STEP: Sequence-to-Sequence Transformer Pre-training for Document Summarization

Yanyan Zou[1*], **Xingxing Zhang**[2], **Wei Lu**[1], **Furu Wei**[2], and **Ming Zhou**[2]
[1] StatNLP Research Group, Singapore University of Technology and Design
[2] Microsoft Research Asia, Beijing, China
yanyan_zou@mymail.sutd.edu.sg
{xizhang,fuwei,mingzhou}@microsoft.com
luwei@sutd.edu.sg

Abstract

Abstractive summarization aims to rewrite a long document to its shorter form, which is usually modeled as a sequence-to-sequence (SEQ2SEQ) learning problem. SEQ2SEQ Transformers are powerful models for this problem. Unfortunately, training large SEQ2SEQ Transformers on limited supervised summarization data is challenging. We therefore propose STEP (as shorthand for **S**equence-to-Sequence **T**ransform**E**r **P**re-training), which can be trained on large scale unlabeled documents. Specifically, STEP is pre-trained using three different tasks, namely sentence reordering, next sentence generation, and masked document generation. Experiments on two summarization datasets show that all three tasks can improve performance upon a heavily tuned large SEQ2SEQ Transformer which already includes a strong pre-trained encoder by a large margin. By using our best task to pre-train STEP, we outperform the best published abstractive model on CNN/DailyMail by 0.8 ROUGE-2 and New York Times by 2.4 ROUGE-2.[1]

1 Introduction

Large pre-trained language models (Peters et al., 2018; Radford et al., 2018; Devlin et al., 2019; Yang et al., 2019; Liu et al., 2019) improved the state-of-the-art of various natural language understanding (NLU) tasks such as question answering (e.g., SQuAD; Rajpurkar et al. 2016), natural language inference (e.g., MNLI; Bowman et al. 2015) as well as text classification (Zhang et al., 2015). These models (i.e., large LSTMs; Hochreiter and Schmidhuber 1997 or Transformers; Vaswani et al. 2017) are pre-trained on large scale unlabeled text with language modeling (Peters et al., 2018; Radford et al., 2018), masked language modeling (Devlin et al., 2019; Liu et al., 2019) and permutation language modeling (Yang et al., 2019) objectives. In NLU tasks, pre-trained language models are mostly used as text encoders.

Abstractive document summarization aims to rewrite a long document to its shorter form while still retaining its important information. Different from *extractive* document summarization that extacts important sentences, *abstractive* document summarization may paraphrase original sentences or delete contents from them. For more details on differences between *abstractive* and *extractive* document summary, we refer the interested readers to Nenkova and McKeown (2011) and Section 2. This task is usually framed as a sequence-to-sequence learning problem (Nallapati et al., 2016; See et al., 2017). In this paper, we adopt the sequence-to-sequence (SEQ2SEQ) Transformer (Vaswani et al., 2017), which has been demonstrated to be the state-of-the-art for SEQ2SEQ modeling (Vaswani et al., 2017; Ott et al., 2019). Unfortunately, training large SEQ2SEQ Transformers on limited supervised summarization data is challenging (Ott et al., 2019) (refer to Section 5). The SEQ2SEQ Transformer has an *encoder* and a *decoder* Transformer. Abstractive summarization requires both *encoding* of an input document and *generation* of a summary usually containing multiple sentences. As mentioned earlier, we can take advantage of recent pre-trained Transformer encoders for the document *encoding* part as in Liu and Lapata (2019). However, Liu and Lapata (2019) leave the decoder randomly initialized. In this paper, we aim to pre-train both the encoder (i.e., the *encoding* part) and decoder (i.e., the *generation* part) of a SEQ2SEQ Transformer , which is able to improve abstractive summarization performance.

Based on the above observations, we propose STEP (as shorthand for **S**equence-to-Sequence

[*] Contribution during internship at Microsoft Research.
[1] We plan to make our code and models public available.

Transform**E**r **P**re-training), which can be pre-trained on large scale unlabeled documents. Specifically, we design three tasks for SEQ2SEQ model pre-training, namely Sentence Reordering (SR), Next Sentence Generation (NSG), and Masked Document Generation (MDG). SR learns to recover a document with randomly shuffled sentences. NSG generates the next segment of a document based on its preceding segment. MDG recovers a masked document to its original form. After pre-trianing STEP using the three tasks on unlabeled documents, we fine-tune it on supervised summarization datasets.

We evaluate our methods on two summarization datasets (i.e., the CNN/DailyMail and the New York Times datasets). Experiments show that all three tasks we propose can improve upon a heavily tuned large SEQ2SEQ Transformer which already includes a strong pre-trained encoder by a large margin. Compared to the best published abstractive models, STEP improves the ROUGE-2 by 0.8 on the CNN/DailyMail dataset and by 2.4 on the New York Times dataset using our best performing task for pre-training. Human experiments also show that STEP can produce significantly better summaries in comparison with recent strong abstractive models.

