Refining Pre-trained NLP Models Through Shuffled-token Detection

Subhadarshi Panda
Graduate Center CUNY
spanda@gc.cuny.edu

Anjali Agrawal **Jeewon Ha** **Benjamin Bloch**
New York University
{aa7513,jh6926,bb1976}@nyu.edu

Abstract

State-of-the-art transformer models have achieved robust performance on a variety of NLP tasks. Many of these approaches have employed domain agnostic pre-training tasks to train models that yield highly generalized sentence representations that can be fine-tuned for specific downstream tasks. We propose refining a pre-trained NLP model using the objective of detecting shuffled tokens[1]. We use a sequential approach by starting with the pre-trained RoBERTa model and training it using our approach. Applying random shuffling strategy on the word-level, we found that our approach enables the RoBERTa model achieve better performance than our baseline on 5 out of 7 GLUE tasks. Our results indicate that learning to detect shuffled tokens is a promising approach to learn more coherent sentence representations.

1 Introduction

The method of pre-training natural language models has been shown to greatly improve model performance on a wide range of NLP tasks (Peters et al., 2018; Radford et al., 2018; Howard and Ruder, 2018). State-of-the-art models that utilize transformers and deep bi-directional representations of text such as BERT, RoBERTa, and ALBERT (Devlin et al., 2019; Liu et al., 2019; Lan et al., 2019) have achieved superior results by pre-training on general, large corpora to learn rich representations from unlabeled data. Particularly helpful in low training data resource scenarios, unsupervised pre-training has become the first step for many language models to build powerful linguistic representations before fine tuning for downstream target tasks.

BERT style models use masked language modeling (MLM) and sometimes next sentence prediction, as pre-training tasks. While these tasks have been shown to produce transferable sentence representations for many NLP tasks, using additional domain-agnostic pre-training tasks such as sentence shuffling may improve model performance. In a seminal cognitive psychology study, McCusker et al. (1981) demonstrated that humans have a well trained ability to parse shuffled sentences. Shuffling as a pre-training task may therefore help expand transformer models to achieve even better performance on NLP tasks.

In this paper, we propose that pre-training the RoBERTa with token modification discrimination head on randomly shuffled sentences provides constructive learning objective, which helps the model learn coherent sentence representations and facilitate model recognition of the key pieces of a sentence and their association. To substantiate the argument, we design experiments to examine the model performance of RoBERTa with the proposed approach. The results demonstrate that pre-training the model with shuffled sentences enhances the scores of a majority of GLUE tasks.

2 Related Work

Shuffling sentences and words has often been used as a downstream task to evaluate model performance. One relevant example is the work by Sakaguchi et al. (2017) to develop a semi-character RNN

[1]Code is available in https://github.com/subhadarship/learning-to-unjumble

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

model that surpasses previous spell-check methodologies on the *Cmadbrigde Uinervtisy* effect, where humans can easily reconstruct the shuffled token. Yang and Gao (2019) explored the performance of BERT on a shuffled sentence downstream task and highlighted some induced bias in the model that is the cause of incorrect predictions for noisy inputs. While the authors propose removing the induced bias from the representations to improve results, they do not consider the possibility of pre-training the model with shuffled sentences.

The use of un-ordered or noisy data in model training itself has proven effective. A number of studies have focused on using shuffled input to create useful sentence representation vectors for language models. Kiros et al. (2015) developed the skip-thoughts method to accomplish the task of reconstructing sentence order from a shuffled input. The authors used an encoder-decoder RNN model at the sentence level that allows a sentence to predict the adjacent sentences.

Logeswaran et al. (2016) explored how sentence ordering tasks can help models learn text coherence. Using an RNN based approach, they train models to identify the correct ordering of sentences and show that models learn both document structure and useful sentence representations during this task. Jernite et al. (2017) employed discourse based learning objectives to help models understand discourse coherence. Specifically, given some sentences, they ask the model to predict if the sentences are in order, or if one sentence comes next to a set of sentences, or to predict the conjunction that joins the sentences. They showed that using these objectives to train models achieves significant reduction in computational training costs and is also effective when using unlabeled data.

There are a number of papers that focus on word-level shuffling, as opposed to sentence-level shuffling. Hill et al. (2016) developed the Sequential Denoising Autoencoder (SDAE) method, where a sentence is corrupted using a noise function determined by free parameters. After a certain percentage of words have been corrupted, an LSTM encoder-decoder model is tasked with predicting the original sentence from the corrupted version. The authors demonstrate training with noisy inputs allowed SDAE to significantly outperform regular SAE models, which did not introduce word-level-noise factors.

One closely related paper in the field of computer vision leverages the use of shuffled input in model training. Noroozi and Favaro (2016) employ a CNN model that is trained to solve jigsaw puzzles to determine correct spatial representation. Their results show that using shuffled input helps models learn that images are made up of different parts, and their relationship to the whole.

