Enhancing Language Generation with Effective Checkpoints of Pre-trained Language Model

Jeonghyeok Park and Hai Zhao[*]
Department of Computer Science and Engineering, Shanghai Jiao Tong University
Key Laboratory of Shanghai Education Commission for Intelligent Interaction
and Cognitive Engineering, Shanghai Jiao Tong University, Shanghai, China
MoE Key Lab of Artificial Intelligence, AI Institute, Shanghai Jiao Tong University
117033990011@sjtu.edu.cn, zhaohai@cs.sjtu.edu.cn

Abstract

This work empirically explores effective exploiting of intermediate output from pre-trained language models (PrLMs) for language generation tasks. For this purpose, we propose an improved method to integrate public checkpoints of PrLMs for the most convenience and perform extensive experiments on 6 different kinds of PrLMs, including BERT, ELECTRA, GPT2, Multi-lingual BERT, and XLM RoBERTa. Evaluation with automatic metrics shows that our approach significantly improves the generation quality on the generation tasks, up to 1.8 BLEU points for neural machine translation (Korean-to-English, Korean-to-Chinese) and 1.8 ROUGE points improvements for text summarization.

1 Introduction

Pre-trained Language Models (PrLMs), such as BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019), and ELECTRA (Clark et al., 2020), have thoroughly changed the landscape of state-of-the-art performance on many Natural Language Understanding (NLU) tasks. Also, publicly released checkpoints of the PrLMs allow natural language processing (NLP) researchers to gain SOTA results while saving vast compute and time resources. The widely used method to exploit PrLM is *fine-tuning*. However, for Natural Language Generation (NLG) tasks, such methods do not get as much performance gain as in the NLU task. Several previous studies proposed methods that better use prior knowledge of the PrLM for NLG tasks (Yang et al., 2020; Zhu et al., 2020; Chen et al., 2020). Expending the previous studies, in this paper, we propose an improved method that exploits the checkpoint of the PrLM into the Transformer models (Vaswani et al., 2017).

The existing methods for leveraging PrLM in NLG tasks can be roughly classified into two categories: *Reusing the PrLM as a starting point* and *Integrating the intermediate output of the PrLM*. The former, the widely used in various NLP tasks, denotes to initialize the part of Transformer from the PrLM for generation tasks (Clinchant et al., 2019; Edunov et al., 2019; Rothe et al., 2020) or replace the input embedding with the PrLM. The latter is an approach that first extracts the contextualized representation from a LM for an input sentence and fuses it into a neural model (Yang et al., 2020; Zhu et al., 2020; Chen et al., 2020). As our preliminary experiment shows, we expand this approach and explore in many ways towards better performance. In both of the preceding approaches, whether to freeze or fine-tun the parameter of PrLM is also an important issue. For the former (*Reusing the PrLM*), several works demonstrated that freezing the PrLM at training time led to a significant performance drop. Meanwhile, for the latter approach (*Integrating the PrLM*), prior studies adopted the whole or half-freezing instead of fine-tuning the parameters of the PrLMs. Yang et al. (2020) suggested that the reason why fine-tuning PrLM in neural machine translation (NMT) does not work as well as in other NLP tasks is due to the availability of large training data and the high capacity of baseline NMT models (i.e., Transformer), where excessive fine-tuning leads to the *catastrophic forgetting* phenomenon (Goodfellow et al., 2015). Also, Zhu et al. (2020) shows that freezing the BERT in NMT is better than fine-tuning with a large gap. This is in line with our experimental results. Thus in this empirical study, we freeze the parameters of the PrLMs in our experiments.

This paper focuses on finding an effective

[*] Corresponding author. This paper was partially supported by National Key Research and Development Program of China (No. 2017YFB0304100) and Key Projects of National Natural Science Foundation of China (U1836222 and 61733011). This work was supported by Huawei Noah's Ark Lab.

