediction tasks that involve entities and relations, which also aligns with syntactic tasks such as parsing [\(Ji et al.,](#page-9-15) [2019\)](#page-9-15), ordering or tagging [\(Zhu et al.,](#page-11-8) [2021;](#page-11-8) [Zhang et al.,](#page-11-9) [2021\)](#page-11-9). Also, [Yasunaga et al.](#page-11-10) [\(2021\)](#page-11-10) use GNNs in pretraining language models for the Question Answering task.

The proposed TOR objective is different along two major aspects when compared with its relevant counterparts. First, it uses a partial *bag-of-words* representation of input sequence compared to full (T5 [\(Raffel et al.,](#page-10-2) [2019\)](#page-10-2) *deshuffling* objective) or trigram window (StructBERT [\(Wang et al.,](#page-11-7) [2019b\)](#page-11-7) *word structural* objective) tokens shuffling. Second, TOR uses a pairwise token-to-token relation to represent the output target, compared to projecting hidden representations to the token vocabulary space unlike *deshuffling* and *word structural*. Further, using the tokens in the input to re-order instead of predicting over the entire vocabulary provides significant computational gains over the other objectives; with TOR, we could fit a batch size which is 33% larger than token prediction objectives like MLM.

3 TOR

We formulate a new pre-training task for selfsupervised representation learning for NLU by proposing TOR, a TOken Reordering objective. We describe the input representations and target design in [§3.1](#page-2-0) and [§3.2](#page-2-1) respectively, and the main details of our proposed BERT+GIN model and the motivations behind it in [§3.3.](#page-2-2)

3.1 Model Input

For a given pretraining token sequence $X = \{x_1, x_2, \ldots, x_n\}$ of length *n*, let $P=[0,1,\ldots,n-1] \in \mathbb{N}^n$ be the absolute position index of X . First, we generate a random binary vector $P' = [p'_1, p'_2, \dots, p'_n]$, where 1 and 0 respectively indicate if a position p_i (element in P) will be masked or not during pre-training:

$$
p'_{i} = \begin{cases} 1 & u \sim \mathcal{U}(0,1) \le \lambda \\ 0 & \text{o.w.} \end{cases}
$$
 (1)

where λ is a threshold parameter and $\mathcal{U}(0, 1)$ refers to the uniform distribution in the range $[0, 1]$. Then, we update p_i as follow:

$$
p_i = \begin{cases} p_i & p'_i == 0 \\ n & \text{o.w.} \end{cases}
$$

For implementation efficiency, we use an extra positional index *n* as a special *mask* index $(p_{[M]}$ in Figure [1\)](#page-0-1). Also, we define $F \in \mathbb{N}^n$ where f_i is the frequency count of x_i in X . For instance, if the same token occurs three times in X at positions i, j, k , then f_i, f_j , and f_k would equal to 0, 1 and 2 respectively. F is crucial to distinguish between the representations of same tokens when their positions are masked. Finally, we obtain a continuous vector representation of the input sequence as follow:

$$
H^s = E_X(X) + E_P(P) + E_F(F)
$$
 (2)

 $E_X(\cdot)$, $E_P(\cdot)$, $E_F(\cdot)$ are embedding lookup functions that are parameterized by $W_X \in \mathbb{R}^{v \times d}$, $W_P \in \mathbb{R}^{(n+1)\times d}$, $W_F \in \mathbb{R}^{n\times d}$ respectively, where d and v are the hidden dimension and vocabulary size, respectively. The sum of the resultant vectors $H^s \in \mathbb{R}^{n \times d}$ is used as input representation of the

encoder described in [§3.3.](#page-2-2) P' and F are dynamically generated using highly efficient vectorized operations on GPU, thus adding no computational overhead during pretraining. Also, it is important to mention that TOR, and MLM can be coupled. However, when pre-training with both objectives, we avoid masking positions $P[i-1:i+1]$ if the token x_i is masked by MLM $(x_i \leftarrow x_{[M]})$.

3.2 Model Output

Given $H^f = [\mathbf{h}_1^f]$ f_1^f,\mathbf{h}_2^f $\mathbf{h}_2^f, \ldots, \mathbf{h}_n^f]^T \in \mathbb{R}^{n \times d}$, a sequence of representation vectors output by an encoder module $(\S 3.3)$, we apply a normalized version of a selfattention operator to H^f in order to obtain the output matrix $O \in \mathbb{R}^{n \times n}$:

$$
O = \text{Softmax}(H^f W_Q W_K H^{f^T}) \tag{3}
$$

 W_Q , $W_K \in \mathbb{R}^{d \times d}$ are learnable self-attention matrices. Then, our training objective is defined as cross-entropy between the output matrix O and the ground-truth target matrix T :

$$
\mathcal{L} = -\sum_{i=1}^{n} \Gamma(i, i+k) T(i) \log (\mathcal{O}[i]) \tag{4}
$$

where $T(i)$ and $O[i]$ refer to the ith row of the T and O matrices respectively. The ground-truth target matrix $T \in \{0,1\}^{n \times n}$ (TARGET matrix in [Figure 2\)](#page-3-0) is defined based on the neighbor position of tokens at distance k (k is a hyper-parameter):

$$
T(i) = \begin{cases} \text{One-Hot}(i+k, n), & 0 \le i+k < n \\ \mathbf{0} \in \mathbb{R}^n, & \text{o.w.} \end{cases}
$$
 (5)

It generates an n dimensional one-hot row vector at index $i + k$ when possible and generates a zero vector otherwise, k is a hyper-parameter which we set to $+1$ in this work. Note that we don't compute loss at position i, if both p_i and p_{i+k} are not masked:

$$
\Gamma[i,j] = \begin{cases} 0, & (p'_i \& p'_j) = 0 \\ 1, & \text{o.w.} \end{cases}
$$
 (6)

3.3 Encoder

In this section, we investigate two encoder architectures that take H^s as input, and output H^f .

Figure 2: Illustration of our GIN encoder placed on top of BERT output during pretraining. Circled numbers are per-token hidden states, while gray and cyan indicate masked and unmasked input positions (same example of Figure [1\)](#page-0-1) respectively. Bold underscored entries indicate that values were overwritten by the edge masking function $EM(.,.)$ of equation [8.](#page-3-1) Solid and dotted arrows indicate overwritten and predicted arc weights respectively, while the opacity level of arcs reflect its value in the adjacency matrix. w is the windows size, H^b and H^g are BERT and GIN output hidden states respectively. H^{G^1} , and H^{G^2} , H^{G^4} are hidden output of GINs G^1 , G^2 , and G^4 respectively. L is concatenation and transparent target lines are ignored during loss calculation.

3.3.1 BERT

We pass H^s to a b-layer BERT encoder to obtain a sequence of hidden representations $H^b =$ $[\mathbf{h}_1^b, \mathbf{h}_2^b, ..., \mathbf{h}_n^b]^T \in \mathbb{R}^{n \times d}$. We set $H^f \leftarrow H^b$ in Equation [3](#page-2-3) to compute $\mathcal L$ when this encoder is used for pretraining.