2 Related Work

This section introduces extractive and abstractive document summarization as well as pre-training methods for natural language processing tasks.

Extractive Summarization Extractive summarization systems learn to find the informative sentences in a document as its summary. This task is usually viewed as a sentence ranking problem (Kupiec et al., 1995; Conroy and O'leary, 2001) using scores from a binary (sequence) classification model, which predicts whether a sentence is in the summary or not. Extractive neural models employ hierarchical LSTMs/CNNs as the feature learning part of the binary (sequence) classifier (Cheng and Lapata, 2016; Nallapati et al., 2017; Narayan et al., 2018; Zhang et al., 2018), which largely outperforms discrete feature based models (Radev et al., 2004; Filatova and Hatzivassiloglou, 2004; Nenkova et al., 2006). Very recently, the feature learning part was replaced again with pre-trained transformers (Zhang et al., 2019; Liu and Lapata, 2019) that lead to another huge improvement of summarization performance. However, extractive models have their own limitations. For example, the extracted sentences might be too long and redundant. Besides, human written summaries in their nature are abstractive. Therefore, we focus on abstractive summarization in this paper.

Abstractive Summarization The goal of abstractive summarization is to generate summaries by rewriting a document, which is a sequence-to-sequence learning problem. SEQ2SEQ attentive LSTMs (Hochreiter and Schmidhuber, 1997; Bahdanau et al., 2015) are employed in Nallapati et al. (2016). Even these models are extended with copy mechanism (Gu et al., 2016), coverage model (See et al., 2017) and reinforcement learning (Paulus et al., 2018), their results are still very close to that of Lead3 which selects the leading three sentences of a document as its summary. One possible reason is that LSTMs without pre-training are not powerful enough. Liu and Lapata (2019) used a SEQ2SEQ Transformer model with its encoder initialized with a pre-trained Transformer (i.e., BERT; Devlin et al. 2019) and achieved the state-of-the-art performance. Our work goes one step further, we propose a method to pre-train the decoder together with the encoder and then initialize both the encoder and decoder of a summarization model with the pre-trained Transformers.

There is also a line of work that bridges extractive and abstractive models with reinforcement learning (Chen and Bansal, 2018), attention fusion (Hsu et al., 2018) and bottom-up attention (Gehrmann et al., 2018), while our model is conceptually simpler.

Pre-training Pre-training methods draw a lot of attention recently. Peters et al. (2018) and Radford et al. (2019) pre-trained LSTM and Transformer encoders using language modeling objectives. To leverage the context in both directions, (Devlin et al., 2019) proposed BERT, which is trained with the mask language modeling objective. XLNet (Yang et al., 2019) is trained with permutation language modeling objective, which removes the independence assumption of masked tokens in BERT. RoBERTa (Liu et al., 2019) extends BERT with more training data and better training strategies. All the methods above focus on pre-training an encoder, while we propose methods to pre-train both the encoder and decoder of a SEQ2SEQ model.

Dong et al. (2019) proposed a Transformer language model that can be used for both natural language understanding and generation tasks, which is pre-trained using masked, unidirectional and SEQ2SEQ language modeling objectives. Their method tries to pre-train a SEQ2SEQ Transformer with its encoder and decoder parameters shared. Differently, we pre-train a SEQ2SEQ Transformer with separate parameters for the encoder and decoder. Song et al. (2019) proposed a method to pre-train a SEQ2SEQ Transformer by masking a span of text and then predicting the original text with masked tokens at other positions. Their pre-training task is similar to our Masked Document Generation task, but we apply a different masking strategy and predict the original text without masked tokens. Besides, we propose another two tasks for SEQ2SEQ model pre-training. Song et al. (2019) tested their model on sentence-level tasks (e.g., machine translation and sentence compression), while we aim to solve document-level tasks (e.g., abstractive document summarization).

3 Sequence-to-Sequence Transformer Pre-training

This section first introduces the backbone architecture of our abstractive summarization model STEP. We then describe methods to pre-train STEP and finally move on to the fine-tuning on summarization datasets.

3.1 Architecture

In this work, the task of abstractive document summarization is modeled as a sequence-to-sequence learning problem, where a document is viewed as a sequence of tokens and its corresponding summary as another sequence of tokens. We adopt the SEQ2SEQ Transformer architecture (Vaswani et al., 2017), which includes an *encoder* Transformer and a *decoder* Transformer. Both the *encoder* and *decoder* Transformers have multiple layers and each layer contains a multi-head attentive sub-layer[2] followed by a fully connected sub-layer with residual connections (He et al., 2016) and layer normalization (Ba et al., 2016).

Let us use $X = (x_1, x_2, \ldots, x_{|X|})$ to denote a document and use $Y = (y_1, y_2, \ldots, y_{|Y|})$ to denote its summary. The encoder takes the document X as input and transforms it to its contextual representations. The decoder learns to generate the summary Y one token at a time based on the contextual representations and all preceding tokens that have been generated so far:

$$p(Y|X;\theta) = \prod_{t=1}^{|Y|} p(y_t|y_{<t}; X; \theta) \quad (1)$$

where $y_{<t}$ stands for all tokens before position t (i.e., $y_{<t} = (y_1, y_2, \ldots, y_{t-1})$). This model can be trained by minimizing the negative log-likelihood of the training document-summary pairs.