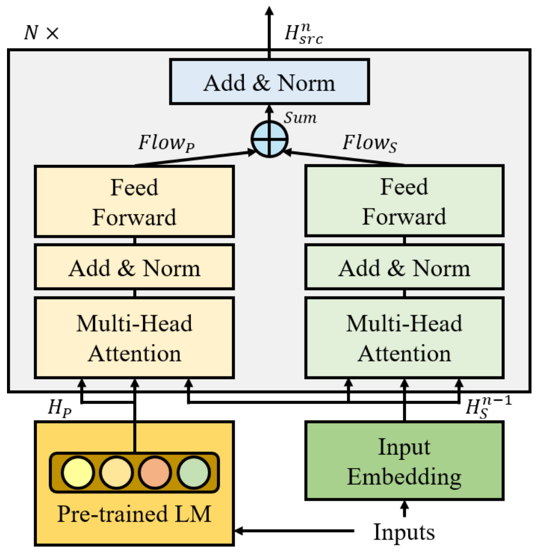

Figure 1: The architecture of our proposed model.

method integrating the intermediate output into the Transformer model to improve the generation quality, unlike the widely used methods such as initializing a part of Transformer and replacing the input embedding with the PrLM. To this end, similar to Zhu et al. (2020), we insert PrLM-dedicated modules that take the intermediate output from the PrLM and make the extra flow for PrLM in Transformer layers. This allows us to integrate PrLM into the Transformer without considering the PrLM's configuration such as modeling, dimension and vocabulary. Based on extensive empirical experiments, we finally adopt an improved method that uses the **Second-to-last (i.g., penultimate) hidden state** of the PrLM as the contextualized representation and proceeds, in only **Source-side (Encoder)**, **Summation** of the source input flow and the PrLM flow generated through the **PrLM-dedicated modules**. In our experiments, we use the publicly released checkpoints of 6 types of PrLMs: BERT, DistilBERT, ELECTRA, GPT2, Multilingual BERT, and XLM-RoBERTa. We release an implementation of our improved method for Korean language generation tasks [1].

2 Model

We propose a modified Transformer-encoder that effectively integrates publicly available checkpoints of PrLMs. Figure 1 shows the architecture of our proposed model. We add an extra flow for PrLM through additional PrLM-dedicated modules including `PrLMAttn`, `Add&Norm`, and `FFN`. Specific mathematical formulations are left at Appendix B. Given an input sequence $x_s = \{x_1, ..., x_N\}$, there is a PrLM-input sequence $x_p = \{x_1^p, ...x_M^p\}$ of length M splited by the PrLM-dedicated tokenizer. The PrLM-input sequence is fed to the PrLM for generating the PrLM representation $H_P = PrLM(x_p)$. Based on preliminary experiment, we adopt the second-to-last hidden state of the PrLM outputs as the contextualized representation. In our proposed encoder, the PrLM representation H_P is merged with the source flow to generate the output H_S^n of n^{th} encoder layer:

$$H_S^n = (Flow_P + Flow_S) + Attn_S, \qquad (1)$$

$$Flow_P = \texttt{FNN}(Attn_P + H_S^{n-1}), \qquad (2)$$

$$Attn_P = \texttt{PrLMAttn}(H_S^{n-1}, H_P, H_P), \qquad (3)$$

$$Flow_S = \texttt{FNN}(Attn_S + H_S^{n-1}), \qquad (4)$$

$$Attn_S = \texttt{Attn}(H_S^{n-1}, H_S^{n-1}, H_S^{n-1}), \qquad (5)$$

where `PrLMAttn` is the PrLM-dedicated attention module that takes the previous hidden state H_S^{n-1} as a query and the PrLM representation H_P as a key and a value and `Attn` is the original one. We adopt the summation strategy for merging the two different flows, and it gains better results than previous works such as gate network (Yang et al., 2020) and dropnet (Zhu et al., 2020).

3 Experiments

To demonstrate the effectiveness of the proposed method, we perform extensive experiments on two NMT and abstractive text summarization tasks. For translation, we use BLEU (Papineni et al., 2002) for the evaluation of translation quality, and for text summarization, we report unigram and bigram overlap (ROUGE-1 and ROUGE2) to assess informativeness, and the longest common subsequence (ROUGE-L) to assess fluency with ROUGE scores (Lin, 2004). All the model training is on a single NVIDIA Tesla V100 GPU (16130MiB, Google Colab).