3.3.2 BERT+GIN

This encoder contains several GIN modules (as depicted in Figure [2\)](#page-3-0) that are layered over the BERT output to refine H^b . We constrain the input of the graphs by explicitly injecting known neighbors information ($\Gamma(i, j) == 0$), in a context window w, as a form of golden links that overwrite the predicted ones. For each window size w , we define a GIN module \mathcal{G}^w which takes as input BERT hidden representations H^b and an adjacency matrix $A^{\mathcal{G}^w}$ and produces $H^{\mathcal{G}^w} = [\mathbf{h}_1^{\mathcal{G}^w}]$ ${\cal G}^{w^\bullet}\!\!, {\bf h}^{{\cal G}^{w}}_2$ $\frac{{\cal G}^w}{2},\ldots,{\bf h}_n^{{\cal G}^w}$ $\frac{\mathcal{G}^w}{n}$]^T \in $\mathbb{R}^{n \times d}$ as follows:

$$
H^{\mathcal{G}^w} = \mathcal{G}^w(H^b, \mathcal{A}^{\mathcal{G}^w})
$$
 (7)

We obtain the adjacency matrix \mathcal{A}^w by passing H^b to a self-attention function followed by an edge masking $EM(\cdot, \cdot)$ operator:

$$
\mathcal{A}^{\mathcal{G}^w} = EM\left(\text{Sigmoid}(H^b W_Q^{\mathcal{G}^w} W_K^{\mathcal{G}^w} H^{b^T}); w\right)
$$

$$
EM(a_{ij}; w) = \begin{cases} 0, & i == j \\ 1, & \mathcal{C}(i, j) \& j \in [i, i + w] \\ 0, & \mathcal{C}(i, j) \& j \notin [i, i + w] \\ a_{ij}, & o.w. \end{cases}
$$
(8)

where $\mathcal{C}(i, j) = \Gamma(i, j) == 0$, indicates whether the input positions of node i and j are not masked, and $W_Q^{\mathcal{G}^w}$ \mathcal{G}^w , $\mathcal{W}_K^{\mathcal{G}^w}$, $\in \mathbb{R}^{d \times d}$ are learnable parameters. Concretely, \mathcal{G}^w consists of L^w Multi Layer Perceptron (MLP) [\(Ramchoun et al.,](#page-10-11) [2016\)](#page-10-11) which updates the representation of a node $h_i^{\mathcal{G}^w}$ $\int_i^{G_w}$ at the l^{th} layer:

$$
\mathbf{h}_{i}^{(l+1)} = \text{MLP}\Big(\left(1 + \varepsilon_{(l)}\right)\mathbf{h}_{i}^{l-1} + \sum_{j \in \mathcal{N}_{i}} \mathbf{h}_{j}^{(l-1)}\Big) \tag{9}
$$

we wrote $h_i^{\mathcal{G}^{w(l)}}$ $\frac{\mathcal{G}^{w(l)}}{i}$ as $\mathbf{h}_i^{(l)}$ $i^{(i)}$ in Equation [9](#page-3-2) for simplicity, $\mathbf{h}_i^{(0)} \leftarrow \mathbf{h}_i^b$, $\varepsilon_{(\cdot)}$ are hyper-parameters, and $\mathbf{h}^{(l)}_i$ $i^{(l)}$ refers to the i^{th} node representation at the

 lth layer within the GIN \mathcal{G}^w . \mathcal{N}_i is the set of all neighbor nodes of the ith node obtained from $A^{\mathcal{G}^w}$. Finally, we concatenate all $H^{\mathcal{G}^w}$ and feed them to a FFNN layer in order to obtain a single hidden representation of all the GIN encoders $\overline{H}^g = [\mathbf{h}^g_1]$ $^{g}_{1}, \mathbf{h}^{\overset{.}{g}}_{2}$ $[2^g, \ldots, \mathbf{h}_n^g]^T \in \mathbb{R}^{n \times d}$. The number of GIN modules, and their corresponding layers and window sizes are hyper-parameters. During pretraining with the BERT+GIN, we set $H^f \leftarrow H^g$ in Equation [3](#page-2-3) for TOR loss computation.

3.3.3 Motivation behind BERT+GIN

GINs, a special family of GNNs, are characterized by their ability to leverage topological signals from an adjacency matrix in order to capture and fuse information from both local and global neighbor nodes [\(Chen et al.,](#page-8-7) [2019;](#page-8-7) [Zhu et al.,](#page-11-8) [2021\)](#page-11-8). We find GIN's sparsity characteristic to align with the inductive biases required to support the TOR task. Further, it is important to mention that we discard the GIN encoder and only use the BERT representation when fine-tuning models trained with TOR. Since we deactivate TOR during fine tuning, the edge of \mathcal{A}^w will be fully masked by $EM(\cdot, \cdot)$. Therefore, each node will only have access to its immediate neighbors, which is not suitable for downstream tasks. However, we empirically found that explicitly injecting known neighbor edges over disjoint w -hops is beneficial for pretraining. It allows us to generate multiple views of the same graph. Since the GIN encoders are disjoint, this enforces the BERT intermediate representations to be comprehensive in order to successfully solve the task.

4 Experiments

4.1 Baselines

We conduct experiments on 4 configurations in order to compare between models pretrained with MLM and TOR objectives. All models use the BERT-base configuration of [Devlin et al.](#page-9-0) [\(2019\)](#page-9-0) $(d=768; b=12)$ as the encoder. **BERT-M, BERT-**T, and BERT-MT are models with BERT encoder of [§3.3.1](#page-3-3) pretrained with MLM only, TOR only, and both MLM and TOR objectives respectively. BERT+GIN-Ts use the encoder of [§3.3.2](#page-3-4) where TOR is the only used pretraining objective.

4.2 Implementation Details

Due to limited computational resources, we define an experimental pretraining protocol similar to the one of [Yamaguchi et al.](#page-11-6) [\(2021\)](#page-11-6). It consists in pretraining our four baseline models from scratch on 8 V100 GPUs during a maximum of 5 days each with the BERT-base configuration [\(Devlin et al.,](#page-9-0) [2019\)](#page-9-0). The pretraining configurations and implementation details are listed in Appendix [A.1.](#page-12-0) On the fine tuning side, we conduct extensive experiments on 8 GLUE [\(Wang et al.,](#page-11-2) [2018\)](#page-11-2) text classification tasks, and 6 Dependency Parsing (DP) datasets. When referring to a score, GLUE and DP indicate the unweighted average scores over benchmark respective tasks. A detailed description of the datasets, evaluation metrics, and fine tuning implementation details are available in Appendix [A.3,](#page-12-1) [A.2.](#page-12-2)

4.3 Results Integrity

Table [1](#page-4-0) shows the average GLUE score of the original BERT-base of [Devlin et al.](#page-9-0) [\(2019\)](#page-9-0) (BERT-ORG), the MLM model re-implementation of [Ya](#page-11-6)[maguchi et al.](#page-11-6) [\(2021\)](#page-11-6) (BERT-5D8G), as well as our BERT-M and BERT-T models. The last three models are all pretrained during 5 days on 8 $V100$ GPUs.

Model	GLUE.	Model	GLUE	
BERT-ORG	82.9	BERT-M	81.6	
BERT-5D8G	77.6	BERT-T	79.4	

Table 1: Average GLUE dev scores of MLM models of [\(Devlin et al.,](#page-9-0) [2019\)](#page-9-0) (BERT-ORG), [\(Yamaguchi et al.,](#page-11-6) [2021\)](#page-11-6) (BERT-5D8G), our re-implementation (BERT-M), as well as our BERT-T model.

BERT-M is only 1.3% behind BERT-ORG, while significantly outperforming BERT-5D8G by 4 points, despite using the same computational budget. This is because we are able to fit a larger batch size (270) on a single GPU compared to the latter work (32). The above figures confirms the validity of our pretraining settings, and subsequently the reliability of our end-task results. It is worth mentioning that BERT-T (79.4) is not only outperforming the MLM implementation of [\(Yamaguchi](#page-11-6) [et al.,](#page-11-6) [2021\)](#page-11-6), but also their best model (79.2) pretrained with their the *Shuffle+Random* objective.

4.4 Full vs. Partial Re-order Pretraining

We highlight the importance of partial token reordering by running three pretraining experiments on the BERT-T model by varying the λ reordering probability. Table [2](#page-5-0) reports the average GLUE and DP results when BERT-T is pretrained with

Figure 3: Models performance on 3 GLUE tasks, as well as average GLUE average score across training set sizes.

different λ values. We notice that values of 0.3 and 0.5 perform similarly, therefore we used the latter as a default to also pretrain (and report results with) all three TOR models.

