s2s-ft: Fine-Tuning Pretrained Transformer Encoders for Sequence-to-Sequence Learning

Hangbo Bao, Li Dong, Wenhui Wang, Nan Yang, Furu Wei
Microsoft Research
https://github.com/microsoft/unilm/tree/master/s2s-ft

Abstract

Pretrained bidirectional Transformers, such as BERT (Devlin et al., 2019), have achieved significant improvements in a wide variety of language understanding tasks, while it is not straightforward to directly apply them for natural language generation. In this paper, we present a sequence-to-sequence fine-tuning toolkit s2s-ft, which adopts pretrained Transformers for conditional generation tasks. Inspired by UniLM (Dong et al., 2019; Bao et al., 2020), we implement three sequence-to-sequence fine-tuning algorithms, namely, causal fine-tuning, masked fine-tuning, and pseudo-masked fine-tuning. By leveraging the existing pretrained bidirectional Transformers, experimental results show that s2s-ft achieves strong performance on several benchmarks of abstractive summarization, and question generation. Moreover, we demonstrate that the package s2s-ft supports both monolingual and multilingual NLG tasks. The s2s-ft toolkit is available at https://github.com/microsoft/unilm/tree/master/s2s-ft.

1 Introduction

Pretrained bidirectional Transformers (Devlin et al., 2019; Yang et al., 2019; Dong et al., 2019; Liu et al., 2019; Conneau et al., 2020; Clark et al., 2020; Bao et al., 2020) have achieved remarkable success on various NLP tasks, such as text classification, and question answering. The BERT-like models are usually pretrained by the masked language modeling task (Taylor, 1953; Devlin et al., 2019), which learns to predict masked tokens based on given context. However, due to the bidirectionality nature, it is not straightforward to directly apply the pretrained bidirectional Transformers to language generation tasks (Wang and Cho, 2019).

There have been several attempts to achieve the above goal. Liu and Lapata (2019) use pretrained BERT (Devlin et al., 2019) as an encoder, and randomly initialize a Transformer-based decoder with larger learning rate. Rothe et al. (2020) initialize the encoder and decoder with different combinations of BERT, GPT (Radford et al., 2018), and RoBERTa (Liu et al., 2019) models. Despite achieving promising results, the performance is still far behind the jointly pretrained encoder-decoder models, such as BART (Lewis et al., 2020) and T5 (Raffel et al., 2019) on generation tasks. We argue that the capability of pretrained bidirectional Transformers has not been fully unleashed on sequence-to-sequence tasks.

In this paper, we present a toolkit (named as s2s-ft) used to fine-tune pretrained bidirectional Transformers on conditional language generation tasks, such as abstractive summarization, and question generation. We follow unified modeling as in (Dong et al., 2019), which shares the same Transformer parameters for both encoding and decoding. Sequence-to-sequence modeling is achieved by employing well-designed self-attention masks in bidirectional Transformers. In other words, the source tokens can attend to each other, while the target tokens can only attend to the left-side context.

We implement three fine-tuning algorithms in s2s-ft. Firstly, causal fine-tuning introduces a position shift for decoding target sequences as in causal language modeling, so that all the decoding tokens can be trained with one forward pass. Secondly, masked fine-tuning randomly masks some target tokens and learns to recover them. The method minimizes the mismatch between pre-training and fine-tuning. Thirdly, pseudo-masked fine-tuning appends pseudo masks into the original target sequence, which combines the benefits of the above two methods.

We build the s2s-ft toolkit upon HuggingFace’s Transformers library (Wolf et al., 2019). We conduct extensive experiments on several language...
Figure 1 shows an overview of three sequence-to-sequence fine-tuning algorithms implemented in the s2s-ft toolkit. We employ special tokens to indicate the boundary of sequences. For example, [CLS] is the first source token, and [SEP] indicates the end of sequences. Sequence-to-sequence learning is achieved by using well-designed self-attention masks (Dong et al., 2019). As shown in Figure 2, all the source tokens can attend to each other, while a target token can only attend to the previously generated tokens and source sequence. We encode the source sequences as conventional bidirectional Transformers. The main difference between three fine-tuning methods lies in how to decode target sequences.

