**Pretraining-Based Natural Language Generation for Text Summarization**

**Haoyu Zhang**1∗ , **Jianjun Xu**1 , **Ji Wang**1

1College of Computer, National University of Defense Technology, Changsha, China

{zhanghaoyu10, jjxu, wj}@nudt.edu.cn

# Abstract

arXiv:1902.09243v2 [cs.CL] 12 Apr 2019

In this paper, we propose a novel pretraining-based encoder-decoder framework, which can generate the output sequence based on the input sequence in a two-stage manner. For the encoder of our model, we encode the input sequence into con- text representations using BERT. For the decoder, there are two stages in our model, in the first stage, we use a Transformer-based decoder to generate a draft output sequence. In the second stage, we mask each word of the draft sequence and feed it to BERT, then by combining the input sequence and the draft representation generated by BERT, we use a Transformer-based decoder to predict the refined word for each masked position. To the best of our knowledge, our approach is the first method which applies the BERT into text generation tasks. As the first step in this direction, we evaluate our proposed method on the text summarization task. Experi- mental results show that our model achieves new state-of-the-art on both CNN/Daily Mail and New York Times datasets.

# Introduction

Text summarization generates summaries from input docu- ments while keeping salient information. It is an important task and can be applied to several real-world applications. Many methods have been proposed to solve the text sum- marization problem [[See *et al.*, 2017;](#_bookmark31) [Nallapati *et al.*, 2017;](#_bookmark24) [Zhou *et al.*, 2018;](#_bookmark36) [Gehrmann *et al.*, 2018](#_bookmark18)]. There are two main text summarization techniques: extractive and abstrac- tive. Extractive summarization generates summary by select- ing salient sentences or phrases from the source text, while abstractive methods paraphrase and restructure sentences to compose the summary. We focus on abstractive summariza- tion in this work as it is more flexible and thus can generate more diverse summaries.

Recently, many abstractive approaches are introduced based on neural sequence-to-sequence framework [[Paulus *et*](#_bookmark26)[*al.*, 2018;](#_bookmark26) [See *et al.*, 2017;](#_bookmark31) [Gehrmann *et al.*, 2018;](#_bookmark18) [Li *et al.*,](#_bookmark22) [2018](#_bookmark22)]. Based on the sequence-to-sequence model with copy

∗Contact Author.

mechanism, [[See *et al.*, 2017](#_bookmark31)] incorporates a coverage vector to track and control attention scores on source text. [[Paulus *et*](#_bookmark26)[*al.*, 2018](#_bookmark26)] introduce intra-temporal attention processes in the encoder and decoder to address the repetition and incoherent problem.

There are two issues in previous abstractive methods: 1) these methods use left-context-only decoder, thus do not have complete context when predicting each word. 2) they do not utilize the pre-trained contextualized language models on the decoder side, so it is more difficult for the decoder to learn summary representations, context interactions and language modeling together.

Recently, BERT has been successfully used in various nat- ural language processing tasks, such as textual entailment, name entity recognition and machine reading comprehen- sions. In this paper, we present a novel natural language generation model based on pre-trained language models (we use BERT in this work). As far as we know, this is the first work to extend BERT to the sequence generation task. To ad- dress the above issues of previous abstractive methods, in our model, we design a two-stage decoding process to make good use of BERT’s context modeling ability. On the first stage, we generate the summary using a left-context-only-decoder. On the second stage, we mask each word of the summary and predict the refined word one-by-one using a refine decoder. To further improve the naturalness of the generated sequence, we cooperate reinforcement objective with the refine decoder.

The main contributions of this work are:

* 1. We propose a natural language generation model based on BERT, making good use of the pre-trained language model in the encoder and decoder process, and the model can be trained end-to-end without handcrafted features.
  2. We design a two-stage decoder process. In this archi- tecture, our model can generate each word of the summary considering both sides’ context information.
  3. We conduct experiments on the benchmark datasets CNN/Daily Mail and New York Times. Our model achieves a 33.33 average of ROUGE-1, ROUGE-2 and ROUGE-L on the CNN/Daily Mail, which is state-of-the-art. On the New York Times dataset, our model achieves about 5.6% relative improvement over ROUGE-1.