	GLUE	Δ	DP	
0.3	78.2	-3.4	90.2 90.4	-0.5
0.5	79.4	-2.2		-0.3
1.0	72.6	-9.0	70.5	-20.2

Table 2: Average GLUE and DP Test score when varying λ during the pretraining of BERT-T model. Δ shows absolute performance gap with BERT-M.

Moreover, full token re-ordering $(\lambda=1.0)$ performs poorly on downstream tasks, 9.0% and 20.2% below BERT-M on GLUE and DP respectively. Interestingly, roughly the same gap on GLUE is reported between the *deshuffling* and MLM objectives in T5 [\(Raffel et al.,](#page-10-2) [2019\)](#page-10-2) experiments. This pushed the authors to prematurely dismiss this objective in their experimental stage. Our work demonstrates that word-order pretraining is meaningful when performed on partially shuffled sequences, which is one of the core features (beside efficiency) supported by TOR.

4.5 Impact of the GIN Module

Figure [4](#page-5-1) shows the GLUE and DP average scores (full results are in Appendix [B\)](#page-12-3) of our two models trained with the TOR objective only. We observe that BERT+GIN-T always performs better compared to BERT-T across all settings. For instance, when using 32 and 64 examples we respectively observe a gap of 5.9% and 5.5% on GLUE averagescore, and 14.2% and 9.5% on DP average. However, we observe that the gap steadily reduces when more examples are added. Not shown in Fig-

ure [4,](#page-5-1) fine-tuning on the full dataset reduce the gap to $+0.5\%$. Since the GIN is discarded during fine tuning (no extra parameter), it is reasonable to conclude that pretraining GIN was a key factor in forcing BERT to encode representations that generalize better on downstream tasks.

Figure 4: Average GLUE (left) and DP (right) performances of BERT-T and BERT+GIN-T models across training set size (few shot setting).

4.6 MLM vs. TOR: Order Insensitive Tasks

Figure [3](#page-5-2) shows few shot setting performances on [3](#page-5-3) GLUE tasks, 3 as well as the average GLUE score for the best TOR model (BERT+GIN-T), our MLM only model (BERT-M), as well as our model using both MLM and TOR (BERT-MT). We observe that BERT+GIN-T underperforms models that use MLM (BERT-M and BERT-MT) across all data sizes. A Similar pattern is observed

³We couldn't put the full dataset performances in the plot for visualization purposes (curves will collapse on each other). We selected RTE because it shows specific results, CoLA since with MNLI they show similar result patterns, and SST-2 as a representative of trends observed for tasks MRPC, STS-B, QQP, MLNI. However, the detailed performances are presented in table [4](#page-13-0) of Appendix [B.](#page-12-3)

on MRPC, STS-B, QQP, MLNI order-insensitive tasks. This observation was expected and is inline with previous works [\(Abdou et al.,](#page-8-5) [2022;](#page-8-5) [Hessel](#page-9-14) [and Schofield,](#page-9-14) [2021;](#page-9-14) [Sinha et al.,](#page-10-3) [2021a\)](#page-10-3) that state that most of GLUE tasks can be solved by ignoring word order.

Pretraining with both MLM and TOR improves the overall performance of BERT-M up to certain number of fine tuning examples, especially on RTE. On very low resource settings, we notice that BERT-MT performs on par with BERT-M on 16 and 32 examples GLUE average, and significantly better (55.8% vs 54.7%) on 64 examples. However, increasing the training data size gradually demolishes gains that come from pretraining with the TOR objective. For instance, when fine tuning on 128 or more examples, BERT-M consistently outperforms BERT-MT on SST-2 (and MRPC, STS-B, QQP, MLNI). Note that BERT-MT has roughly the same average score performance of BERT-M trained with 128 examples, which is due to an unexpected gain of 7.6% on CoLA. While on full dataset, BERT-MT is only able to retain a gain of 1.1% and 0.8% on CoLA and RTE respectively compared to BERT-M. The observations suggest that word-order pretraining objectives, like TOR, are useful when the end task requires syntax understanding, and the labeled data is not abundant.

4.7 MLM vs. TOR: Order Sensitive Tasks

Nevertheless, we notice that BERT+GIN-T significantly outperforms BERT-M and BERT-MT on CoLA (QNLI shows a similar pattern) on all few shot settings. For instance, BERT+GIN-T reports a gain of 3.1% and 7.9% on top of BERT-M on 32 and 128 examples respectively. CoLA, which tests a model's ability to predict the linguistic acceptability of sentences, presumably relies on word order. However, BERT+GIN-T is only able to maintain top performance on CoLA (and QNLI) for up to 256 examples, before being outperformed by BERT-MT on the full dataset.

The results on CoLA motivated us to evaluate on Dependency Parsing (DP), a task that requires predicting if the *head* relationship exists between all word pairs of a sentence (link prediction), and its relation type (classification). The arcs prediction sub-task of DP is inline with the decision making in TOR. Figure [5](#page-6-0) shows the LAS average score on the test set 4 of 6 dependency parsing benchmarks

across various training set sizes. Per dataset dev and test performances and standard deviation statistics are presented in Table [5](#page-14-0) and [6](#page-15-0) in Appendix [B.](#page-12-3)

Figure 5: LAS average score on test set of six dependency parsing datasets across training set sizes.

First, it is important to note that our BERT-M performance on PTB full dataset (94.7) is inline with that of the BERT-base model of [Zhou and](#page-11-11) [Zhao](#page-11-11) [\(2019\)](#page-11-11) (95.4). Second, BERT+GIN-T systematically outperforms BERT-M and BERT-MT across all few shot configurations. These observations were expected as dependency parsing relies more on word-order indicative bias compared to GLUE tasks. The results highlight the importance of order-aware pretraining objective (e.g. TOR) and encoder (e.g. GIN) when the task comprises word-word relationships.

However, we observe that the gains of BERT+GIN-T on top of BERT-M is — again inversely proportional to the number of fine tuning examples. For instance, BERT+GIN-T outperforms BERT-M by 12.3%, 7.2% and 2.8% on 16, 32, and 64 examples respectively. Unfortunately, training on more data (e.g. 40k PTB examples) steadily decreases this gain.

Based on those extensive experiments, we conclude the following. First, pretraining with language modelling objectives (MLM and its variants) is vital for end task NLU performance. Second, we highlight the importance of labelled data size as the most critical factor for NLU performance. For those reasons, new pretraining objectives (like TOR) should be used as auxiliary objectives when training a language (e.g. MLM+TOR). The contribution of the novel pretraining objectives we propose become however less important when enough fine-tuning data is available. A similar observation

⁴Performances on DEV set show very similar trends.

is reported in [\(Ram et al.,](#page-10-5) [2021;](#page-10-5) [Jia et al.,](#page-9-9) [2022\)](#page-9-9), both proposing new pretraining objectives specifically designed for the Question Answering task. This also may partially explain why works on extremely large PLM [\(Brown et al.,](#page-8-8) [2020b;](#page-8-8) [Du et al.,](#page-9-16) [2021;](#page-9-16) [Chowdhery et al.,](#page-8-9) [2022\)](#page-8-9) also prefer to report results on few shot and zero shot settings.

4.8 MLM vs. TOR: Perturbation Probing

Following recent works on probing [\(Sinha et al.,](#page-10-4) [2021b](#page-10-4)[,a;](#page-10-3) [Clouâtre et al.,](#page-8-6) [2022;](#page-8-6) [Abdou et al.,](#page-8-5) [2022\)](#page-8-5), we modify the dev set of GLUE tasks by randomly shuffling n -grams^{[5](#page-7-0)}, and also by randomly masking some tokens in the input sequence. Figure [6](#page-7-1) shows the average GLUE score of BERT-M and BERT-T models on shuffling (left) and masking (right) perturbation experiments respectively. Detailed results can be found in Table [7](#page-16-0) and [8](#page-16-1) in Appendix [B.](#page-12-3)

Figure 6: Average dev GLUE score of n -gram shuffling (left) and token masking (right) perturbation probing.