2 Sequence-to-Sequence Fine-Tuning

Sequence-to-sequence learning aims at generate a target sequence $t = t_1, \cdots, t_{|t|}$ by conditioning on the given source sequence $s = s_1, \cdots, s_{|s|}$. In s2s-ft, all tokens are encoded into hidden vectors by Transformer (Vaswani et al., 2017). The target tokens are autoregressively generated via:

$$p(t|s) = \prod_{i=1}^{|t|} p(t_i|t_{<i}, s)$$

(1)

where $t_{<i} = t_1, \cdots, t_{i-1}$.

First, the model is initialized by a pretrained Transformer. Then we use a sequence-to-sequence learning objective to fine-tune the network. Inspired by UniLM (Dong et al., 2019), we share the same model architecture and parameters for both encoding and decoding, which reduces the modeling discrepancies between pre-training and fine-tuning.

generation benchmarks, such as XSum and CNN / DailyMail for abstractive summarization, and SQuAD question generation. We also compare off-the-shelf pretrained bidirectional Transformers (i.e., BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019), ELECTRA (Clark et al., 2020), and UniLM (Dong et al., 2019; Bao et al., 2020)) for sequence-to-sequence learning. In addition, we show that s2s-ft can be easily applied to multilingual language generation tasks by using XLM-RoBERTa (Conneau et al., 2020) as the multilingual pretrained model. Experimental results demonstrate that s2s-ft achieves strong performance across different tasks, and languages.

2.1 Causal Fine-Tuning

The first method learns to decode target in a similar way as causal language models, such as GPT (Radford et al., 2018). In the decoding part, the model generates the current token by feeding the previous prediction at each time step. As shown in Figure 1(a), we feed the start-of-sequence token [SOS] into the model and predict $t_1$ by conditioning on the hidden state. Similarly, $t_1$ is the input at the next time step, which is used to produce $t_2$. The target tokens are completely generated until the end-of-sequence token [SEP] is emitted.

The fine-tuning objective is to maximize the likelihood of generating target tokens conditioning on the source sequence. Unlike masked language model pre-training, where only a portion of tokens are masked and predicted, causal fine-tuning can gather supervision signals from every target predictions within one forward pass. However, the
method involves a *position shift* between model input and prediction in the decoding part, which results in a discrepancy compared with bidirectional Transformer pre-training.

### 2.2 Masked Fine-Tuning

Following Dong et al. (2019), we randomly mask a certain percentage of target tokens, and learn to recover them. The masked fine-tuning algorithm is identical to masked language model pre-training, despite we use a sequence-to-sequence self-attention mask as shown in Figure 2(b). The masked position is supposed to predict the current target token, while other tokens are given as context. Notice that the end-of-sequence token [SEP] can also be masked during fine-tuning in order to learn when to terminate the decoding process.

The fine-tuning objective is to maximize the likelihood of masked tokens given source and uncorrupted target tokens. The method overcomes the position-shift discrepancy between pre-training and fine-tuning described in Section 2.1.

### 2.3 Pseudo-Masked Fine-Tuning

Following Bao et al. (2020), we append pseudo-masked tokens [P] for all the target tokens. The pseudo mask is assigned with the same position embedding as the corresponding original token. Compared with masked fine-tuning, the original tokens are still kept in the input rather than being masked.

The self-attention mask used for pseudo-masked fine-tuning is illustrated in Figure 2(c). All the source tokens can be accessed by others. The pseudo masks and target tokens can only attend to the previous given tokens and themselves. Moreover, the original target token instead of its corresponding pseudo mask is attended by the future time steps.