3.1 Datasets and Experimental Setting

We evaluate our approach on language generation tasks such as translation and text summarization. For translation tasks, we use two machine translation datasets: AIHub Ko→En [2] (containing 1.6M

[1]https://github.com/tmtmaj/Exploiting-PrLM-for-NLG-tasks

[2]http://www.aihub.or.kr/

Systems	Ko→Ch	Ko→En
Transformer	30.35 (-)	41.19 (-)
(Zhu et al., 2020)	31.33 (+0.9)	41.97 (+0.7)
(Clinchant et al., 2019)	31.75 (+1.4)	41.55 (+0.4)
Korean-specific PrLM		
+KoBERT	31.20 (+0.8)	42.17 (+0.9)
+HanBERT	31.39 (+0.9)	42.03 (+0.8)
+DistilKoBERT	30.94 (+0.5)	41.91 (+0.7)
+ELEC. small	31.51 (+1.1)	42.20 (+1.0)
+ELEC. base	**32.17 (+1.8)**	**42.59 (+1.4)**
+KoGPT2	30.38 (+0.0)	41.74 (+0.5)
Multi-language PrLM		
+BERT cased	30.57 (+0.2)	41.17 (-0.0)
+BERT uncased	30.78 (+0.4)	41.22 (+0.0)
+RoBERTa base	31.09 (+0.7)	41.85 (+0.6)
+RoBERTa large	31.64 (+1.2)	42.01 (+0.8)

Table 1: Experimental results on translation tasks. Both Clinchant et al. (2019) and Zhu et al. (2020) use ELECTRA base.

Systems	R-1	R-2	R-L	△
Transformer	46.32	29.56	37.88	-
Oracle	57.17	44.00	44.46	-
Korean-specific PrLM				
+KoBERT	47.05	30.40	38.68	+0.8
+HanBERT	47.49	31.22	39.51	+1.5
+DistilKoBERT	46.64	29.86	38.43	+0.4
+ELEC. small	47.10	30.88	39.01	+1.1
+ELEC. base	**47.90**	**31.44**	**39.91**	**+1.8**
+KoGPT2	46.99	30.51	38.65	+0.8
Multi-language PrLM				
+BERT cased	46.44	29.61	38.11	+0.1
+BERT uncased	46.92	29.80	38.40	+0.4
+RoB. base	47.01	30.55	38.89	+0.9
+RoB. large	46.77	30.31	38.66	+0.6

Table 2: Experimental results on summarization task. △ denotes average improvements.

training, 3K development and 3K test sentence pairs) and Ko→Ch [3] (317K, 3K, 3K pairs). For text summarization, we use a news document summarization dataset (40K, 1K, 1K pairs) from two institutes [4]. In our experiment, we adopt a `base` Transformer (Vaswani et al., 2017) as the baseline model and use 6 different PrLMs: BERT, DistilBERT, ELECTRA, GPT2, Multi-lingual BERT, and XLM-RoBERTa. More details about the settings are included in Appendix A.

3.2 Experimental Results

Table 1 and 2 report the results of machine translation and abstractive text summarization task, respectively. For the experimental results, we made the following observation: (1) For all the tasks, our proposed methods outperform the strong baseline Transformer w.r.t BLEU (up to 1.8) and ROUGE (up to 1.8). (2) For PrLMs, **ELECTRA base** gains the most significant improvements. Note that **ELECTRA small** is smaller in size than other models, but it shows better performance. This means that factors other than the size of the PrLM have a greater influence on the generation quality. (3) Another observation is that in most cases, using Korean-specific PrLMs leads to better performance than using multi-language PrLMs.

4 Explorations for leveraging PrLM

Our proposed method for leveraging PrLM is to use the **Second-to-last hidden state** of the PrLM as the contextualized representation and proceeds, in only **Source-side (Encoder)**, **Summation** of the source input flow and the PrLM flow after the **FNN**. In this subsection, the setting is the default, and we change only the target part of each experiment. We conducted the following four analyses on the Korean-Chinese and Korean-English datasets.

4.1 Which hidden state of the PrLM to extract?

We evaluated the impact on how to extract the contextualized representation from the PrLM. As shown in Table 3a, using the second-to-last (i.e., penultimate) hidden state of the PrLM performs the best. It has also been demonstrated in Yang et al. (2020). Moreover, as another attempt, we dynamically extracted the hidden state of each layer based on sentence embedding of n^{th} layer, which can be gained by averaging the PrLM layer (known as PrLM embeddings, Dyn. [Aver] in Table 3a) or using the output of the first token (the [CLS] token, Dyn. [CLS]). However, they did not get a big performance boost.