We observe that BERT-T outperforms BERT-M on fully shuffled sequences $(n = 1)$ by 2.1%. We think that, even after fine-tuning, BERT-T has preserved some of its ordering ability induced by the TOR objective. Increasing n (span-level shuffling) reduces the gap between models, as results tend to converge to the pattern saw on full dataset in Table [2.](#page-5-0) Results are inline with the ones of the PLMs probing literature [\(Sinha et al.,](#page-10-3) [2021a;](#page-10-3) [Clouâtre](#page-8-6) [et al.,](#page-8-6) [2022;](#page-8-6) [Abdou et al.,](#page-8-5) [2022\)](#page-8-5), which confirms that PLMs are insensitive to global language structure. Expectedly, the performance of BERT-M is significantly higher $(+4.5%)$ compared to BERT-T when the range of masking probability is similar to the one that BERT-M was pretrained with (10- 20%). However, the performances of both models

steadily converge to the one of the random guessing baseline, when increasing the masking probability to high values.

4.9 Token Reordering Ability

We leverage the token ordering performance of pretrained BERT-T and BERT+GIN-T models by measuring their token re-ordering abilities on raw sentences. We do so by partially masking the absolute position (as in [§3.1\)](#page-2-0) of GLUE and DP dev sets input sequences using a $\lambda = \{0.5, 1.0\}$. Then, we measure pairwise ordering accuracy, which is a binary score indicating if a true subsequent tokens pairs are correctly predicted. Table [3](#page-7-2) shows models average pairwise ordering accuracy (binary score indicating if a true subsequent tokens pairs are correctly predicted.) on 8 GLUE and 6 DP datasets with different values of λ applied on input sequence. Per-task detailed results are presented in Table [9](#page-17-0) and [10](#page-17-1) of Appendix [B.](#page-12-3)

		GLUE	DP		
	$0.5 -$	1.0	0.5	10	
BERT-T		24\% 17\% 27\% 24\%			
BERT+GIN-T	32%	19%	137% 26\%		

Table 3: Average pairwise ordering accuracy on 8 GLUE dev sets, where the position of input sequence are masked a with probability λ (0.5 and 1.0).

Expectedly, BERT+GIN-T systematically outperforms BERT-T which showcases the value of our proposed BERT+GIN architecture. Also, it is promising to see a positive correlation between the token ordering and end-task performance, where improving the first may naturally reflect as an improvement on the second. The overall poor performances, especially on full re-reordering ($\lambda = 1.0$), is not surprising since TOR is designed for representation learning, not for *text linearization* [\(Elman,](#page-9-17) [1990\)](#page-9-17). The latter is out of the scope of this paper, as its is commonly approached with computationally expensive search algorithms powered with a LM scorer [\(De Gispert et al.,](#page-8-10) [2014;](#page-8-10) [Malkin et al.,](#page-10-12) [2021\)](#page-10-12). For instance, the *IBSB* algorithm of [\(Malkin et al.,](#page-10-12) [2021\)](#page-10-12) performs 27.8k query per sentence on average to GPT-small [\(Radford et al.,](#page-10-6) [2018\)](#page-10-6) to guide the re-ordering heuristic.

⁵We concatenate *n*-grams before performing shuffling

5 Conclusion

We revisit word-order pretraining for NLU by proposing a novel self-supervision task (TOR), as well as a dedicated encoder architecture. The goal is to investigate if injecting syntactic biases into PLM during pretraining would improves their awareness to language structure. While experiments on TOR show promises in enhancing PLM understanding of language structure, still many challenges remain in maintaining performances on word order insensitive tasks. We thereby highlight the importance of word-order pretraining objectives as an interesting research direction to explore in future.

Limitations

Ablations on pretraining hyperparameters, as well as on GIN architecture design choices (e.g. number of layers and window sizes) may have further enhanced the performance or provided information on the sensitivity of the architecture to those choices. The evaluation on syntactic tasks is done on Dependency parsing only. Extending the experiments to other syntactic tasks such as constituency parsing or syntax diagnosing benchmarks like SyntaxGym [\(Gauthier et al.,](#page-9-18) [2020\)](#page-9-18) or BLiMP [\(Warstadt](#page-11-12) [et al.,](#page-11-12) [2020\)](#page-11-12) could have improved the generality of the claims on the usefulness of word order pretraining objective.

Acknowledgements

We thank Mindspore^{[6](#page-8-11)} for the partial support of this work, which is a new deep learning computing framework. We thank the anonymous reviewers for their insightful comments.

References

- Mostafa Abdou, Vinit Ravishankar, Artur Kulmizev, and Anders Søgaard. 2022. Word order does matter and shuffled language models know it. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6907–6919.
- Lars Ahrenberg. 2007. Lines: An english-swedish parallel treebank. In *Proceedings of the 16th Nordic Conference of Computational Linguistics (NODAL-IDA 2007)*, pages 270–273.
- Ahmed Alajrami and Nikolaos Aletras. 2022. How does the pre-training objective affect what large language

models learn about linguistic properties? In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 131–147.

- Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020a. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.
- Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020b. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.
- Danqi Chen and Christopher D Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 740–750.
- Zhengdao Chen, Soledad Villar, Lei Chen, and Joan Bruna. 2019. On the equivalence between graph isomorphism testing and function approximation with gnns. *Advances in neural information processing systems*, 32.
- Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2022. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*.
- Kevin Clark, Minh-Thang Luong, Quoc V Le, and Christopher D Manning. 2019. Electra: Pre-training text encoders as discriminators rather than generators. In *International Conference on Learning Representations*.
- Louis Clouâtre, Prasanna Parthasarathi, Amal Zouaq, and Sarath Chandar. 2022. Local structure matters most: Perturbation study in nlu. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 3712–3731.
- Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *ACL*.
- Jennifer Culbertson and David Adger. 2014. Language learners privilege structured meaning over surface frequency. *Proceedings of the National Academy of Sciences*, 111(16):5842–5847.
- Adrià De Gispert, Marcus Tomalin, and Bill Byrne. 2014. Word ordering with phrase-based grammars. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 259–268.