As shown in Figure 1(c), the target tokens are predicted at the positions of pseudo masks. The fine-tuning objective is to maximize the likelihood of target tokens given source sequence. Pseudo-masked fine-tuning gets the best of the above two methods. The algorithm avoids the position-shift discrepancy compared with causal fine-tuning (Section 2.1). Moreover, all the target tokens can back-propagate error signals, rather than only a portion of target sequence are masked and predicted as in masked fine-tuning (Section 2.2).

### 2.4 Decoding

Given input source $s$, the target sequence is autoregressively generated via $\hat{t} = \arg \max_{t'} p(t'|s)$, where $p(t'|s)$ is factorized as in Equation (1). We approximately find the best decoding results by greedy search or beam search, similar in conventional encoder-decoder methods.

It is worth noting that the hidden states of previous time steps can be cached without re-computing them during the decoding process. So the decoding process has the same computation complexity compared with conventional Transformer sequence-to-sequence models. Moreover, the implementation becomes more unified because s2s-ft uses the same architecture for both encoding and decoding. In contrast, conventional Transformers need to distinguish encoder and decoder (Vaswani et al., 2017), where different architecture are implemented.

For causal fine-tuning, a start-of-sequence to-
ken is fed into the model in order to predict the first target token. Then we in turn append the prediction to input and generate the next token. We repeat the process until the end-of-sequence token is emitted. In contrast, the other two methods use a mask $[M]/[P]$ as input to predict the current target token. The mask will be substituted with its prediction in the next time step.

3 Experiments

$s2s-ft$ is built upon HuggingFace’s Transformers library (Wolf et al., 2019), so that we can load various off-the-shelf pretrained models. We implement the sequence-to-sequence fine-tuning algorithms described in Section 2. We conduct experiments on a set of language generation benchmarks, including abstractive summarization, and question generation. The hyperparameters are chosen on the development set of each dataset.

3.1 Benchmarks

We summarize all the evaluation benchmarks in Table 1. The datasets cover two tasks, and two more languages. We report ROUGE (Lin, 2004) scores as the evaluation metrics for abstractive summarization. In addition, we include BLEU (Papineni et al., 2002) and METEOR (Banerjee and Lavie, 2005) metrics for question generation.

3.1.1 Monolingual Dataset

**CNN / DailyMail** (See et al., 2017) The abstractive summarization dataset aims at generating a concise and fluent summary from an English news article crawled from CNN and DailyMail.

**XSum** (Narayan et al., 2018) The extreme summarization dataset compresses a BBC news article to a one-sentence summary.

**SQuAD** (Du and Cardie, 2018) The question generation dataset aims at generating relevant questions given a paragraph and an answer span, which is based on SQuAD v1.1 (Rajpurkar et al., 2016).

3.1.2 Multilingual Dataset

**Gigaword** (Chi et al., 2020) The headline generation datasets are built upon French (fr) and Chinese (zh) article-headline pairs.

**WebQA** (Chi et al., 2020) The Chinese question generation dataset is built upon WebQA (Li et al., 2016).

3.2 Comparison of Fine-Tuning Methods

We first compare the three sequence-to-sequence fine-tuning algorithms using the BERT-base-uncased checkpoint as the pretrained model. We report evaluation results on the developments sets of XSum and SQuAD in Table 2. The results show that pseudo-masked fine-tuning achieves the best performance on two datasets, except that masked fine-tuning obtains the highest ROUGE-1 score on XSum. Moreover, causal fine-tuning is consistently worse than the other two algorithms. The results indicate that reducing the discrepancy between masked language model pre-training and sequence-to-sequence fine-tuning is beneficial. We therefore use pseudo-masked fine-tuning in the rest of the experiments.