# Background

## Text Summarization

In this paper, we focus on single-document multi-sentence summarization and propose a supervised abstractive model based on the neural attentive sequence-to-sequence frame- work which consists of two parts: a neural network for the encoder and another network for the decoder. The encoder encodes the input sequence to intermediate representation and the decoder predicts one word at a time step given the input sequence representation vector and previous decoded output. The goal of the model is to maximize the probability of gener- ating the correct target sequences. In the encoding and gener- ation process, the attention mechanism is used to concentrate on the most important positions of text. The learning objec- tive of most sequence-to-sequence models is to minimize the

negative log likelihood of the generated sequence as shown in following equation, where *yt*∗ is the t-th ground-truth sum- mary token.

put *hi*. The last layer outputs is context encoding of input sequence.

*N*

Σ

*oi* = *hi* + *eij*(*hjWV* ) (4)

*j*=1

Despite the wide usage and huge success, there is also a mismatch problem between these pre-trained context en- coders and sequence-to-sequence models. The issue is that while using a pre-trained context encoder like BERT, they model token-level representations by conditioning on both di- rection context. During pre-training, they are fed with com- plete sequences. However, with a left-context-only decoder, these pre-trained language models will suffer from incom- plete and inconsistent context and thus cannot generate good enough context-aware word representations, especially dur- ing the inference process.

|*y*|

Σ

*Loss* = − log *P* (*yt*∗|*y<*∗ *t, X*) (1)

*t*=1

However, with this objective, traditional sequence gener- ation models consider only one direction context in the de- coding process, which could cause performance degradation since complete context of one token contains preceding and following tokens, thus feeding only preceded decoded words to the decoder so that the model may generate unnatural se- quences. For example, attentive sequence-to-sequence mod- els often generate sequences with repeated phrases which harm the naturalness. Some previous works mitigate this problem by improving the attention calculation process, but in this paper we show that feeding bi-directional context in- stead of left-only-context can better alleviate this problem.

## Bi-Directional Pre-Trained Context Encoders

Recently, context encoders such as ELMo, GPT, and BERT have been widely used in many NLP tasks. These models are pre-trained on a huge unlabeled corpus and can generate better contextualized token embeddings, thus the approaches built on top of them can achieve better performance.

Since our method is based on BERT, we illustrate the pro- cess briefly here. BERT consists of several layers. In each layer there is first a multi-head self-attention sub-layer and then a linear affine sub-layer with the residual connection. In each self-attention sub-layer the attention scores *eij* are first calculated as Eq. [(2) (3),](#_bookmark1) in which *de* is output dimension, and *WQ, WK, WV* are parameter matrices.

(*hiWQ*)(*hjWK*)*T*

# Model

In this section, we describe the structure of our model, which learns to generate an abstractive multi-sentence summary from a given source document.

Based on the sequence-to-sequence framework built on top of BERT, we first design a refine decoder at word-level to tackle the two problems described in the above section. We also introduce a discrete objective for the refine decoders to reduce the exposure bias problem. The overall structure of our model is illustrated in Figure [1.](#_bookmark2)

## Problem Formulation

We denote the input document as *X* = *x*1*, . . . , xm* where *xi* represents one source token. The corresponding sum- mary is denoted as *Y* = *y*1*, . . . , yL* , *L* represents the sum- mary length.

{ }

∈ X

{ }

Given input document *X*, we first predict the summary draft by a left-context-only decoder, and then using the gen- erated summary draft we can condition on both context sides and refine the content of the summary. The draft will guide and constrain the refine process of summary.

## Summary Draft Generation

The summary draft is based on the sequence-to-sequence model. On the encoder side the input document *X* is en- coded into representation vectors *H* = *h*1*, . . . , hm* , and then fed to the decoder to generate the summary draft *A* =

{ }

{*a*1*, . . . , a*|*a*|}.