⁶ <https://www.mindspore.cn/>

- Marie-Catherine De Marneffe and Christopher D Manning. 2008. Stanford typed dependencies manual. Technical report, Technical report, Stanford University.
- 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.
- Timothy Dozat and Christopher D Manning. 2016. Deep biaffine attention for neural dependency parsing. *arXiv preprint arXiv:1611.01734*.
- Nan Du, Yanping Huang, Andrew M. Dai, Simon Tong, Dmitry Lepikhin, Yuanzhong Xu, Maxim Krikun, Yanqi Zhou, Adams Wei Yu, Orhan Firat, Barret Zoph, Liam Fedus, Maarten Bosma, Zongwei Zhou, Tao Wang, Yu Emma Wang, Kellie Webster, Marie Pellat, Kevin Robinson, Kathy Meier-Hellstern, Toju Duke, Lucas Dixon, Kun Zhang, Quoc V Le, Yonghui Wu, Zhifeng Chen, and Claire Cui. 2021. [Glam: Efficient scaling of language mod](https://doi.org/10.48550/ARXIV.2112.06905)[els with mixture-of-experts.](https://doi.org/10.48550/ARXIV.2112.06905)
- Jeffrey L Elman. 1990. Finding structure in time. *Cognitive science*, 14(2):179–211.
- Richard Futrell, Roger P Levy, and Edward Gibson. 2020. Dependency locality as an explanatory principle for word order. *Language*, 96(2):371–412.
- Jon Gauthier, Jennifer Hu, Ethan Wilcox, Peng Qian, and Roger Levy. 2020. Syntaxgym: An online platform for targeted evaluation of language models. In *Association for Computational Linguistics (ACL)*.
- Goran Glavaš and Ivan Vulić. 2021. Is supervised syntactic parsing beneficial for language understanding tasks? an empirical investigation. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 3090–3104.
- Yoav Goldberg. 2019. Assessing bert's syntactic abilities. *arXiv preprint arXiv:1901.05287*.
- Ashim Gupta, Giorgi Kvernadze, and Vivek Srikumar. 2021. Bert & family eat word salad: Experiments with text understanding. In *Proceedings of the AAAI Conference on Artificial Intelligence*.
- Md Akmal Haidar, Nithin Anchuri, Mehdi Rezagholizadeh, Abbas Ghaddar, Philippe Langlais, and Pascal Poupart. 2021. Rail-kd: Random intermediate layer mapping for knowledge distillation. *arXiv preprint arXiv:2109.10164*.
- John Hale. 2017. Models of human sentence comprehension in computational psycholinguistics. *Oxford Research Encyclopedia of Linguistics*.
- Jack Hessel and Alexandra Schofield. 2021. How effective is bert without word ordering? implications for language understanding and data privacy. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 204–211.
- Junjie Hu, Sebastian Ruder, Aditya Siddhant, Graham Neubig, Orhan Firat, and Melvin Johnson. 2020. [Xtreme: A massively multilingual multi-task](http://arxiv.org/abs/2003.11080) [benchmark for evaluating cross-lingual generaliza](http://arxiv.org/abs/2003.11080)[tion.](http://arxiv.org/abs/2003.11080) *CoRR*, abs/2003.11080.
- Tao Ji, Yuanbin Wu, and Man Lan. 2019. [Graph-based](https://doi.org/10.18653/v1/P19-1237) [dependency parsing with graph neural networks.](https://doi.org/10.18653/v1/P19-1237) In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2475– 2485, Florence, Italy. Association for Computational Linguistics.
- Robin Jia, Mike Lewis, and Luke Zettlemoyer. 2022. Question answering infused pre-training of generalpurpose contextualized representations. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 711–728.
- Amirhossein Kazemnejad, Mehdi Rezagholizadeh, Prasanna Parthasarathi, and Sarath Chandar. 2023. Measuring the knowledge acquisition-utilization gap in pretrained language models. *arXiv preprint arXiv:2305.14775*.
- Sandra Kübler, Ryan McDonald, and Joakim Nivre. 2009. Dependency parsing. *Synthesis lectures on human language technologies*, 1(1):1–127.
- Artur Kulmizev and Joakim Nivre. 2021. Schr\" odinger's tree–on syntax and neural language models. *arXiv preprint arXiv:2110.08887*.
- Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*.
- Quentin Lhoest, Albert Villanova del Moral, Yacine Jernite, Abhishek Thakur, Patrick von Platen, Suraj Patil, Julien Chaumond, Mariama Drame, Julien Plu, Lewis Tunstall, et al. 2021. Datasets: A community library for natural language processing. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 175–184.
- 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*.
- Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.
- Nikolay Malkin, Sameera Lanka, Pranav Goel, and Nebojsa Jojic. 2021. Studying word order through iterative shuffling. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 10351–10366.
- Ryan McDonald, Joakim Nivre, Yvonne Quirmbach-Brundage, Yoav Goldberg, Dipanjan Das, Kuzman Ganchev, Keith Hall, Slav Petrov, Hao Zhang, Oscar Täckström, et al. 2013. Universal dependency annotation for multilingual parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 92–97.
- Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, and Hao Wu. 2018. Mixed precision training. In *In International Conference on Learning Representations*.
- Francis Mollica, Matthew Siegelman, Evgeniia Diachek, Steven T Piantadosi, Zachary Mineroff, Richard Futrell, Hope Kean, Peng Qian, and Evelina Fedorenko. 2020. Composition is the core driver of the language-selective network. *Neurobiology of Language*, 1(1):104–134.
- Joakim Nivre and Chiao-Ting Fang. 2017. Universal dependency evaluation. In *Proceedings of the NoDaLiDa 2017 Workshop on Universal Dependencies (UDW 2017)*, pages 86–95.
- Joe O'Connor and Jacob Andreas. 2021. What context features can transformer language models use? *arXiv preprint arXiv:2106.08367*.
- Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32:8026– 8037.
- Thang M Pham, Trung Bui, Long Mai, and Anh Nguyen. 2020. Out of order: How important is the sequential order of words in a sentence in natural language understanding tasks? *arXiv preprint arXiv:2012.15180*.
- Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. *URL https://s3-us-west-2. amazonaws. com/openaiassets/researchcovers/languageunsupervised/language understanding paper. pdf*.
- Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*.
- 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. *arXiv preprint arXiv:1910.10683*.

- Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know What You Don't Know: Unanswerable Questions for SQuAD. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 784–789.
- Ori Ram, Yuval Kirstain, Jonathan Berant, Amir Globerson, and Omer Levy. 2021. Few-shot question answering by pretraining span selection. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3066–3079.
- Hassan Ramchoun, Youssef Ghanou, Mohamed Ettaouil, and Mohammed Amine Janati Idrissi. 2016. Multilayer perceptron: Architecture optimization and training. *International Journal of Interactive Multimedia and Artificial Intelligence . . .* .
- Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. 2020. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 3505–3506.
- Manuela Sanguinetti and Cristina Bosco. 2015. Parttut: The turin university parallel treebank. In *Harmonization and development of resources and tools for italian natural language processing within the parli project*, pages 51–69. Springer.
- Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. 2008. The graph neural network model. *IEEE transactions on neural networks*, 20(1):61–80.
- Natalia Silveira, Timothy Dozat, Marie-Catherine de Marneffe, Samuel Bowman, Miriam Connor, John Bauer, and Christopher D. Manning. 2014. A gold standard dependency corpus for English. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC-2014)*.
- Koustuv Sinha, Robin Jia, Dieuwke Hupkes, Joelle Pineau, Adina Williams, and Douwe Kiela. 2021a. Masked language modeling and the distributional hypothesis: Order word matters pre-training for little. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 2888–2913.
- Koustuv Sinha, Prasanna Parthasarathi, Joelle Pineau, and Adina Williams. 2021b. Unnatural language inference. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 7329–7346.
- Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2019. Mass: Masked sequence to sequence pre-training for language generation. *arXiv preprint arXiv:1905.02450*.
- 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.
- Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2019a. Superglue: A stickier benchmark for general-purpose language understanding systems. *arXiv preprint arXiv:1905.00537*.
- 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.
- Wei Wang, Bin Bi, Ming Yan, Chen Wu, Jiangnan Xia, Zuyi Bao, Liwei Peng, and Luo Si. 2019b. Structbert: Incorporating language structures into pre-training for deep language understanding. In *International Conference on Learning Representations*.
- 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.
- Thomas Wolf, Julien Chaumond, Lysandre Debut, Victor Sanh, Clement Delangue, Anthony Moi, Pierric Cistac, Morgan Funtowicz, Joe Davison, Sam Shleifer, et al. 2020. Transformers: State-of-theart natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45.
- Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.
- Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2018. How powerful are graph neural networks? In *International Conference on Learning Representations*.
- Atsuki Yamaguchi, George Chrysostomou, Katerina Margatina, and Nikolaos Aletras. 2021. Frustratingly simple pretraining alternatives to masked language modeling. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3116–3125.
- Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in neural information processing systems*, pages 5754–5764.
- Michihiro Yasunaga, Hongyu Ren, Antoine Bosselut, Percy Liang, and Jure Leskovec. 2021. Qa-gnn: Reasoning with language models and knowledge graphs for question answering. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 535–546.
- Amir Zeldes. 2017. [The GUM corpus: Creating mul](https://doi.org/http://dx.doi.org/10.1007/s10579-016-9343-x)[tilayer resources in the classroom.](https://doi.org/http://dx.doi.org/10.1007/s10579-016-9343-x) *Language Resources and Evaluation*, 51(3):581–612.
- Xikun Zhang, Antoine Bosselut, Michihiro Yasunaga, Hongyu Ren, Percy Liang, Christopher D Manning, and Jure Leskovec. 2021. Greaselm: Graph reasoning enhanced language models. In *International Conference on Learning Representations*.
- Junru Zhou and Hai Zhao. 2019. Head-driven phrase structure grammar parsing on penn treebank. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2396– 2408.
- 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 *The IEEE International Conference on Computer Vision (ICCV)*.
- Yutao Zhu, Kun Zhou, Jian-Yun Nie, Shengchao Liu, and Zhicheng Dou. 2021. Neural sentence ordering based on constraint graphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