3.3 Comparison of Pretrained Models

We compare different pretrained models for initialization, including BERT (Devlin et al., 2019),...
### Table 4: Abstractive summarization results on the test set of CNN / DailyMail, and XSum. The evaluation metric is the F1 version of ROUGE (RG) scores. We also present the number of parameters (#Param) for the methods using pretrained models.

| Model                                      | #Param | Corpus   | CNN / DailyMail          | XSum        |
|--------------------------------------------|--------|----------|--------------------------|-------------|
| **Without pre-training**                   |        |          |                          |             |
| PTN-NN (See et al., 2017)                  | -      | -        | 39.53/17.28/36.38        | 28.10/8.02/21.72 |
| **Fine-tuning base-size pretrained models**|        |          |                          |             |
| MASS (Song et al., 2019)                   | 123M   | -        | 42.12/19.50/39.01        | 39.75/17.24/31.95 |
| BERTSUMans (Liu, 2019)                     | 156M   | 16GB     | 41.72/19.39/38.76        | 38.76/16.33/31.15 |
| ERNIE-GEN (Xiao et al., 2020)              | 110M   | 16GB     | 42.30/19.92/39.68        | -           |
| T5 (Raffel et al., 2019)                   | 220M   | 750GB    | 42.05/20.34/39.40        | -           |
| s2s-ft RoBERTa-base                        | 125M   | 160GB    | 42.28/20.21/39.87        | 43.39/20.55/35.63 |
| s2s-ft UniLMv2-base                       | 110M   | 160GB    | **43.89**/21.05/41.02    | **44.37**/21.54/36.61 |
| **Fine-tuning large-size pretrained models**|        |          |                          |             |
| UniLM (Dong et al., 2019)                  | 340M   | 16GB     | 43.08/20.43/40.34        | -           |
| ERNIE-GEN (Xiao et al., 2020)              | 340M   | 16GB     | 44.02/21.17/41.26        | -           |
| BART (Lewis et al., 2020)                  | 400M   | 160GB    | 44.16/21.28/40.90        | 45.14/22.27/37.25 |
| ProphetNet (Yan et al., 2020)              | 400M   | 160GB    | 44.20/21.17/41.30        | -           |
| PEGASUSc4 (Zhang et al., 2020)             | 568M   | 750GB    | 43.90/21.20/40.76        | 45.20/22.06/36.99 |
| PEGASUSHurtNews (Zhang et al., 2020)       | 568M   | 380GB    | 44.17/21.47/41.11        | **47.21**/24.56/39.25 |
| T511B (Raffel et al., 2019)                | 11B    | 750GB    | 43.52/21.55/40.69        | -           |
| s2s-ft RoBERTa-large                       | 355M   | 160GB    | 43.92/21.25/41.06        | 45.63/22.72/37.86 |
| s2s-ft UniLMv2-large                       | 340M   | 160GB    | **44.79**/21.98/41.93    | **47.58**/24.35/39.50 |

ELECTRA (Clark et al., 2020), RoBERTa (Liu et al., 2019) and UniLMv2 (Bao et al., 2020). The base-size checkpoints are used in the comparison. As shown in Table 3, we report the results of pseudo-masked fine-tuning (Section 2.3) on XSum and SQuAD.

Among the four pretrained models, UniLMv2 performs best in terms of the automatic evaluation metrics, which contains a partially autoregressive pre-training objective that is similar to sequence-to-sequence modeling. The models initialized by BERT and RoBERTa obtain better results compared with ELECTRA. The results indicate that masked language model pre-training over the full vocabulary are helpful for sequence-to-sequence tasks. Although ELECTRA obtains comparable performance on a wide range of language understanding tasks (e.g., text classification, and question answering), the language modeling ability is not fully pretrained (Clark et al., 2020).

### 3.4 Comparisons with Previous Work

We conduct evaluation by using s2s-ft to fine-tune RoBERTa (Liu et al., 2019) and UniLMv2 (Bao et al., 2020) on abstractive summarization (i.e., CNN / DailyMail, and XSum) and question generation (i.e., SQuAD). Pseudo-masked fine-tuning is used both base-size and large-size models.