3.2 Pre-training Tasks

Training a SEQ2SEQ Transformer model on a summarization dataset from scratch is difficult due to the limited number of document-summary pairs. Pre-trained Transformer encoders such as BERT (Devlin et al., 2019) and RoBERTa (Liu et al., 2019) have achieved great success in many natural language understanding tasks. Therefore, we first initialize the encoder of our SEQ2SEQ Transformer summarization model STEP with an existing pre-trained Transformer encoder (i.e., RoBERTa) to enhance its language understanding capabilities. To help STEP gain language generation capabilities and the abilities of associating generated text with encoder outputs, we continue to pre-train it on unlabeled text. In the following, we describe our pre-training tasks.

Sentence Reordering A document is typically composed of multiple sentences separated by full stops. In this task, we first shuffle the document by sentences and then recover the original document. There are several reasons why we design this task. First, a summary of a document usually consists of multiple sentences. We expect that STEP learns to generate long and coherent summaries (across sentences). The output of the task (i.e., the original document) also contains multiple sentences. Second, sentence reordering (or content reordering) is necessary for summarization. According to the statistics on training sets of our summarization datasets, contents of the original documents are *reordered* in their summaries for 40% of cases. We define *content reordering* as follows. For each document-summary pair, we first map each sentence in the summary to one sentence in its paired document by maximizing the ROUGE score. If the sequence of sentences

[2]Note that each layer of the *decoder* Transformer has two attentive sub-layers: one for tokens that have already been generated and the other for the output of the *encoder*.

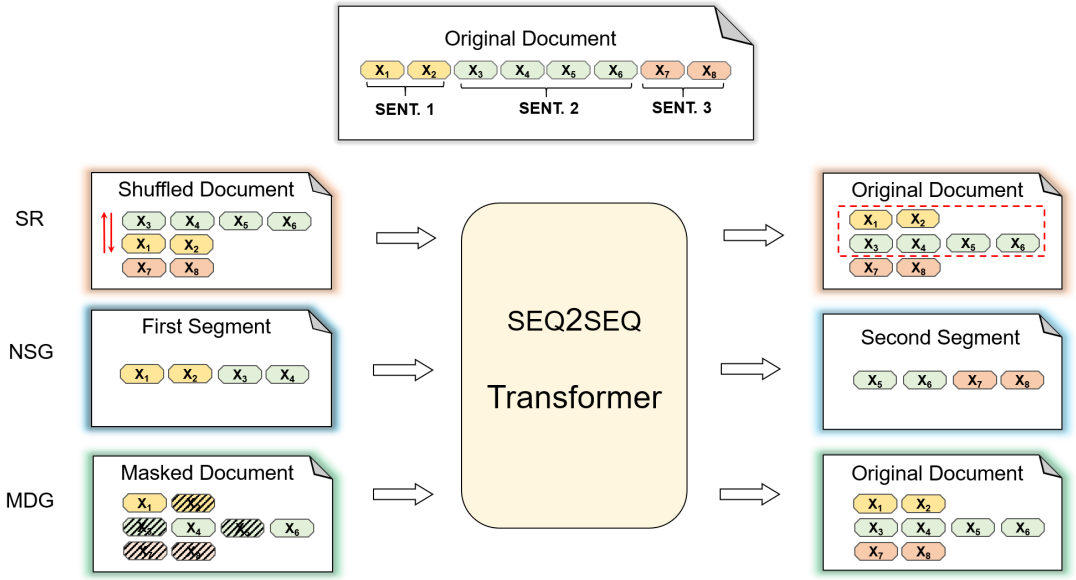

Figure 1: Pre-training of STEP. The document $(x_1, x_2, \cdots, x_8)$ contains three sentences (i.e., SENT. 1, SENT. 2 and SENT. 3). STEP adopts the SEQ2SEQ Transformer architecture. It takes the transformed document (i.e., a shuffled document, the first segment of a document, or a masked document) as input and learns to recover the original document (or part of the original document) by generation. SR: Sentence Reordering; NSG: Next Sentence Generation; MDG: Masked Document Generation.

in the summary is different from the sequence of their mapped sentences in the original document, we count this as one *content reordering*. Thirdly, abstractive summary requires reproducing factual details (e.g., named entities, figures) from source text. We also expect STEP to learn to copy tokens. Here is a formal definition of this task. Let us change the notation of a document slightly in this paragraph. Let $X = (S_1, S_2, \ldots, S_m)$ denote a document, where $S_i = (w_1^i, w_2^i, \ldots, w_{|S_i|}^i)$ is a sentence in it, w_j^i is a word in S_i and m is the number of sentences. X is still a sequence of tokens (by concatenating tokens in all sentences). Let $A = \texttt{permutation}(m) = (a_1, a_2, \ldots, a_m)$ denote a permuted range of $(1, 2, \ldots, m)$ and therefore $\hat{X}_S = (S_{a_1}, S_{a_2}, \ldots, S_{a_m})$ is the *shuffled* document. Note that $\hat{X}_S$ is a sequence of tokens by concatenating all *shuffled* sentences. STEP can be trained on $\langle \hat{X}_S, X \rangle$ pairs constructed from unlabeled documents, as demonstrated in Figure 1.