A Experimental Protocol

A.1 Pretraining Implementation Details

Following [\(Devlin et al.,](#page-9-0) [2019\)](#page-9-0), we use BERT-baseuncased architecture (12 layers and model and 768 hidden size) as a backbone for all models. Also, we use the same 32k WordPiece [\(Wu et al.,](#page-11-13) [2016\)](#page-11-13) vocabulary and WikiBooks corpus of [\(Devlin et al.,](#page-9-0) [2019\)](#page-9-0). More precisely, we use English Wikipedia and BookCorpus [\(Zhu et al.,](#page-11-14) [2015\)](#page-11-14), that we obtain from the datasets library [\(Lhoest et al.,](#page-9-19) [2021\)](#page-9-19).

Each model is pretrained on a single GPU server that consists of 8 NVIDIA Tesla V100 cards with 32GB of memory. The pre-training code is based on the PyTorch [\(Paszke et al.,](#page-10-13) [2019\)](#page-10-13) version of the Transformers library [\(Wolf et al.,](#page-11-15) [2020\)](#page-11-15). We use the AdamW [\(Loshchilov and Hutter,](#page-9-20) [2017\)](#page-9-20) optimizer with a learning rate decay setting the initial learning rate to 1e-4 with 10,000 warm-up steps.

To speed up the pretraining in our experiments, we use mixed-precision training [\(Micikevicius](#page-10-14) [et al.,](#page-10-14) [2018\)](#page-10-14), and DeepSpeed library [\(Rasley et al.,](#page-10-15) [2020\)](#page-10-15). In addition, we train all models on full sequences (no padding) of 128 of length, and set the maximum per-GPU batch size for each model, which is 260 for MLM models and 390 otherwise. However, all models are fairly pretrained for 35 epochs over the pretraining data. We ensure this by setting the gradient accumulation step to 2 and 3 when the batch size is set to 390 and 260 respectively. Pretraining experiments took approximately take 5 days for the slowest models (ones with MLM).

Following [\(Devlin et al.,](#page-9-0) [2019\)](#page-9-0), we use a probability of 15% when pretraining with MLM objective (BERT-M and BERT-MT models). We search TOR probability lambda from {0.3, 0.5, 1.0} on the BERT-T model and found 0.5 to work the best. Therefore, we use a value of $lambda = 0.5$ with to the three models using TOR. On top of BERT encoder, the BERT+GIN-T model uses three GIN encoders with context windows $w = \{1, 2, 4\}$ and L^w ={2, 3, 5} number of layers respectively. $\varepsilon_{(.)}$ are always set to 0, while layer numbers and window sizes where selected empirically based trade-off between performance a pretraining latency, which is inspired from [\(Zhu et al.,](#page-11-8) [2021\)](#page-11-8).

A.2 Fine-Tuning Datasets

We experiment on 8 tasks from the GLUE benchmark [\(Wang et al.,](#page-11-2) [2018\)](#page-11-2): 2 single-sentence (CoLA and SST-2), one regression (STS-B), and

5 sentence-pair (MRPC, RTE, QQP, QNLI, and MNLI) classification tasks. Following prior works, we report Pearson correlation on STS-B, Matthews correlation on CoLA, F1 score on MRPC, and use the accuracy otherwise. We also report the unweighted average sum over the 7 tasks.

For Dependency parsing, we evaluate models on the well established English Penn Treebank (PTB) [\(De Marneffe and Manning,](#page-9-21) [2008\)](#page-9-21) corpus using the train/dev/test split of [\(Chen and Man](#page-8-12)[ning,](#page-8-12) [2014\)](#page-8-12). Also, we run experiments on 5 Universal Dependency [\(McDonald et al.,](#page-10-16) [2013\)](#page-10-16) corpora: EWT [\(Silveira et al.,](#page-10-17) [2014\)](#page-10-17), PARTUT [\(San](#page-10-18)[guinetti and Bosco,](#page-10-18) [2015\)](#page-10-18), GUM [\(Zeldes,](#page-11-16) [2017\)](#page-11-16), LINES [\(Ahrenberg,](#page-8-13) [2007\)](#page-8-13), and $ATIS⁷$ $ATIS⁷$ $ATIS⁷$. We report the Labeled Attachment Score (LAS) score [\(Nivre](#page-10-19) [and Fang,](#page-10-19) [2017\)](#page-10-19) for each corpus, as well as the unweighted average sum over the six corpora. Each DP corpus is already have its default train/dev/test splits.

A.3 Fine-Tuning Implementation Details

Following [\(Devlin et al.,](#page-9-0) [2019\)](#page-9-0), we use the representation of the [CLS] token of the last layer as input for GLUE classification tasks. For dependency parsing, we first use the last layer representation of the first sub-token of each word as input for Biaffine classifier [\(Dozat and Manning,](#page-9-22) [2016\)](#page-9-22), which in turn generates the arcs and relation types between words. Then, we use greedy decoding to get the final dependency parsing tree.

For full dataset experiments, we set the batch size to 32, learning rate to 2e-5, and the dropout rate of 0.1. We train all models under all settings for a maximum of 20 epochs and use early stopping. We report the average and standards deviation over 5 runs with different random seed.

We simulate a low resource setting for both GLUE and Dependency Parsing by randomly sampling tiny subsets of {16, 32, 64, 128, 256} examples of the training data. We report the average and standard deviation of 5 randomly selected folds. We use a batch size of 1 when training on low resource setting, as we find it to systematically work the best across all models.

B Results

⁷ [https://github.com/UniversalDependencies/UD_](https://github.com/UniversalDependencies/UD_English-ATIS) [English-ATIS](https://github.com/UniversalDependencies/UD_English-ATIS)

Table 4: Dev GLUE performances across training set sizes. BERT-ORG and BERT-5D8G respectively refer to the original BERT-base model of [\(Devlin et al.,](#page-9-0) [2019\)](#page-9-0) and to the MLM one of [\(Yamaguchi et al.,](#page-11-6) [2021\)](#page-11-6) pretrained during 5 days with 8 GPUs.