[3]http://www.donga.com/ and http://semanticweb.kaist.ac.kr

[4]https://corpus.korean.go.kr/ and https://dacon.io/competitions/official/235673/overview/

Systems	Ko→Ch	Ko→En
Dyn. [CLS]	31.09 (+0.7)	41.44 (+0.3)
Dyn. [Aver]	31.17 (+0.8)	41.46 (+0.3)
Last	31.34 (+1.0)	41.99 (+0.8)
Second-to-Last	**32.17 (+1.8)**	**42.59 (+1.4)**
Third-to-Last	32.03 (+1.7)	**42.53 (+1.3)**

(a) Methods extracting PrLM output)

Systems	Ko→Ch	Ko→En
Direct	30.24 (-0.1)	40.88 (-0.3)
Summation	**32.17 (+1.8)**	**42.59 (+1.4)**
Average	31.48 (+1.1)	41.94 (+0.8)
Gate Network	31.99 (+1.6)	42.31 (+1.1)
Dropnet	31.64 (+1.3)	41.99 (+0.8)

(b) Merging strategies for PrLM.

Systems	Ko→Ch	Ko→En
SelfAttn	31.78 (+1.4)	42.03 (+0.8)
1^{st} Add&Norm	31.68 (+1.3)	42.01 (+0.8)
FFN	**32.17 (+1.8)**	**42.59 (+1.4)**
2^{nd} Add&Norm	32.01 (+1.7)	42.44 (+1.3)

(c) Merging positions for PrLM.

Systems	Ko→Ch	Ko→En
Both-sides	31.94 (+1.6)	42.23 (+1.0)
Source-side	**32.17 (+1.8)**	**42.59 (+1.4)**
Target-side	31.29 (+0.9)	41.98 (+0.8)

(d) Adding positions for PrLM.

Table 3: The explorations for leveraging PrLM.

4.2 How to merge the PrLM representation with the source input flow?

We compared the impact of different merging strategies for the contextualized representation of PrLM. There are directly using the PrLM as the input embedding (Direct in Table 3b) and four merging strategies such as Summation, Average, using Gate Network (Yang et al., 2020), and using Dropnet (Zhu et al., 2020) As shown in Table 3b, the summation of the PrLM flow and the source input flow got the better improvement over others, so we adopted the Summation strategy in our experiments.

4.3 Where do the PrLM merge with the source input flow?

In the Table 3c, we analyzed where in the encoder layer of Transformer it would be better to merge the PrLM and source flows. There are four positions in a Transformer-encoder layer: after Attn, after 1^{st} Add&Norm, after FFN, and after 2^{nd} Add&Norm. It is interesting to find that merging after FFN can get the best performance. Another observation is that merging the contextualized representation of the PrLM before the Add&Norm (i.e., after Attn or FFN) works better.

4.4 Where do the PrLM flow add?

We evaluated where to add the PrLM: source-side, target-side, and both sides. As shown in the Table 3d report the results. Among them, adding the PrLM to only source-side (i.e., encoder) gained the best result. Intuitively, since contextual representation from the fixed PrLM contains univeral information, not information for generation tasks, combining it directly with the target context may adversely affect performance improvement.

Systems	Ko→Ch
Transformer	30.35 (-)
Multi-PrLMs	
+ELEC. base	32.17 (+1.8)
+ELEC. small, ELEC. base	32.11 (+1.7)
+KoBERT, ELEC. base	32.11 (+1.7)
+(Ko, Han)BERT, ELEC. base	32.19 (+1.8)

Table 4: Leveraging multi-LMs

5 More analyses

5.1 Leveraging Multi-PrLMs

We assumed that because PrLMs were trained with different datasets (size, domain) and diverse configurations, they would contain specific prior knowledge. So, we tried to integrate two or more PrLMs simultaneously (Multi-PrLMs) by adding more extra modules in each encoder layer. Contrary to our expectations, as shown in Table 4, using Multi-PrLMs cannot get a significant performance boost over using single-PrLM.