### Encoder

We simply use BERT as the encoder. It first maps the input

*aij* = √*d*

*e*

(2)

sequence to word embeddings and then computes document

*eij* =

exp *eij*

Σ*N* exp *eik*

*k*=1

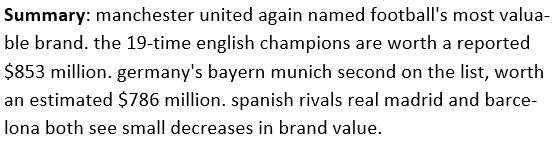
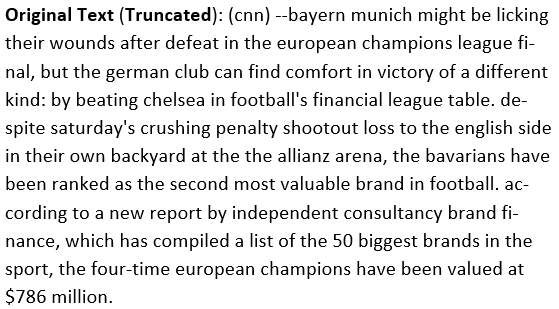
(3)

embeddings as the encoder’s output, denoted by following equation.

Then the output is calculated as Eq. [(4),](#_bookmark0) which is the weighted sum of previous outputs *h* added by previous out-

*H* = *BERT* (*x*1*, . . . , xm*) (5)

× 𝐿



**Summary Output**

**Document embedding**

**BERT**

**Summary Draft Output**

**Mask**

**Multi-Head** × 𝑁

**Attention Decoder**

**Multi-Head**

**Att der**

**ention Deco**

× 𝑁

**Summary Draft embedding**

× 𝐿

**Summary Draft Output (Shifted Right)**

**BERT**

**Document**

Stage 1 Stage 2

Figure 1: Model Overview, N represents decoder layer number and L represents summary length.

### Summary Draft Decoder

In the draft decoder, we first introduce BERT’s word em- bedding matrix to map the previous summary draft outputs *y*1*, . . . , yt* 1 into embeddings vectors *q*1*, . . . , qt* 1 at t- th time step. Note that as the input sequence of the decoder is

{ } { }− −

not complete, we do not use the BERT network to predict the context vectors here.

Then we introduce an *N* layer Transformer decoder to learn the conditional probability *P* (*A H*). Transformer’s encoder-decoder multi-head attention helps the decoder learn soft alignments between summary and source document. At the t-th time step, the draft decoder predicts output prob- ability conditioned on previous outputs and encoder hid- den representations as shown in Eq. [(6),](#_bookmark4) in which *q<t* =

|

*q*1*, . . . , qt* 1 . Each generated sequence will be truncated in the first position of a special token ’[PAD]’. The total sum-

{ }−

This issue exists when using any other contextualized word representations, so we design a refine process to mitigate it in our approach which will be described in the next sub-section.

### Copy Mechanism

As some summary tokens are out-of-vocabulary words and occurs in input document, we incorporate copy mecha- nism [[Gu *et al.*, 2016](#_bookmark19)] based on the Transformer decoder, we will describe it briefly.

At decoder time step *t*, we first calculate the attention prob- ability distribution over source document *X* using the bi- linear dot product of the last layer decoder output of Trans- former *ot* and the encoder output *hj*, shown in Eq. [(8) (9).](#_bookmark3)

*uj* = *otWchj* (8)

*t*

exp *uj*

Σ

mary draft decoder progress is shown in Stage 1 of Figure [1.](#_bookmark2)

*αj* = *t*

(9)

*t N*

*t*

*k*=1

∈

exp *uk*

*Pvocab*(*w*) = *fdec*(*q<t, H*) (6)

*t*

We then calculate copying gate *gt* [0*,* 1], which makes a soft choice between selecting from source and generating

*Ldec* = Σ|*a*|

*t*=1

— log *P* (*at* = *yt*∗|*a<t, H*) (7)

from vocabulary, *Wc, Wg, bg* are parameters:

As Eq. [(7)](#_bookmark4) shows, the decoder’s learning objective is to minimize negative likelihood of conditional probability, in which *yt*∗ is the t-th ground truth word of summary.