Model	PTB	EWT	PARTUT	ATIS	GUM	LINES	Avg.
16 Examples							
BERT-M	45.1 ± 0.8	33.4 ± 1.2	41.7 ± 1.5	65.6 ± 1.6	31.2 ± 2.1	34.4 ± 0.8	41.9 ± 1.3
BERT-MT	48.6 ± 0.8	33.7 ± 1.7	45.2 ± 1.2	65.6 ± 1.5	32.6 ± 2.5	37.6 ± 0.9	43.9±1.4
BERT-T	36.9 ± 1.0	24.1 ± 1.0	34.7 ± 1.3	56.5 ± 2.6	22.3 ± 1.9	28.8 ± 1.0	33.9 ± 1.5
BERT+GIN-T	56.2 ± 0.8	46.8 ± 0.9	55.0 ± 0.9	69.4 ± 1.5	47.1 ± 2.4	49.5 \pm 0.6	54.0 ± 1.2
32 Examples							
BERT-M	61.7 ± 1.4	48.1 ± 0.4	59.9 ± 0.5	74.7 ± 0.6	49.5 ± 1.5	50.2 ± 0.9	57.3 ± 0.9
BERT-MT	63.6 ± 1.1	49.4 ± 0.8	62.6 ± 0.7	74.5 ± 0.6	52.1 ± 1.4	52.6 ± 0.9	59.1 ± 0.9
BERT-T	54.1 ± 1.2	40.5 ± 1.0	52.6 ± 0.6	69.0 ± 0.6	41.4 ± 1.5	44.0 ± 1.1	50.3 ± 1.0
BERT+GIN-T	66.5 ± 1.0	58.1 ± 0.5	65.5 ± 0.7	77.6 ± 0.7	60.2 ± 1.0	59.4 ± 0.4	64.5 ± 0.7
64 Examples							
BERT-M	73.8 ± 0.7	61.4 ± 0.4	73.8 ± 0.5	79.9 ± 0.6	64.3 ± 1.0	62.8 ± 0.6	69.3 ± 0.6
BERT-MT	74.5 ± 0.4	62.0 ± 0.7	74.8 ± 0.7	79.7 \pm 0.5	65.6 ± 0.8	64.0 ± 0.3	70.1 ± 0.6
BERT-T	68.2 ± 0.6	55.6 ± 0.9	67.3 ± 0.9	77.0 ± 0.3	57.4 ± 0.8	57.6 ± 0.3	63.8 ± 0.6
BERT+GIN-T	74.7 \pm 0.4	66.4 ± 0.5	75.1 ± 0.7	81.0 ± 0.5	69.2 ± 0.4	67.1 \pm 0.4	72.3 ± 0.5
128 Examples							
BERT-M	80.5 ± 0.4	71.8 ± 0.5	80.8 ± 0.5	82.9 ± 0.4	74.0 ± 0.8	71.7 ± 0.3	77.0 ± 0.4
BERT-MT	80.4 ± 0.3	72.0 ± 0.3	81.1 ± 0.3	82.9 ± 0.2	74.3 ± 0.5	71.4 ± 0.2	77.0±0.3
BERT-T	76.7 ± 0.3	67.1 ± 0.2	76.9 ± 0.2	81.8 ± 0.2	69.1 ± 0.7	66.6 ± 0.3	73.0 ± 0.3
BERT+GIN-T	80.4 ± 0.3	73.6 ± 0.3	80.3 ± 0.4	84.1 ± 0.2	75.8 ± 0.4	72.9 ± 0.3	77.8 ± 0.3
256 Examples							
BERT-M	85.2 ± 0.1	78.1 ± 0.3	84.0 ± 0.4	85.2 ± 0.3	80.3 ± 0.2	77.5 ± 0.3	81.7 ± 0.3
BERT-MT	85.2 ± 0.2	78.1 ± 0.2	84.8 ± 0.3	84.9 ± 0.2	80.6 ± 0.2	77.4 ± 0.2	81.8 ± 0.2
BERT-T	82.9 ± 0.2	74.0 ± 0.4	82.6 ± 0.1	83.7 ± 0.1	77.3 ± 0.2	74.2 ± 0.2	79.1±0.2
BERT+GIN-T	84.8 ± 0.2	78.4 ± 0.2	84.1 ± 0.1	85.9 ± 0.2	80.9 ± 0.2	77.7 ± 0.2	82.0 ± 0.2
Full Dataset Examples							
BERT-M	94.2 ± 0.0	90.6 ± 0.0	89.3 ± 0.1	89.8 ± 0.1	91.3 ± 0.0	86.4 ± 0.1	90.3 ± 0.1
BERT-T	94.0 ± 0.0	90.1 ± 0.0	88.3 ± 0.2	89.6 ± 0.1	90.9 ± 0.0	86.2 ± 0.1	89.9 ± 0.1
BERT-MT	94.2 ± 0.0	90.8 ± 0.0	89.6 ± 0.1	89.8 ± 0.1	91.5 ± 0.0	87.2 ± 0.0	90.5 ± 0.1
BERT+GIN-T	94.1 ± 0.0	90.8 ± 0.0	89.4 ± 0.1	90.0 ± 0.1	91.6 ± 0.0	87.2 ± 0.1	90.5 ± 0.1

Table 5: Average Dev performance LAS across 5 dependency parsing datasets and training set sizes.

Model	PTB	EWT	PARTUT	ATIS	GUM	LINES	Avg.
16 Examples							
BERT-M	45.0 ± 0.8	33.8 ± 1.4	42.6 ± 0.9	65.8 ± 1.4	32.4 ± 1.9	36.0 ± 0.7	42.6 ± 1.2
BERT-MT	48.5 ± 0.7	34.2 ± 1.8	46.8 ± 1.6	65.7 ± 1.5	34.2 ± 2.3	39.1 ± 0.8	44.7 ± 1.5
BERT-T	36.9 ± 1.0	24.5 ± 1.1	36.6 ± 1.1	56.4 ± 2.7	23.4 ± 1.9	29.6 ± 1.1	34.6 ± 1.5
BERT+GIN-T	56.0 ± 0.5	46.9 ± 1.0	57.6 ± 1.1	69.7 ± 1.4	48.6 ± 2.4	50.6 ± 0.7	54.9 ± 1.2
32 Examples							
BERT-M	61.6 ± 1.4	48.4 ± 0.4	61.8 ± 0.7	76.2 ± 0.6	50.3 ± 1.4	52.6 ± 0.7	58.5 ± 0.9
BERT-MT	63.7 ± 1.1	49.6 ± 0.8	63.9 ± 0.4	75.8 ± 0.8	52.9 ± 1.4	54.6 ± 0.8	60.1 ± 0.9
BERT-T	54.3 ± 1.3	41.0 ± 1.2	55.9 ± 0.8	69.9 ± 0.6	42.5 ± 1.3	45.6 ± 1.1	51.5 ± 1.0
BERT+GIN-T	66.7 ± 1.0	58.5 ± 0.6	68.5 ± 0.7	78.9 ± 0.7	60.9 ± 0.9	60.9 ± 0.6	65.7 ± 0.7
64 Examples							
BERT-M	74.0 ± 0.6	61.7 ± 0.4	75.5 ± 0.6	82.5 ± 0.6	65.3 ± 0.9	65.4 ± 0.6	70.7 ± 0.6
BERT-MT	74.8 ± 0.3	62.1 ± 0.8	75.4 ± 0.7	82.1 ± 0.6	66.5 ± 0.8	66.2 ± 0.4	71.2 ± 0.6
BERT-T	68.4 ± 0.4	55.9±0.8	69.7 ± 0.8	79.9 \pm 0.4	58.9 ± 0.8	59.7 \pm 0.4	65.4 ± 0.6
BERT+GIN-T	75.0 ± 0.3	66.4 ± 0.4	76.6 ± 0.4	83.6 ± 0.9	70.1 ± 0.4	69.2 ± 0.5	73.5 ± 0.5
128 Examples							
BERT-M	80.8 ± 0.3	71.8 ± 0.5	82.3 ± 0.2	86.0 ± 0.3	74.9±0.7	74.0 ± 0.3	78.3±0.4
BERT-MT	80.7 ± 0.2	71.7 ± 0.3	81.5 ± 0.3	85.8 ± 0.6	75.4 ± 0.5	73.7 ± 0.3	78.1 ± 0.4
BERT-T	77.2 ± 0.2	67.3 ± 0.3	78.5 ± 0.5	85.3 ± 0.3	70.4 ± 0.7	68.9 ± 0.5	74.6±0.4
BERT+GIN-T	80.9 ± 0.2	73.6 ± 0.3	81.8 ± 0.2	87.5 ± 0.3	77.0 ± 0.4	74.5 ± 0.2	79.2 ± 0.3
256 Examples							
BERT-M	85.5 ± 0.1	78.2 ± 0.2	85.3 ± 0.2	88.1 ± 0.2	81.0 ± 0.4	79.5 ± 0.3	82.9 ± 0.2
BERT-MT	85.5 ± 0.3	78.0 ± 0.2	84.7 ± 0.4	88.1 ± 0.1	81.5 ± 0.3	79.3 ± 0.2	82.8 ± 0.3
BERT-T	83.3 ± 0.3	74.4 ± 0.3	83.0 ± 0.5	87.5 ± 0.2	78.2 ± 0.3	76.3 ± 0.1	80.4 ± 0.3
BERT+GIN-T	85.3 ± 0.3	78.2 ± 0.2	84.8 ± 0.3	89.1 \pm 0.2	81.8 ± 0.3	79.3 ± 0.2	83.1 ± 0.2
Full Dataset Examples							
BERT-M	94.7 ± 0.0	90.0 ± 0.0	89.9 ± 0.1	92.3 ± 0.2	90.3 ± 0.0	86.9 ± 0.0	90.7 ± 0.1
BERT-MT	94.7 ± 0.0	90.4 ± 0.0	90.2 ± 0.2	92.3 ± 0.1	90.7 ± 0.0	87.3 ± 0.0	90.9 ± 0.1
BERT-T	94.6 ± 0.0	89.8 ± 0.0	89.0 ± 0.1	92.5 ± 0.2	89.9 ± 0.1	86.6 ± 0.1	90.4 ± 0.1
BERT+GIN-T	94.7 ± 0.0	90.4 ± 0.0	89.8 ± 0.1	92.6 ± 0.2	90.8 ± 0.0	87.3 ± 0.0	90.9 ± 0.1