As shown in Table 4 and Table 5, s2s-ft UniLMv2 achieves state-of-the-art performance on all three benchmarks compared with the models that use more parameters, larger corpus, or task-specific pre-training. Specifically, T511B (Raffel et al., 2019) uses 11 billion parameters and 750GB text corpus to pretrain a sequence-to-sequence model. PEGASUS (Zhang et al., 2020) is a task-specific pretrained model designed for abstractive summarization. The comparisons indicate that s2s-ft can obtain strong performance on sequence-to-sequence tasks by leveraging the pretrained models.

It is notable that RoBERTa obtains very competitive performance compared with previous work. The comparisons show that the masked language modeling pre-training (Devlin et al., 2019) is helpful for language generation tasks. Moreover, s2s-ft provides a unified modeling method to employ the existing pretrained Transformers for sequence-to-sequence tasks.

### 3.5 Results of Multilingual Generation

Apart from monolingual generation tasks, we can use s2s-ft to leverage the multilingual pretrained models, such as mBERT (Devlin et al., 2019), and XLM-RoBERTa (Conneau et al., 2020). We conduct language generation experiments on both abstractive summarization (French Gigaword, and Chinese Gigaword) and Chinese question generation (WebQAzh).
| Model                          | #Param | Corpus | Official Split | Reversed Split |
|-------------------------------|--------|--------|----------------|----------------|
|                               |        |        | BLEU-4/MTR/RG-L | BLEU-4/MTR/RG-L |
| **Without pre-training**      |        |        |                |                |
| (Du and Cardie, 2018)         | -      | -      | 15.16/19.12/-  | -              |
| (Zhao et al., 2018)           | -      | -      | -              | 16.38/20.25/44.48 |
| (Zhang and Bansal, 2019)      | -      | -      | 18.37/22.65/46.68 | 20.76/24.20/48.91 |
| **Fine-tuning base-size pretrained models** |        |        |                |                |
| ERNIE-GEN (Xiao et al., 2020) | 110M   | 16GB   | 22.28/25.13/50.58 | 23.52/25.61/51.45 |
| s2s-ftRoBERTa-BASE           | 125M   | 160GB  | 23.86/25.93/51.68 | 25.32/26.61/52.62 |
| s2s-ftUniLMv2-BASE           | 110M   | 160GB  | 24.70/26.33/52.13 | 26.30/27.09/53.19 |
| **Fine-tuning large-size pretrained models** |        |        |                |                |
| UniLM (Dong et al., 2019)    | 340M   | 16GB   | 22.12/25.06/51.07 | 23.75/25.61/52.04 |
| ERNIE-GEN (Xiao et al., 2020) | 340M   | 16GB   | 24.03/26.31/52.36 | 25.57/26.89/53.31 |
| ProphetNet (Yan et al., 2020) | 400M   | 16GB   | 25.01/26.83/52.57 | 26.72/27.64/53.79 |
| s2s-ftRoBERTa-LARGE          | 400M   | 160GB  | 25.30/26.85/52.66 | 26.82/27.48/53.92 |
| s2s-ftUniLMv2-LARGE          | 340M   | 160GB  | 25.97/27.33/53.43 | 27.12/27.95/54.25 |

Table 5: Question generation results on the test set of SQuAD. MTR is short for METEOR, and RG for ROUGE. The official split is from (Du and Cardie, 2018), while the reversed split is the same as in (Zhao et al., 2018).

| Model                          | #Param | Corpus | Official Split | Reversed Split |
|-------------------------------|--------|--------|----------------|----------------|
|                               |        |        | BLEU-4/MTR/RG-L | BLEU-4/MTR/RG-L |
| WebQA.ch (Chinese)            |        |        |                |                |
| XLM (Chi et al., 2020)        | 23.41/23.32/47.40 | 55.30/42.57/52.95 | 56.27/39.20/52.84 |
| XNLG (Chi et al., 2020)       | 24.89/24.53/49.72 | 57.65/44.93/54.95 | 57.95/41.30/54.54 |
| s2s-ftXLM-RoBERTa-BASE        | 27.45/25.20/49.76 | 60.29/47.24/57.46 | 58.48/41.79/55.04 |
| s2s-ftXLM-RoBERTa-LARGE       | 28.49/26.48/52.94 | 60.95/47.94/58.09 |                |

Table 6: Evaluation results of Chinese and French abstractive summarization, and Chinese question generation. QG is short for question generation, AS for abstractive summarization, BL for BLEU, MTR for METEOR, and RG for ROUGE.