However a decoder with this structure is not sufficient

enough: if we use the BERT network in this decoder, then during training and inference, in-complete context(part of sentence) is fed into the BERT module, and although we can fine-tune BERT’s parameters, the input distribution is quite

*gt* = *sigmoid*(*Wg* · [*ot, h*] + *bg*) (10)

Using *gt* we calculate the weighted sum of copy probabil- ity and generation probability to get the final predicted prob- ability of extended vocabulary + , where is the set of out of vocabulary words from the source document. The final probability is calculated as follow:

V X X

different from the pre-train process, and thus harms the qual- ity of generated context representations.

If we just use the embedding matrix here, it will be more difficult for the decoder with fresh parameters to learn to

*vocab i*

*t t*

Σ*P* (*w*) = (1 − *g* )*P* (*w*) + *gt t t*

*α*

*i*:*wi*=*w*

## Summary Refine Process

(11)

model representations as well as vocabulary probabilities, from a relative small corpus compared to BERT’s huge pre- training corpus. In a word, the decoder cannot utilize BERT’s ability to generate high quality context vectors, which will also harm performance.

The main reason to introduce the refine process is to enhance the decoder using BERT’s contextualized representations, so we do not modify the encoder and reuse it during this process. On the decoder side, we propose a new word-level refine decoder. The refine decoder receives a generated summary

draft as input, and outputs a refined summary. As Figure [1](#_bookmark2) Stage 2 shows, it first masks each word in the summary draft one by one, then feeds the draft to BERT to generate context vectors. Finally it predicts a refined summary word using an *N* layer Transformer decoder which is the same as the draft decoder. At t-th time step the t-th word of input summary is masked, and the decoder predicts the refined word given other words of the summary.

The learning objective of this process is shown in Eq. [(12),](#_bookmark5) *yt* is the t-th summary word and *yt*∗ for the ground-truth sum- mary word, and *a*/=*t* = {*a*1*, . . . , at*−1*, at*+1*, . . . , a*|*y*|}.

|*y*|

Σ

*Lrefine* = − log *P* (*yt* = *yt*∗|*a*=/ *t, H*) (12)

*t*=1

From the view of BERT or other contextualized embed- dings, the refine decoding process provides a more complete input sequence which is consistent with their pre-training pro- cesses. Intuitively, this process works as follows: first the draft decoder writes a summary draft based on a document, and then the refine decoder edits the draft. It concentrates on one word at a time, based on the source document as well as other words.

We design the word-level refine decoder because this pro- cess is similar to the cloze task in BERT’s pre-train pro- cess, therefore by using the ability of the contextual language model the decoder can generate more fluent and natural se- quences.

The parameters are shared between the draft decoder and refine decoder, as we find that using individual parameters the model’s performance degrades a lot. The reason may be that we use teach-forcing during training, and thus the word- level refine decoder learns to predict words given all the other ground-truth words of summary. This objective is similar to the language model’s pre-train objective, and is probably not enough for the decoder to learn to generate refined sum- maries. So in our model all decoders share the same parame- ters.

### Mixed Objective

For summarization, ROUGE is usually used as the evalua- tion metric, however during model training the objective is to maximize the log likelihood of generated sequences. This mis-match harms the model’s performance. Similar to pre- vious work [[Krys´cin´ski *et al.*, 2018](#_bookmark21)], we add a discrete ob- jective to the model, and optimize it by introducing the pol- icy gradient method. The discrete objective for the summary draft process is as shown in Eq. [(13),](#_bookmark6) where *as* is the draft summary sampled from predicted distribution, and *R*(*as*) is the reward score compared with the ground-truth sum- mary, we use ROUGE-L in our experiment. To balance be- tween optimizing the discrete objective and generating read- able sequences, we mix the discrete objective with maximum- likelihood objective. Eq. [(14)](#_bookmark6) shows the final objective for the draft process, note here *Ldec* is *logP* (*a x*). In the refine process we introduce similar objectives.