Note that document rotation[3] is a special case of sentence reordering with significant amount of partially ordered sentences, which we believe is a simpler task. In this work, we thus only consider the general case of sentence reordering.

[3]A document is randomly divided into two fragments $X = (F_1, F_2)$ using full stops. The rotated document is $\hat{X}_R = (F_2, F_1)$. Document rotation recovers X using $\hat{X}_R$.

Next Sentence Generation The second pre-training task leverages the natural order of text. Next Sentence Generation (NSG) uses one span of text in a document to predict its next span of text, as shown in Figure 1. Specifically, we split a document into two segments (i.e., G_1 and G_2). Note that each segment might contain multiple sentences, which fits the document summarization task very well, since either a document or its summary usually includes multiple sentences. Intuitively, in a document, sentences are highly correlated with their preceding sentences due to the context dependent nature of documents or language. We intend our model to learn to generate multiple sentences and also learn to focus on preceding context.

We have at least two options for the splitting position of the two segments. **Option one**: the position right after a full-stop symbol (such as period, question mark, etc.) is selected as the splitting point, which ensures full sentences for each segment. **Option two**: the splitting point can be at any position within the document. We choose the second option, which may lead to incomplete sentences in segments. We intend to force the encoder and decoder to understand input text without complete information, which we believe is more challenging compared to option one. Besides, as a common wisdom in abstractive summarization, documents are truncated to a fixed num-

ber of tokens, which may also contain incomplete sentences. We use option two to reduce the pre-training and fine-tuning input mismatch. In this task, we train the model STEP on large amount of $\langle G_1, G_2 \rangle$ pairs constructed following the option two splitting strategy.

Next sentence prediction has been used in skip-thought vectors (Kiros et al., 2015). There are two differences. First, each segment in their model only has one sentence; second, they use this task to pre-train an encoder rather than an entire SEQ2SEQ model. (Devlin et al., 2019) introduced a task named next sentence prediction (NSP), which is different from this task. NSP is a classification task, but NSG is a generation task, which intends to pre-train a generation model.

Masked Document Generation The third task we consider is Masked Document Generation (MDG) that learns to recover a document with a masked span of tokens (see Figure 1). For simplicity, a document consisting of a sequence of tokens is denoted as $X = (x_1, x_2, \cdots, x_{|X|})$. We randomly sample the length of the span l from a discrete uniform distribution $\mathcal{U}(a, b)$ and the span start position k from another discrete uniform distribution $\mathcal{U}(1, |X| - l + 1)$ (see Section 4 for more details). Thus, $\mathcal{M} = (x_k, x_{k+1}, \cdots, x_{k+l-1})$ is the text span to be masked.

One straightforward masking strategy is to replace each token residing in $\mathcal{M}$ with a special `[MASK]` token. However, we refrain from doing so because of the following three reasons. Usually, `[MASK]` tokens will not appear in downstream tasks. Second, entirely masking a continuous subsequence of X may make the whole document incomprehensible, which might be too challenging for our model to learn. Third, similar to SR, avoiding replacing every token with `[MASK]` also helps our model learn the ability of copying tokens from the input while preserving the ability of generating novel tokens.

In the sub-sequence $\mathcal{M}$, each token is processed with one of the three strategies: 1) replaced with the `[MASK]` token; 2) replaced with a random token; 3) remain unchanged. Inspired by BERT (Devlin et al., 2019), for 80% tokens, we follow strategy 1). In 10% of cases, we employ strategy 2) and we use strategy 3) for the remaining 10% of cases. Let $\hat{X}_M$ denote the document after the application of our masking strategy. We could create infinite amount of $\langle \hat{X}_M, X \rangle$ pairs to train STEP.

During pre-training, we could also employ all the three tasks (i.e., SR, NSG, MDG) together. For each training batch, we randomly choose one task and each task is used for $1/3$ of the time.

3.3 Fine-tuning

After pre-training STEP with the three tasks introduced in Section 3.2, we fine-tune the model on abstractive document summarization datasets. The fine-tuning process is straightforward. We simply continue to train STEP on the supervised document-summary pairs. Similar to other SEQ2SEQ summarization models, we do beam search during the generation of summaries.

4 Experimental Setup

In this section, we present the experimental setup for evaluating our summarization models. We first introduce the datasets used for our experiments. Then we describe training details of our models as well as our evaluation protocols.