Finally, a variety of studies demonstrate that further pre-training performed after the general purpose BERT pre-training leads to better model results instead of simply fine-tuning downstream. Domain specific pre-training, such as BioBERT (Lee et al., 2019), story ending prediction by TransBERT (Li et al., 2019), and video caption classification by videoBERT (Sun et al., 2019) are all examples where expanding the pre-training tasks for BERT has achieved enhancement in model performance. TransBERT in particular demonstrates that further pre-training using targeted supervised tasks achieves better results than relying only on the unsupervised pre-training in BERT.

3 Methodology

Consider a sequence of tokens $\boldsymbol{x}$. We first obtain $\boldsymbol{x}^{\text{shuffled}}$ from $\boldsymbol{x}$ by shuffling a set of tokens of $\boldsymbol{x}$. Given $\boldsymbol{x}^{\text{shuffled}}$, we detect if tokens are shuffled or not by using a token modification discrimination head on top of the RoBERTa `base` model. Our choice of the discriminative head is motivated by the recent success of ELECTRA (Clark et al., 2019). We built our model using HuggingFace transformers (Wolf et al., 2019). For our baseline model, we trained the RoBERTa model with the token modification discrimination head for detecting masked tokens for the same number of steps as the proposed approach for a fair comparison.

Data We extracted $133,295$ articles from Wikidump[2]. We used each paragraph in the extracted text as a data sample for our model. We filtered out samples that were either spaces-only or had more than 512 tokens after tokenizing with the pretrained `RobertaTokenizer` of the `roberta-base` model. We finally randomly split the samples into $1.3M$ for training and $14K$ for validation. We used the same Wikidump dataset for training the baseline model as well.

[2]Timestamp May 9th, 2020. We used the scripts from `https://github.com/NVIDIA/DeepLearningExamples/tree/master/PyTorch/LanguageModeling/BERT#getting-the-data` to extract the data.

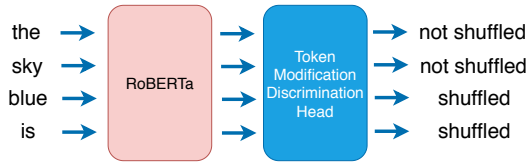

Figure 1: Illustration of our model for detecting shuffled tokens. The original sentence is "the sky is blue".

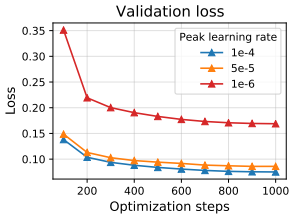

Figure 2: Validation loss as training progresses

Task →	CoLA	SST-2	MRPC	STS-B	QNLI	RTE	WNLI
Metric →	Matthew's corr.	Accuracy	F1 score	Spearman corr	Accuracy	Accuracy	Accuracy
Plain pre-trained RoBERTa	0.557	0.946	0.901	**0.896**	**0.928**	0.661	0.423
Baseline	0.508	**0.950**	0.869	0.888	0.924	0.631	**0.563**
Shuffled-token detection (p=0.15)	**0.621**	0.92	0.905	0.886	**0.928**	**0.704**	0.437
Shuffled-token detection (p=0.3)	0.565	0.943	**0.906**	0.894	0.925	0.675	0.451

Table 1: Results on GLUE tasks

3.1 Creating Shuffled Tokens for Training

We permute text sequences at the word level based on a probability p. We consider shuffling on a word level rather than a sub-word level. One straightforward approach is to create the shuffled tokens from a sequence and then use `RobertaTokenizer` to tokenize the shuffled sequence. However, this approach is problematic since the number of sub-words after tokenization may differ between the original and the shuffled sentence. In order to ensure that the sub-words belonging to a word stay intact and are not shuffled away, we create a mapping, which maps each sub-word to the corresponding word. Then, we tokenize the original sequence and shuffle the tokens based on the mapping so that all the sub-words belonging to a word occur together. Further, we define the target tensor which has binary labels for each token that specifies whether the token was shuffled or not. An illustrative example for constructing the shuffled sequence and the label has been outlined in Appendix A.

Shuffling strategy We randomly permute the words in a sequence based on a probability p for our experiments highlighted in Section 4. Note that fraction $\geq p$ of the input tokens would be shuffled since one or more input tokens (sub-words) belong to a single word.