5.2 Fine-tuning v.s. Freezing

We compared the impact of fine-tuning and freezing the parameters of PrLM when using our method. Table 5 shows the results. We can see that freez-

Systems	Ko→Ch	Ko→En
Transformer	30.35 (-)	41.19 (-)
Freezing PrLM (Ours)		
+ELEC. small	31.51 (+1.1)	42.20 (+1.0)
+ELEC. base	32.17 (+1.8)	42.59 (+1.4)
Fine-tuning PrLM		
+ELEC. small	31.52 (+1.1)	41.98 (+0.8)
+ELEC. base	29.89 (-0.4)	38.92 (-2.3)

Table 5: Fine-tuning v.s. Freezing

Systems	sentences/s	tokens/s
Transformer	164.20 (-)	4.56k (-)
Our Systems		
+ELEC. small	155.59 (-5%)	4.35k (-5%)
+ELEC. base	139.81 (-17%)	3.92k (-16%)

Table 6: Inference Speed on Ko→En NMT task

ing the parameters of PrLM gains more significant improvement than fine-tuning. Another interesting observation is that using the ELECTRA base (112M parameters) when it is fine-tuning led to a significant performance drop, especially for relatively large corpus (Ko→En, 1.6M). It means that *catastrophic forgetting* issue is more pronounced in a resource-rich scenario and using large PrLM. Additionally, tuning separate learning rates (Yang et al., 2020) for the PrLM and the Transformer model may lead to better performance but we leave this to future work.

5.3 Inference Speed

We experimented with the inference speed of our method. Since our method has to obtain the intermediate output of PrLM for an input sentence, it takes more time in the inference process than the baseline model. The experimental results are shown in Table 6. Integrating PrLM into the Transformer model reduced the inference speed by about 5% (ELECTRA small, 14M parameters) to 17% (ELECTRA base, 112M parameters). However, considering the significant performance improvement and the ease of application to any language, it is acceptable of such extra cost.

6 Related Work

Previous studies relies on the structural compatibility of Transformer and PrLM. For example, Clinchant et al. (2019) presented initializing the encoder of Transformer from BERT (fine-tuned or fixed) and observed that freezing the PrLM causes a considerable performance drop. Conneau and Lample (2019) verified that initialization methods with CLM or MLM trained on multi-lingual corpora and showed such initialization are useful on MT. Rothe et al. (2020) used the publicly available PrLM checkpoints to initialize Transformer. While the initialization method is useful to some extent, there is a prerequisite for matching vocabulary and model size/hyper-parameters to them of PrLM.

Zhu et al. (2020) proposed a new method that extracts the last hidden state of BERT for an input sentence and fuses it into the encoder and decoder of the Transformer through an extra attention module, and evaluated the effectiveness of their method on supervised, semi-supervised and unsupervised NMT. Yang et al. (2020) introduced a concerted training framework with three techniques for fusing PrLM and NMT model. Although they also extract the hidden state of PrLM and integrate it into NMT model, the NMT model must follow the PrLM model's configurations such as word segmentation rule and vocabulary.

Our work is related to both Zhu et al. (2020) and Yang et al. (2020) in the sense that we all aim to extract the intermediate output of the PrLM and integrate it into a neural model for better generation quality. As an extension of Zhu et al. (2020), we propose an upgraded method adopted through extensive empirical experiments. Our work differs from Yang et al. (2020) in that we use the publicly available checkpoints that have various configurations and fix the PrLM at training time.

7 Conclusion

While most of the previous works on PrLM address the integration of PrLMs with *fine-tuning*, we propose an alternative in which a modified Transformer-encoder takes the intermediate output from PrLM to exploit its prior knowledge effectively in a straightforward way. Our method does not have to consider the PrLM's configuration, such as its model size, model dimension, and vocabulary. Correspondingly, our approach and reported empirical settings can be smoothly applied to any languages using any checkpoints of PrLMs.

References

Lei Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer normalization. *CoRR*, abs/1607.06450.

Yen-Chun Chen, Zhe Gan, Yu Cheng, Jingzhou Liu, and Jingjing Liu. 2020. Distilling knowledge learned in BERT for text generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7893–7905, Online. Association for Computational Linguistics.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. ELECTRA: pre-training text encoders as discriminators rather than generators. In *Proceedings of the 8th International Conference on Learning Representations, ICLR*, Addis Ababa, Ethiopia.