4.1 Datasets

We assess the summarization performance of our models on two benchmark datasets: the CNN/DailyMail (CNNDM) dataset (Hermann et al., 2015; See et al., 2017) and the New York Times (NYT) dataset (Sandhaus, 2008). We pre-train our models on the GIGA-CM dataset introduced in Zhang et al. (2019).

CNNDM CNNDM contains news articles and their associated highlights (i.e., summaries) collected from the CNN and Daily Mail Online websites[4]. Following previous work (See et al., 2017; Zhang et al., 2019; Liu and Lapata, 2019), we use the non-anonymized version of CNNDM. Specifically, we preprocess the dataset with the publicly available scripts[5] provided by See et al. (2017) and obtain 287,226 document-summary pairs for training, 13,368 for validation and 11,490 for test.

NYT The NYT dataset is a collection of articles along with multi-sentence summaries written by library scientists. We closely follow the preprocessing procedures described in Durrett et al. (2016) and Liu and Lapata (2019). The test set is constructed by including all articles published on January 1, 2017 or later, which contains 9,076

[4] `https://edition.cnn.com` and `https://dailymail.co.uk`

[5] https://github.com/abisee/cnn-dailymail

articles. The remaining 100,834 articles are split into a training set of 96,834 examples and a validation set of 4,000 examples. As in (Durrett et al., 2016), we also remove articles whose summaries contain less than 50 words from the test set, and the resulting test set contains 3,452 examples.

GIGA-CM To pre-train our model with the tasks introduced in Section 3.2, following the procedures in (Zhang et al., 2019), we created the GIGA-CM dataset, which contains only unlabeled documents. The training set of GIGA-CM is composed of 6,521,658 documents sampled from the English Gigaword dataset[6] and the training documents in CNNDM. We used the 13,368 documents in the validation split of CNNDM as its validation set. Note that the Gigaword dataset overlaps with the NYT dataset and we therefore exclude the test set of NYT from the training set of GIGA-CM.

For CNNDM, NYT and GIGA-CM datasets, we segment and tokenize documents and/or summaries (GIGA-CM only contains documents) using the Stanford CoreNLP toolkit (Manning et al., 2014). To reduce the vocabulary size, we further apply the UTF8 based BPE (Sennrich et al.) introduced in GPT-2 (Radford et al., 2019) to all datasets. As a common wisdom in abstractive summarization, documents and summaries in CNNDM and NYT are usually truncated to 512 and 256 tokens, respectively.

We leverage unlabeled documents differently for different pre-training tasks (see Section 3.2). We first split each document into 512 token segments if it contains more than 512 tokens (segments or documents with less than 512 tokens are removed). In Sentence Reordering (SR) and Masked Document Generation (MDG), we use the segment after transformation to predict the original segment. We set the minimum masked length $a = 100$ and the maximum masked length $b = 256$ in MDG. In Next Sentence Generation (NSG), each segment is used to predict its next 256 tokens.

4.2 Implementation Details

As mentioned in Section 3, our model is a SEQ2SEQ Transformer model (Vaswani et al., 2017). The encoder is initialized with the $\text{RoBERTa}_{\text{LARGE}}$ model[7] (Liu et al., 2019), and therefore they share the same architecture. Specifically, the encoder is a 24-layer Transformer. Each layer has 16 attention heads and its hidden size and feed-forward filter size are 1,024 and 4,096, respectively. The decoder is shallower with 6 layers. The hidden size and number of attention head of the decoder are identical to these of the encoder, but the feed-forward filter size is 2,048. We use a smaller filter size in the decoder to reduce the computational and memory cost. The dropout rates of all layers in the encoder are set to 0.1 and all dropout rates in the decoder are set to 0.3. Our models are optimized using Adam (Kingma and Ba, 2015) with $\beta_1 = 0.9$, $\beta_2 = 0.98$. The other optimization hyper-parameters for pre-training and fine-tuning are different. In the pre-training stage, the encoder is initialized with a pre-trained model while the decoder is randomly initialized. Therefore, we used two separate optimizers for the encoder and decoder with a smaller learning rate for the encoder optimizer. Learning rates of the encoder and decoder are set to $2e-5$ and $1e-4$ with 10,000 warmup steps, respectively. We also adopted the same learning rate schedule strategies as Vaswani et al. (2017). We used smaller batch sizes for datasets with less examples (i.e., 1,024 for GIGA-CM, 256 for CNNDM and 128 for NYT) to ensure each epoch has sufficient number of model updates. We trained our models until their convergence of validation perplexities (around 30 epochs on GIGA-CM, 60 epochs on CNNDM and 40 epochs on NYT). One epoch on GIGA-CM takes around 24 hours with 8 Nvidia Tesla V100 GPUs. The time costs for different pre-training tasks are close.