— |

## Learning and Inference

During model training, the objective of our model is sum of the two processes, jointly trained using ”teacher-forcing” al- gorithm. During training we feed the ground-truth summary to each decoder and minimize the following objective.

*Lmodel* = *L*ˆ*dec* + *L*ˆ*ref ine* (15) At test time, each time step we choose the predicted word

by *y*ˆ = *argmaxy' P* (*y*′ *x*), use beam search to generate the draft summaries, and use greedy search to generate the re-

|

fined summaries.

# Experiment

## Settings

In this work, all of our models are built on *BERTBASE*, al- though another larger pre-trained model with better perfor- mance (*BERTLARGE*) has published but it costs too much time and GPU memory. We use WordPiece embeddings with a 30,000 vocabulary which is the same as BERT. We set the layer of transformer decoders to 12(8 on NYT50), and set the attention heads number to 12(8 on NYT50), set fully-connected sub-layer hidden size to 3072. We train the model using an Adam optimizer with learning rate of 3*e* 4,

−

*β*1 = 0*.*9, *β*2 = 0*.*999 and *ϵ* = 10−9 and use a dynamic

learning rate during the training process. For regularization, we use dropout [[Srivastava *et al.*, 2014](#_bookmark33)] and label smooth- ing [[Szegedy *et al.*, 2016](#_bookmark34)] in our models and set the dropout rate to 0.15, and the label smoothing value to 0.1. We set the RL objective factor *γ* to 0.99.

During training, we set the batch size to 36, and train for 4 epochs(8 epochs for NYT50 since it has many fewer train- ing samples), after training the best model are selected from last 10 models based on development set performance. Due to GPU memory limit, we use gradient accumulation, set ac- cumulate step to 12 and feed 3 samples at each step. We use beam size 4 and length penalty of 1.0 to generate logical form sequences.

We filter repeated tri-grams in beam-search process by set- ting word probability to zero if it will generate an tri-gram which exists in the existing summary. It is a nice method to avoid phrase repetition since the two datasets seldom contains repeated tri-grams in one summary. We also fine tune the gen- erated sequences using another two simple rules. When there are multi summary sentences with exactly the same content, we keep the first one and remove the other sentences; we also remove sentences with less than 3 words from the result.

### Datasets

To evaluate the performance of our model, we conduct experi- ments on CNN/Daily Mail dataset, which is a large collection of news articles and modified for summarization. Follow- ing [[See *et al.*, 2017](#_bookmark31)] we choose the non-anonymized version of the dataset, which consists of more than 280,000 training samples and 11490 test set samples.

We also conduct experiments on the New York Times(NYT) dataset which also consists of many news

*rl dec*

*L*

= *R*(*as*) · [− log(*P* (*as*|*x*))] (13)

articles. The original dataset can be applied here.[1](#_bookmark7) In our

*L*ˆ*dec* = *γ* ∗ *Lrl*

*dec*

+ (1 − *γ*) ∗ *Ldec* (14)