Stephane Clinchant, Kweon Woo Jung, and Vassilina Nikoulina. 2019. On the use of BERT for neural machine translation. In *Proceedings of the 3rd Workshop on Neural Generation and Translation*, pages 108–117, Hong Kong. Association for Computational Linguistics.

Alexis Conneau and Guillaume Lample. 2019. Cross-lingual language model pretraining. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019*, pages 7057–7067, Vancouver, BC, Canada.

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.

Sergey Edunov, Alexei Baevski, and Michael Auli. 2019. Pre-trained language model representations for language generation. 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 4052–4059, Minneapolis, Minnesota. Association for Computational Linguistics.

Ian J. Goodfellow, Mehdi Mirza, Da Xiao, Aaron Courville, and Yoshua Bengio. 2015. An empirical investigation of catastrophic forgetting in gradient-based neural networks.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, pages 770–778, Las Vegas, NV, USA. IEEE Computer Society.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 177–180, Prague, Czech Republic. Association for Computational Linguistics.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

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. *CoRR*, abs/1907.11692.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 48–53, Minneapolis, Minnesota. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Romain Paulus, Caiming Xiong, and Richard Socher. 2018. A deep reinforced model for abstractive summarization. In *Proceedings of the 6th International Conference on Learning Representations, ICLR, Conference Track Proceedings*, Vancouver, BC, Canada.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2018. Language models are unsupervised multitask learners.

Sascha Rothe, Shashi Narayan, and Aliaksei Severyn. 2020. Leveraging pre-trained checkpoints for sequence generation tasks. *Transactions of the Association for Computational Linguistics*, 8:264–280.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of BERT: smaller, faster, cheaper and lighter. In *Proceedings of the 5th Workshop on Energy Efficient Machine Learning and Cognitive Computing, NeurIPS 2019*, volume abs/1910.01108.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. 2019. How to fine-tune BERT for text classification? *CoRR*, abs/1905.05583.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.

Jiacheng Yang, Mingxuan Wang, Hao Zhou, Chengqi Zhao, Yong Yu, Weinan Zhang, and Lei Li. 2020. Towards making the most of BERT in neural machine translation. In *Proceedings of the AAAI Conference on Artificial Intelligence, 34(05), 9378-9385.*, pages 9378–9385.

Jinhua Zhu, Yingce Xia, Lijun Wu, Di He, Tao Qin, Wengang Zhou, Houqiang Li, and Tie-Yan Liu. 2020. Incorporating BERT into neural machine translation. In *Proceedings of the 8th International Conference on Learning Representations, ICLR*, Addis Ababa, Ethiopia.

A Experimental Settings

A.1 Model Setting

In our experiment, we use Transformer (Vaswani et al., 2017) as the baseline model for NMT and abstractive text summarization tasks. Additionally, for NMT tasks, we compare our approach to the following baselines:

- (Zhu et al., 2020): A method that inserts the last-hidden state of fixed PrLM through PrLM-dedicated attention module in Transformer-encoder and decoder.
- (Clinchant et al., 2019) (Direct* in Table 1): A method that replaces the input embedding with the PrLM that is fine-tuned in training time.

They all use ELECTRA base as the PrLM.

For the Transformer model, we use a `base` Transformer configuration (Vaswani et al., 2017) with an embedding size of 512, 6 encoder and decoder layers, 8 attention heads, shared source and target embedding, the standard `relu` activation function, and sinusoidal positional embedding. We train with a batch size of 3500 tokens and optimize the model parameters using Adam optimizer with a learning rate 7e-4 β_1= 0.9 and $\beta_2 = 0.98$, learning rate warm-up over the first 4000 steps. Additionally, we apply label smoothing with a factor of 0.1. We average over the last 5 checkpoints and run inference with a beam size of 5. All models are trained for 50 epochs using the Torch-based toolkit, Fairseq(-py) (Ott et al., 2019). For the text summarization task, we reduce the number of encoder and decoder layers to 4 and use *Trigram Blocking* (Paulus et al., 2018) to reduce redundancy during inference time. Other settings are the same as above.