Most of the hyper-parameters in the fine-tuning stage are the same as these in the pre-training stage. The differences are as follows. The learning rates for both the encoder and decoder are set to $2e-5$ with 4,000 warmup steps, since both the encoder and decoder are already pre-trained. We trained our models for 50 epochs (saved per epoch) and selected the best model w.r.t. ROUGE score on the validation set . During decoding, we applied beam search with beam size of 5. Following (Paulus et al., 2018), we also blocked repeating trigrams during beam search and tuned the minimum summary length on the validation set. Similar to the pre-training process, the datasets with less instances were fine-tuned with smaller batch size (i.e., 768 for CNNDM and 64 for NYT).

[6] https://catalog.ldc.upenn.edu/LDC2012T21

[7] We tried $\text{RoBERTa}_{\text{BASE}}$ and obtained inferior results.

Model		R-1	R-2	R-L
Extractive				
Lead3		40.34	17.70	36.57
BERTExt (Liu and Lapata, 2019)		**43.85**	**20.34**	**39.90**
Abstractive				
PTGen (See et al., 2017)		39.53	17.28	36.38
DRM (Paulus et al., 2018)		39.87	15.82	36.90
BottomUp (Gehrmann et al., 2018)		41.22	18.68	38.34
DCA (Celikyilmaz et al., 2018)		41.69	19.47	37.92
BERTAbs (Liu and Lapata, 2019)		42.13	19.60	39.18
UniLM (Dong et al., 2019)		43.47	**20.30**	40.63
TRANSFORMER-S2S		40.43	17.66	37.44
ROBERTA$_{\text{BASE}}$-S2S		42.39	19.52	39.68
ROBERTA-S2S		**43.40**	20.19	**40.67**
Ours				
STEP (in-domain)	SR	**43.77***	20.78*	40.92*
	NSG	43.48	20.70*	40.72*
	MDG	43.72*	20.77*	40.88*
	ALL	43.75*	**20.81***	**40.99***
STEP	SR	44.03*	**21.13***	41.20*
	NSG	44.03*	21.02*	41.18*
	MDG	**44.07***	20.97*	41.22*
	ALL	44.06*	21.07*	**41.24***

Table 1: Results on the test split of CNNDM using full-length F1 based ROUGE-1 (R-1), ROUGE-2 (R-2) and ROUGE-L (R-L). $*$ indicates significant improvements ($p < 0.05$ measured with the ROUGE script) compared to models in the first two blocks.

4.3 Evaluations

We used ROUGE (Lin, 2004) to measure the quality of different summarization model outputs. We reported full-length F1 based ROUGE-1, ROUGE-2 and ROUGE-L scores on CNNDM, while we used the limited-length recall based ROUGE-1, ROUGE-2 and ROUGE-L on NYT following (Durrett et al., 2016). The ROUGE scores are computed using the `ROUGE-1.5.5.pl` script.

Since summaries generated by abstractive models may produce disfluent or ungrammatical outputs, we also evaluated abstractive systems by eliciting human judgements. Following previous work (Cheng and Lapata, 2016; Narayan et al., 2018), 20 documents are randomly sampled from the test split of CNNDM. Participants are presented with a document and a list of outputs generated by different abstractive summarization systems. Then they are asked to rank the outputs according to informativeness (*does the summary capture the informative part of the document?*), fluency (*is the summary grammatical?*), and succinctness (*does the summary express the document clearly in a few words?*)

Model		R-1	R-2	R-L
Extractive				
Lead3		39.58	20.11	35.78
BERTExt (Liu and Lapata, 2019)		**46.66**	**26.35**	**42.62**
Abstractive				
PTGen (See et al., 2017)		43.71	26.40	-
DRM (Paulus et al., 2018)		42.94	26.02	-
BERTAbs (Liu and Lapata, 2019)		**49.02**	**31.02**	**45.55**
TRANSFORMER-S2S		35.75	17.23	31.41
ROBERTA-S2S		47.67	30.56	44.35
Ours				
STEP (in-domain)	SR	48.72	31.07	45.13
	NSG	48.64	30.71	45.08
	MDG	48.76	31.09	45.27
	ALL	**49.11**	**31.39**	**45.49**
STEP	SR	**51.74***	**33.38***	**47.89***
	NSG	51.37*	33.02*	47.93*
	MDG	51.38*	32.99*	47.58*
	ALL	51.14*	33.00*	47.38*

Table 2: Results on the test set of NYT dataset using limited-length recall based ROUGE. $*$ indicates significant improvements ($p < 0.05$ measured with the ROUGE script) to models in the first two blocks.