3.2 RoBERTa Model with Token Modification Discrimination Head

Figure 1 shows an overview of our complete model. We use the RoBERTa model to map a sequence of input tokens $\boldsymbol{x}^{\text{shuffled}} = [x_1, \ldots, x_n]$ into a sequence of contexualized vectors $h(\boldsymbol{x}) = [h_1, \ldots, h_n]$. We add an token modification discriminator head based to classify each hidden representation h_i to 0 (if the token at i-th place is not shuffled) or 1 (if the token at the i-th place is shuffled). Specifically, the head contains two linear layers with parameters $\{W_A\}$ and $\{W_B\}$. First, for every hidden vector h_i, we compute $h_i^{'} = GELU(W_A^T h_i)$. Then, we compute the output of the model $D(\boldsymbol{x}^{\text{shuffled}}, i) = sigmoid(W_B^T h_i^{'})$. During training, we minimize the loss which is sum of the binary cross entropy loss for every token.

$$\mathcal{L}(\boldsymbol{x}, \theta) = \mathbb{E}\left(\sum_{i=1}^{n} -\mathbb{1}\left(x_i^{\text{shuffled}} = x_i\right) \log D\left(\boldsymbol{x}^{\text{shuffled}}, i\right) \quad -\mathbb{1}\left(x_i^{\text{shuffled}} \neq x_i\right) \log\left(1 - D\left(\boldsymbol{x}^{\text{shuffled}}, i\right)\right)\right)$$

4 Experiments

4.1 Implementation

All experiments have been performed using the RoBERTa model with the token modification discrimination head described in Section 3.2.

The hyperparameters used in our experiments follow the hyperparameters of the RoBERTa-base model except for the warmup steps, batch size, peak learning rate, and the maximum training steps.

For our experiments, we use 100 linear warmup steps followed by linear decay of the learning rate outlined in the Appendix in Figure 4. We use gradient accumulation and an effective batch size of 4096 for training.

Evaluation To find the optimal learning rate and the maximum steps, we evaluate our model using the validation loss. However, to compare the performance of our proposed approach with $p = 0.15, 0.3$, and that of the baseline model, we evaluate the approaches[3] on 7 GLUE tasks using the metrics outlined in Table 1. We use the same hyperparameters, outlined in the Appendix B, for the baseline as well as the proposed approach for an unbiased comparison.

4.2 Results & Analysis

The results for the validation loss with an increasing number of optimization steps for the different learning rates is illustrated in Figure 2. The training loss is outlined in Figure 3 in Appendix C. We observe that the minimum training loss, as well as validation loss, are achieved with the peak learning rate of 1e-4. Moreover, the training loss and the validation loss keeps on decreasing with the number of optimization steps which shows that training the model for more number of steps could be beneficial. The optimal maximum steps as depicted in Figure 3 and 2 is 1000[4]. Table 1 presents the results for the 7 GLUE tasks. Our model trained on randomly shuffled tokens, either with $p = 0.15$ or $p = 0.30$, performs better than the baseline in 5 of the 7 downstream tasks. The model performance based on the proposed approach on these tasks gives us insights about what aspects of natural language our model improved in learning.

Our model's performance on CoLA, which predicts grammatical correctness of a sentence, is better, indicating that the pre-traing task may have enhanced the model's ability to learn grammatical information. Moreover, better performance on RTE, MRPC and STS-B shows that with the proposed approach, the model better understands the semantic relationships such as similarity and entailment.

However, random shuffling hurts the performance of the model on WNLI significantly in comparison to the baseline. This may be due to the fact that WNLI forms a pair of sentences by replacing the ambiguous pronouns with their referents. Since we are shuffling the words, it is likely that the nouns will be shuffled, resulting in misleading replacement of the ambiguous pronoun.

Our baseline model outperforms the random shuffling approach on SST-2 task which predicts the sentiment polarity of the movie reviews. One possible explanation is that shuffling negations in presence of contrasting conjunctions can significantly change the sentiment associated with the sentence[5].

5 Conclusion & Future Work

In this paper, we examine the performance of a RoBERTa model with token modification discrimination head on detecting randomly shuffled tokens. We have demonstrated that detecting shuffled tokens is indeed a challenging yet advantageous task, which allows the model to learn coherent representations of the sentences. In this work, we start with pretrained `roberta-base` model and train it further on the shuffled token detection task.

For future work, the model can be further explored by expanding the shuffling strategy. One possible strategy is Part Of Speech (POS) shuffling, which randomly permutes specific POS tokens such as noun, verb, etc. We would also like to study our approach when applied to other pretrained models such as ALBERT and ELECTRA. The prediction task of the original indices of the shuffled tokens could be another future direction.

[3] We do not use the validation loss here since the validation loss can be higher for a difficult task even if the model is learning more about the natural language representation.

[4] An actual optimum number of steps could be more than 1000 and training further would give us the best value for the maximum steps.

[5] For instance, consider the sentence-"That movie was good but I did not watch it." A random shuffled sentence can be "The movie was not good but I did watch it."