As shown in Table 6, we employ s2s-ft to fine-tune XLM-RoBERTa on the three benchmarks. We compare our results with fine-tuning XNLG (Chi et al., 2020) and XLM (Conneau and Lample, 2019) that are pretrained conventional sequence-to-sequence Transformers. s2s-ft achieves significantly better performance than previous work across different languages and tasks. The results indicate that s2s-ft can unleash the multilinguality of XLM-RoBERTa on generation tasks. More importantly, the support of multilingual pretrained models greatly widens the application range of our s2s-ft toolkit.

4 Conclusion

We introduce a sequence-to-sequence toolkit s2s-ft to fine-tune the pretrained bidirectional Transformers for language generation tasks. The toolkit follows the UniLM (Dong et al., 2019; Bao et al., 2020) fine-tuning algorithms, which unifies encoding and decoding with the same modeling method. We conduct extensive experiments on abstractive summarization and question generation, including both monolingual and multilingual settings. We plug in different pretrained models in our toolkit and evaluate three fine-tuning approaches. Then we compare s2s-ft with previous work using both base-size and large-size models. In addition, we use s2s-ft to apply off-the-shelf multilingual pretrained model on Chinese and French sequence-to-sequence learning. Experimental results show that the proposed toolkit achieves strong performance across the tasks and languages. We believe the toolkit is important to unleash the abilities of BERT-like bidirectional Transformers on sequence-to-sequence tasks.

References

Satanjeev Banerjee and Alon Lavie. 2005. **METEOR: An automatic metric for MT evaluation with improved correlation with human judgments.** In *Proceedings of the ACL Workshop on Intrinsic and Ex-
A Hyperparameters for Fine-Tuning

Table 7 reports the most hyperparameters used in this paper.

| Hyperparameter          | Value                  |
|-------------------------|------------------------|
| Batch size              | 64                     |
| Label smoothing         | 0.1                    |
| Adam $\epsilon$        | $1e^{-6}$              |
| Adam $\beta$           | (0.9, 0.999)          |
| Learning rate schedule | Linear                |
| Warmup steps            | 1000                   |
| Gradient clipping       | 1.0                    |
| Dropout                 | 0.1                    |
| Weight decay            | 0.01                   |

Table 7: Hyperparameters for fine-tuning.

The optimal hyperparameter values are task-specific and we provide a range of possible values that work well for various downstream tasks:

- **Learning rate for base-sized models**: 5e-5, 7e-5, 1e-4
- **Learning rate for large-sized models**: 1e-5, 1.5e-5, 2e-5, 3e-5
- **Number of fine-tuning epochs**: 10, 15, 20, 30
- **Mask prob for target sequence**: 40%, 50%, 60%, 70%

Mask prob for target sequence denotes the probability that each token in target sequence is masked. We conduct grid search on the development sets to find the best hyperparameters and use for the test sets. The other task-specific hyperparameters are listed in Table 8.

| Task               | Max input tokens | Max output tokens | Beam size | Length penalty | Min output tokens |
|--------------------|------------------|-------------------|-----------|---------------|------------------|
| CNN / DailyMail    | 608              | 160               | 5         | 0.9           | 48               |
| XSum               | 720              | 48                | 8         | 0.7           | 1                |
| SQuAD QG           | 384              | 32                | 8         | 1.3           | 5                |
| WebQA$_a$ QG       | 384              | 32                | 8         | 1.3           | 5                |
| Gigawords$_e$      | 96               | 48                | 5         | 0.9           | 1                |

Table 8: Task-specific hyperparameters for evaluation benchmarks.