5 Results

Automatic Evaluation The results on the CNNDM are summarized in Table 1. The first and second blocks show results of previous extractive and abstractive models, respectively. Results of STEP are all listed in the third block. Lead3 is a baseline which simply takes the first three sentences of a document as its summary. BERTExt (Liu and Lapata, 2019) is an extractive model fine-tuning on BERT (Devlin et al., 2019) that outperforms other extractive systems. PTGen (See et al., 2017), DRM (Paulus et al., 2018), and DCA (Celikyilmaz et al., 2018) are sequence-to-sequence learning based models extended with copy and coverage mechanism, reinforcement learning, and deep communicating agents individually. BottomUp (Gehrmann et al., 2018) assisted summary generation with a word prediction model. BERTAbs (Liu and Lapata, 2019) and UniLM (Dong et al., 2019) are both pre-training based SEQ2SEQ summarization models. We also implemented three abstractive models as our baselines. Transformer-S2S is 6-layer SEQ2SEQ Transformer (Vaswani et al., 2017) with random initialization. When we replaced the encoder of Transformer-S2S with RoBERTa$_{\text{BASE}}$ (Liu et al., 2019), RoBERTa$_{\text{BASE}}$-S2S outperforms Transformer-S2S by nearly 2 ROUGE, which demonstrates the effectiveness of pre-trained models. With even larger pre-trained model RoBERTa$_{\text{LARGE}}$, RoBERTa-S2S is compa-

rable with the best published abstractive model UniLM (Dong et al., 2019).

Based on RoBERTa-S2S (the sizes of STEP and RoBERTa-S2S are identical), we study the effect of different pre-training tasks (see Section 3.2). We first pre-train STEP on unlabeled documents of CNNDM training split to get quick feedback[8], denoted as STEP (in-domain). From the top part of the third block in Table 1, we can see that Sentence Reordering (SR), Next Sentence Generation (NSG) and Masked Document Generation (MDG) can all improve RoBERTa-S2S significantly measured by the ROUGE script. Note that according to the ROUGE script, ± 0.22 ROUGE almost always means a significant difference with $p < 0.05$. Interesting, even STEP is pre-trained on 230 million words, it outperforms UniLM that is pre-trained on 3,000 million words (Dong et al., 2019). When we pre-train STEP on even larger dataset (i.e., GIGA-CM), the results are further improved and STEP outperforms all models in comparison, as listed in the bottom part of Table 1.

Table 2 presents results on NYT dataset. Following the same evaluation protocol as Durrett et al. (2016), we adopted the limited-length recall based ROUGE, where we truncated the predicted summaries to the length of the gold ones. Again, the first and second blocks show results of previous extractive and abstractive models, respectively. Results of STEP are listed in the third block. Similar to the trends in CNNDM, STEP leads significant performance gains (with $p < 0.05$) compared to all other models in Table 2.

Among all three pre-training tasks, SR works slightly better than the other two tasks (i.e., NSG and MDG). We also tried to randomly use all the three tasks during training with 1/3 probability each (indicated as ALL). Interesting, we observed that, in general, All outperforms all three tasks when employing unlabeled documents of training splits of CNNDM or NYT, which might be due to limited number of unlabeled documents of the training splits. After adding more data (i.e., GIAG-CM) to pre-training, SR consistently achieves highest ROUGE-2 on both CNNDM and NYT. We conclude that SR is the most effective task for pre-training since sentence reordering task requires comprehensively understanding a document in a wide coverage, going beyond individual words and sentences, which is highly close to the essense of abstractive document summarization.

[8]One epoch takes 3 hours on CNNDM and 0.5 on NYT.

Systems	1st	2nd	3rd	4th	5th	MR
BERTAbs	0.08	0.10	0.18	0.26	0.38	3.76
UniLM	0.06	0.22	0.21	0.26	0.25	3.42
RoBERTa-S2S	0.13	0.15	0.24	0.29	0.19	3.26
STEP	0.25	0.36	0.18	0.10	0.11	2.46
Gold	0.48	0.17	0.19	0.09	0.07	2.10

Table 3: Human evaluation results: proportions of system rankings. MR: mean rank (the lower the better).

Human Evaluation We also conducted human evaluation with 20 documents randomly sampled from the test split of CNNDM. We compared the best preforming STEP model (i.e., pre-training on the GIGA-CM dataset using SR task) with human references (denoted as Gold), RoBERTa-S2S, and two pre-training based models, BERTAbs (Liu and Lapata, 2019) and UniLM (Dong et al., 2019)[9]. Participants were asked to rank the outputs of these systems from best to worst. We report the proportions of system rankings and mean rank (lower is better) in Table 3. The output of STEP is selected as the best for the 25% of cases and we obtained lower mean rank than all systems except for Gold, which shows the participants' preference for our model. Then we converted ranking numbers into ratings (i.e., rank i is converted into $6 - i$) and applied the student t-test on the ratings. STEP is significantly better than all other systems in comparison with $p < 0.05$. But it still lags behind human. One possible reason is that STEP (as well as other systems) only takes the first 512 tokens of a long document as input and thus may lose information residing in the following tokens.

6 Conclusion

We proposed STEP, a SEQ2SEQ transformer pre-training approach, for abstractive document summarization. Specifically, three pre-training tasks are designed, sentence reordering, next sentence generation, and masked document generation. When we only employ the unlabeled documents in the training splits of summarization datasets to pre-training STEP with our proposed tasks, the summarization model based on the pre-trained STEP outperforms the best published abstractive system. Involving large scale data to pre-training leads to larger performance gains. By using the best performing pre-training task, STEP

[9]Outputs of BERTAbs and UniLM are publicly available at https://github.com/nlpyang/PreSumm and https://github.com/microsoft/unilm

achieves 0.8 absolute ROUGE-2 improvements on CNN/DailyMail and 2.4 absolute ROUGE-2 improvements on New York Times. In the future, we would like to investigate other tasks to pre-train the SEQ2SEQ transformer model. Pre-training for unsupervised abstractive summarization is also an interesting direction and worth exploration.