Acknowledgments

We thank Sam Bowman for insightful discussions and suggestions. We also thank the anonymous reviewers for their useful comments and suggestions.

References

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*.

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, Minneapolis, Minnesota. Association for Computational Linguistics.

Felix Hill, Kyunghyun Cho, and Anna Korhonen. 2016. Learning distributed representations of sentences from unlabelled data. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1367–1377, San Diego, California. Association for Computational Linguistics.

Jeremy Howard and Sebastian Ruder. 2018. Fine-tuned language models for text classification. *CoRR*.

Yacine Jernite, Samuel R Bowman, and David Sontag. 2017. Discourse-based objectives for fast unsupervised sentence representation learning. *arXiv preprint arXiv:1705.00557*.

Ryan Kiros, Yukun Zhu, Russ R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 3294–3302. Curran Associates, Inc.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. Albert: A lite bert for self-supervised learning of language representations.

Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. 2019. Biobert: a pre-trained biomedical language representation model for biomedical text mining. *CoRR*.

Zhongyang Li, Xiao Ding, and Ting Liu. 2019. Story ending prediction by transferable BERT. *CoRR*.

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

Lajanugen Logeswaran, Honglak Lee, and Dragomir Radev. 2016. Sentence ordering using recurrent neural networks.

Leo McCusker, Philip Gough, and Randolph Bias. 1981. Word recognition inside out and outside in. journal of experimental psychology: Human perception and performance. *7(3):538–551*.

Mehdi Noroozi and Paolo Favaro. 2016. Unsupervised learning of visual representations by solving jigsaw puzzles. In *European Conference on Computer Vision*, pages 69–84. Springer.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.

Keisuke Sakaguchi, Kevin Duh, Matt Post, and Benjamin Van Durme. 2017. Robsut wrod reocginiton via semi-character recurrent neural network.

Chen Sun Sun, Austin Myers, Carl Vondrick, Kevin Murphy, and Cordelia Schmid. 2019. Videobert: A joint model for video and language representation learning. *CoRR*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, R'emi Louf, Morgan Funtowicz, and Jamie Brew. 2019. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771.

Runzhe Yang and Zhongqiao Gao. 2019. Can machines read jmulbed senetcnes?

A Example of Preparing The Shuffled Dataset

Consider an example sentence S1-

S1: "The birds are chirping."

Then, after tokenizing using the Roberta-base tokenizer, we would have a list of tokens as-

['The', 'Ġbirds', 'Ġare', 'Ġch', 'ir', 'ping', '.']

Since we want to shuffle on the word-level instead of the sub-word level, we create a mapping list of dimension equal to the length of the tokens such that each entry specifies the index of the word it belongs to. Here, the mapping list would be a 7-dimensional vector [0, 1, 2, 3, 3, 3, 3]. Then, the random shuffled sequence with probability $p = 0.5$ would be as follows -

['The', 'Ġch', 'ir', 'ping', '.', 'Ġare', 'Ġbirds']

In addition to the input tokens, we create a binary label for each token which specifies whether the token was shuffled or not. Here, the labels would be a 7-dimensional vector [0, 1, 1, 1, 1, 1, 1] where 1 means "shuffled" and 0 means "not shuffled".

B Hyperparameters Used for Fine-tuning

Table 2 outlines the fine-tuning hyperparameters used for fine-tuning the plain pre-trained RoBERTa, baseline, and the proposed model on the GLUE downstream tasks.

Hyperparameter	Value
Maximum Sequence Length	128
Batch Size	64
Learning Rate	2e-5
Number of epochs	3

Table 2: Hyperparameters for fine-tuning RoBERTa model

C Monitoring Learning Rate

Figure 4 shows the learning rate with the increase in optimization steps.

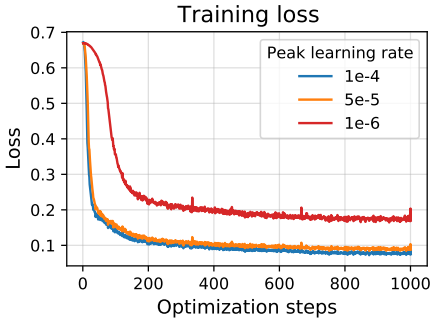

Figure 3: Training loss logged after every training step

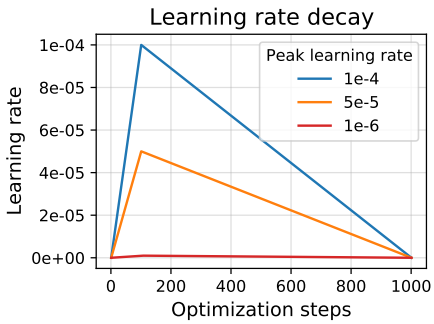

Figure 4: Learning rate as training progresses