Table 6: Average Test performance LAS across 5 dependency parsing datasets and training set sizes.

Model	CoLA	SST-2	MRPC	STS-B	QQP	MNLI	QNLI	RTE	Avg.
1-Gram Shuffle									
BERT-M	1.0 ± 1.4	81.4 ± 0.2	66.5 ± 0.5	86.8 ± 0.0	83.1 ± 0.1	69.1 ± 0.1	81.3 ± 0.1	52.6 ± 0.3	65.2 ± 0.3
BERT-MT	1.9 ± 0.8	80.9 ± 0.4	61.5 ± 0.9	85.8 ± 0.1	83.6 ± 0.1	70.4 ± 0.1	80.1 ± 0.2	55.1 ± 0.8	64.9 ± 0.4
BERT-T	2.3 ± 0.9	82.5 ± 0.1	69.7 ± 0.7	85.9 ± 0.1	83.9 \pm 0.0	72.8 ± 0.1	83.1 ± 0.1	58.5 ± 0.6	67.3 ± 0.3
BERT+GIN-T	7.4 ± 1.0	82.8 ± 0.3	65.0 ± 1.3	86.7 ± 0.1	84.6 ± 0.1	72.5 ± 0.1	82.2 ± 0.2	60.9 ± 0.8	67.8 ± 0.5
2-Gram Shuffle									
BERT-M	20.5 ± 1.3	84.6 ± 0.3	69.6 ± 0.8	87.4 ± 0.1	86.0 ± 0.1	74.0 ± 0.1	83.8 ± 0.2	53.6 ± 0.8	69.9 ± 0.5
BERT-MT	20.6 ± 1.1	83.6 ± 0.2	67.5 ± 1.0	86.2 ± 0.1	86.0 ± 0.0	74.5 ± 0.1	83.0 ± 0.2	58.3 ± 0.7	70.0 ± 0.4
BERT-T	22.1 ± 1.8	84.5 ± 0.5	72.6 ± 0.6	86.2 ± 0.1	85.6 ± 0.0	75.2 ± 0.1	84.5 ± 0.2	58.2 ± 0.8	71.1 ± 0.5
BERT+GIN-T	24.9 ± 1.7	85.6 ± 0.2	68.5 ± 0.5	87.1 ± 0.1	86.2 ± 0.1	75.3 ± 0.1	83.9 ± 0.1	61.4 ± 1.1	71.6 ± 0.5
3-Gram Shuffle									
BERT-M	33.0 ± 1.5	85.8 ± 0.5	71.3 ± 1.3	87.4 ± 0.0	86.9 ± 0.0	76.2 ± 0.1	85.3 ± 0.1	58.2 ± 0.4	73.0 ± 0.5
BERT-MT	32.9 ± 0.6	85.2 ± 0.4	70.0 ± 0.7	86.3 ± 0.1	86.9 ± 0.1	76.8 ± 0.1	84.8 ± 0.1	59.6 ± 1.1	72.8 ± 0.4
BERT-T	34.0 ± 0.6	85.6 ± 0.2	74.5 ± 0.9	86.0 ± 0.0	86.3 ± 0.1	76.8 ± 0.0	85.5 ± 0.1	59.3 ± 0.6	73.5 ± 0.3
BERT+GIN-T	36.8 ± 0.5	85.9 ± 0.4	68.9 ± 0.3	86.9 ± 0.1	86.8 ± 0.0	76.7 ± 0.1	84.8 ± 0.1	62.2 ± 0.4	73.6 ± 0.2
4-Gram Shuffle									
BERT-M	40.7 ± 1.2	87.1 ± 0.4	72.2 ± 0.9	87.5 ± 0.1	87.5 ± 0.0	78.1 ± 0.1	86.4 ± 0.1	60.1 ± 1.0	74.9±0.5
BERT-MT	43.5 ± 0.5	85.6 ± 0.2	74.0±1.2	86.2 ± 0.1	87.4 ± 0.0	78.3 ± 0.1	85.6 ± 0.2	63.2 ± 0.6	75.5 ± 0.4
BERT-T	40.8 ± 1.3	85.3 ± 0.3	76.7 ± 0.6	85.9 ± 0.1	86.7 ± 0.0	77.8 ± 0.1	86.0 ± 0.0	59.2 ± 0.9	74.8 ± 0.4
BERT+GIN-T	42.5 ± 0.7	86.2 ± 0.3	72.3±0.7	86.9 ± 0.0	87.3 ± 0.0	77.7 ± 0.1	85.4 ± 0.1	63.0 ± 0.9	75.2 ± 0.4
5-Gram Shuffle									
BERT-M	46.3 ± 0.5	87.9 ± 0.2	73.3 ± 0.9	87.7 ± 0.1	88.1 ± 0.0	79.3 ± 0.0	87.3 ± 0.1	60.2 ± 0.5	76.3 ± 0.3
BERT-MT	48.6 ± 0.7	87.3 ± 0.2	73.2 ± 0.7	86.6 ± 0.1	87.8 ± 0.1	78.9 ± 0.1	86.6 ± 0.1	59.8 ± 0.8	76.1 ± 0.3
BERT-T	45.2 ± 0.8	85.8 ± 0.3	76.0 ± 0.6	86.4 ± 0.1	87.1 ± 0.0	78.3 ± 0.1	86.1 ± 0.1	62.0 ± 0.9	75.9 ± 0.4
BERT+GIN-T	47.5 ± 0.4	87.3 ± 0.3	72.2±1.3	87.3 ± 0.1	87.7 ± 0.0	78.5 ± 0.1	86.1 ± 0.2	62.6 ± 0.8	76.2 ± 0.4

Table 7: Dev GLUE performances and standards deviation (we run experiments on 5 different seeds) across word shuffling n-grams.

Table 8: Dev GLUE performances and standards deviation (we run experiments on 5 different seeds) across masked sequences.

Table 9: Pairwise token order accuracy and standards deviation on GLUE dev sets. % indicate lambda value applied on input sequences, we run experiments on 5 different seeds.

Table 10: Pairwise Token order accuracy and standards deviation on Dependency parsing datasets. % indicate lambda value applied on input sequences, we run experiments on 5 different seeds.