References

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

Dzmitry Bahdanau, KyungHyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proc. of ICLR*.

Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 2015. A large annotated corpus for learning natural language inference. In *Proc. of EMNLP*.

Asli Celikyilmaz, Antoine Bosselut, Xiaodong He, and Yejin Choi. 2018. Deep communicating agents for abstractive summarization. In *Proc. of NAACL*.

Yen-Chun Chen and Mohit Bansal. 2018. Fast abstractive summarization with reinforce-selected sentence rewriting. In *Proc. of ACL*.

Jianpeng Cheng and Mirella Lapata. 2016. Neural summarization by extracting sentences and words. In *Proc. of ACL*.

John M Conroy and Dianne P O'leary. 2001. Text summarization via hidden markov models. In *Proc. of SIGIR*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proc. of ACL*.

Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. 2019. Unified language model pre-training for natural language understanding and generation. In *Proc. of NIPS*.

Greg Durrett, Taylor Berg-Kirkpatrick, and Dan Klein. 2016. Learning-based single-document summarization with compression and anaphoricity constraints. In *Proc. of ACL*.

Elena Filatova and Vasileios Hatzivassiloglou. 2004. Event-based extractive summarization. In *Text Summarization Branches Out*.

Sebastian Gehrmann, Yuntian Deng, and Alexander Rush. 2018. Bottom-up abstractive summarization. In *Proc. of EMNLP*.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O.K. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proc. of ACL*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Proc. of NIPS*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Wan-Ting Hsu, Chieh-Kai Lin, Ming-Ying Lee, Kerui Min, Jing Tang, and Min Sun. 2018. A unified model for extractive and abstractive summarization using inconsistency loss. In *Proc. of ACL*.

Diederik P Kingma and Jimmy Lei Ba. 2015. Adam: A method for stochastic optimization. In *Proc. of ICLR*.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *Proc. of NIPS*.

Julian Kupiec, Jan Pedersen, and Francine Chen. 1995. A trainable document summarizer. In *Proc. of SIGIR*.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Proc. of ACL-Workshop*.

Yang Liu and Mirella Lapata. 2019. Text summarization with pretrained encoders. In *Proc. of EMNLP*.

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. In *Proc. of ACL*.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *Proc. of ACL: System Demonstrations*.

Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. 2017. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. In *Proc. of AAAI*.

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Caglar Gulcehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. In *Proc. of SIGNLL*.

Shashi Narayan, Shay B Cohen, and Mirella Lapata. 2018. Ranking sentences for extractive summarization with reinforcement learning. In *Proc. of NAACL*.

Ani Nenkova and Kathleen McKeown. 2011. Automatic summarization. *Foundations and Trends in Information Retrieval*, 5(2–3):103–233.

Ani Nenkova, Lucy Vanderwende, and Kathleen McKeown. 2006. A compositional context sensitive multi-document summarizer: exploring the factors that influence summarization. In *Proc. of SIGIR*.

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 *Proc. of NAACL: Demonstrations)*.

Romain Paulus, Caiming Xiong, and Richard Socher. 2018. A deep reinforced model for abstractive summarization. In *Proc. of ICLR*.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proc. of NAACL*.

Dragomir Radev, Timothy Allison, Sasha Blair-Goldensohn, John Blitzer, Arda Çelebi, Stanko Dimitrov, Elliott Drabek, Ali Hakim, Wai Lam, Danyu Liu, Jahna Otterbacher, Hong Qi, Horacio Saggion, Simone Teufel, Michael Topper, Adam Winkel, and Zhu Zhang. 2004. MEAD - a platform for multidocument multilingual text summarization. In *Proc. of LREC*.

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/openai-assets/researchcovers/languageunsupervised/language understanding paper. pdf*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8).

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Proc. of EMNLP*.

Evan Sandhaus. 2008. The new york times annotated corpus. *Linguistic Data Consortium, Philadelphia*, 6(12):e26752.

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proc. of ACL*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. In *Proc. of ACL*.

Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2019. Mass: Masked sequence to sequence pre-training for language generation. In *Proc. of ICML*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proc. of NIPS*.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *arXiv preprint arXiv:1906.08237*.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Proc. of NIPS*.

Xingxing Zhang, Mirella Lapata, Furu Wei, and Ming Zhou. 2018. Neural latent extractive document summarization. In *Proc. of ACL*.

Xingxing Zhang, Furu Wei, and Ming Zhou. 2019. HIBERT: Document level pre-training of hierarchical bidirectional transformers for document summarization. In *Proc. of ACL*.