For all datasets, we first tokenize sentences using language-specific tokenizer such as KoNLPy[5] for Korean, jieba[6] for Chinese, and Moses (Koehn et al., 2007) for English and then apply Byte-Pair Encoding (Sennrich et al., 2016) to the tokenized sentences with 32K merge-operations. Besides, most of PrLMs have a limit for input sequence length (e.g., 512), so we cut out the middle of some long text for text summarization dataset as proposed in Sun et al. (2019).

[5]https://konlpy.org/en/latest/

[6]https://github.com/fxsjy/jieba

A.2 Pre-trained Language Model Setting

In our experiments, we use 6 types of different PrLMs including BERT (Devlin et al., 2019), DistilBERT (Sanh et al., 2019), ELECTRA (Clark et al., 2020), GPT2 (Radford et al., 2018), multi-lingual BERT, and XLM-RoBERTa (Conneau and Lample, 2019; Liu et al., 2019). Specifically, we use 10 different pre-trained checkpoints depending on the model size, training data set, and training level:

1. KoBERT: a BERT with 768-hidden, 12-layer, 12-heads, 8002-vocab, Korean dataset (4GB), 92M parameters; `https://github.com/SKTBrain/KoBERT.git`.
2. HanBERT: a BERT with 768-hidden, 12-layer, 12-heads, 54000-vocab, Korean dataset (70GB), 127M parameters; `https://github.com/tbai2019/HanBert-54k-N.git`.
3. DistilKoBERT: a DistilBERT with 768-hidden, 3-layer, 12-heads, 8002-vocab, Korean dataset (10GB), 28M parameters; `https://huggingface.co/monologg/distilkobert`.
4. ELECTRA small: a ELECTRA with 256-hidden, 12-layer, 4-heads, 35000-vocab, Korean dataset (34GB), 14M parameters; `https://huggingface.co/monologg/koelectra-small-v3-discriminator`.
5. ELECTRA base: a ELECTRA with 768-hidden, 12-layer, 12-heads, 35000-vocab, Korean dataset (34GB), 112M parameters; `https://huggingface.co/monologg/koelectra-base-v3-discriminator`.
6. KoGPT2: a GPT2 with 768-hidden, 12-layer, 12-heads, 50000-vocab, Korean dataset (20GB), 125M parameters; `https://github.com/SKT-AI/KoGPT2.git`.
7. Multi-lingual BERT cased: a BERT with 768-hidden, 12-layer, 12-heads, 119547-vocab, 104 languages, 177M parameters; `https://huggingface.co/bert-base-multilingual-cased`.
8. Multi-lingual BERT uncased: a BERT with 768-hidden, 12-layer, 12-heads, 105879-vocab, 102 languages, 167M parameters; `https://huggingface.co/bert-base-multilingual-uncased`.
9. XLM RoBERTa base: a BERT with 768-hidden, 12-layer, 12-heads, 250002-vocab, 100 languages (2.5TB), 277M parameters; `https://huggingface.co/xlm-roberta-base`.

10. XLM RoBERTa large: a BERT with 1024-hidden, 24-layer, 16-heads, 250002-vocab, 100 languages (2.5TB), 561M parameters; https://huggingface.co/xlm-roberta-large.

B Details of the Notations

Let Attn denote a multi-head attention module, which takes three matrices containing a query matrix Q, a key matrix K, and a value matrix V and product an output matrix as follows:

$$\texttt{Attn}(Q, K, V) = concat(head_1, ..., head_i)W^o, \tag{6}$$

$$head_i = attn(Q_i, K_i, V_i), \tag{7}$$

$$attn(q, k, v) = softmax(\frac{qW^q kW^k}{\sqrt{d_{model}}})vW^v, \tag{8}$$

where $concat$ denotes a concatenation operation, $softmax$ denotes a softmax function, d_{model} is the dimension of the model, and W^o, W^q, W^k, W^v are parameter matrices. FFN consists of two fully-connected layers with a $relu$ activation in between.

$$\texttt{FFN} = max(0, xW^1 + b^1)W^2 + b^2, \tag{9}$$

where $\max(0, x)$ is $relu$ activation function, and W^1, b^1, W^2, b^2 are parameter matrices. Finally, Attn and FFN are connected with Add&Norm, which denotes a combination module containing a residual connection (He et al., 2016) and a layer normalization (Ba et al., 2016).