1<http://duc.nist.gov/>

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Model | ROUGE-1 | ROUGE-2 | ROUGE-L | R-AVG |
| Extractive |  | | | |
| lead-3 [[See *et al.*, 2017](#_bookmark31)] | 40.34 | 17.70 | 36.57 | 31.54 |
| SummmaRuNNer [[Nallapati *et al.*, 2017](#_bookmark24)] | 39.60 | 16.20 | 35.30 | 30.37 |
| Refresh [[Narayan *et al.*, 2018](#_bookmark25)] | 40.00 | 18.20 | 36.60 | 31.60 |
| DeepChannel [[Shi *et al.*, 2018](#_bookmark32)] | 41.50 | 17.77 | 37.62 | 32.30 |
| rnn-ext + RL [[Chen and Bansal, 2018](#_bookmark15)] | 41.47 | 18.72 | 37.76 | 32.65 |
| MASK-*LMglobal* [[Chang *et al.*, 2019](#_bookmark14)] | 41.60 | 19.10 | 37.60 | 32.77 |
| NeuSUM [[Zhou *et al.*, 2018](#_bookmark36)] | 41.59 | 19.01 | 37.98 | 32.86 |
| Abstractive |  | | | |
| PointerGenerator+Coverage [[See *et al.*, 2017](#_bookmark31)] | 39.53 | 17.28 | 36.38 | 31.06 |
| ML+RL+intra-attn [[Paulus *et al.*, 2018](#_bookmark26)] | 39.87 | 15.82 | 36.90 | 30.87 |
| inconsistency loss[[Hsu *et al.*, 2018](#_bookmark20)] | 40.68 | 17.97 | 37.13 | 31.93 |
| Bottom-Up Summarization [[Gehrmann *et al.*, 2018](#_bookmark18)] | 41.22 | 18.68 | 38.34 | 32.75 |
| DCA [[Celikyilmaz *et al.*, 2018](#_bookmark13)] | 41.69 | 19.47 | 37.92 | 33.11 |
| Ours |  | | | |
| One-Stage | 39.50 | 17.87 | 36.65 | 31.34 |
| Two-Stage | 41.38 | 19.34 | 38.37 | 33.03 |
| Two-Stage + RL | **41.71** | **19.49** | **38.79** | **33.33** |

Table 1: ROUGE F1 results for various models and ablations on the CNN/Daily Mail test set. R-AVG calculates average score of Rouge-1, Rouge-2 and Rouge-L.

experiment, we follow the dataset splits and other pre-process settings of [[Durrett *et al.*, 2016](#_bookmark17)]. We first filter all samples without a full article text or abstract and then remove all samples with summaries shorter than 50 words. Then we choose the test set based on the date of publication(all examples published after January 1, 2007). The final dataset contains 22,000 training samples and 3,452 test samples and is called NYT50 since all summaries are longer than 50 words.

We tokenize all sequences of the two datasets using the WordPiece tokenizer. After tokenizing, the average article length and summary length of CNN/Daily Mail are 691 and 51, and NYT50’s average article length and summary length are 1152 and 75. We truncate the article length to 512, and the summary length to 100 in our experiment(max summary length is set to 150 on NYT50 as its average golden summary length is longer).

### Evaluation Metrics

On CNN/Daily Mail dataset, we report the full-length F- 1 score of the ROUGE-1, ROUGE-2 and ROUGE-L met- rics, calculated using PyRouge package[2](#_bookmark9) and the Porter stem- mer option. On NYT50, following [[Paulus *et al.*, 2018](#_bookmark26)] we evaluate limited length ROUGE recall score(limit the gener- ated summary length to the ground truth length). We split NYT50 summaries into sentences by semicolons to calculate the ROUGE scores.

## Results and Analysis

Table [1](#_bookmark8) gives the results on CNN/Daily Mail dataset, we com- pare the performance of many recent approaches with our model. We classify them to two groups based on whether they are extractive or abstractive models. As the last line of the ta- ble lists, the ROUGE-1 and ROUGE-2 score of our full model

2pypi.python.org/pypi/pyrouge/0.1.3

is comparable with DCA, and outperforms on ROUGE-L. Also, compared to extractive models NeuSUM and MASK- *LMglobal*, we achieve slight higher ROUGE-1. Except the four scores, our model outperforms these models on all the other scores, and since we have 95% confidence interval of at most ± 0.20, these improvements are statistically significant.

### Ablation Analysis

As the last four lines of Table [1](#_bookmark8) show, we conduct an abla- tion study on our model variants to analyze the importance of each component. We use three ablation models for the ex- periments. One-Stage: A sequence-to-sequence model with copy mechanism based on BERT; Two-Stage: Adding the re- fine decoder to the One-Stage model; Two-Stage + RL: Full model with refine process cooperated with RL objective.

First, we compare the Two-Stage+RL model with Two- Stage ablation, we observe that the full model outperforms by 0.30 on average ROUGE, suggesting that the reinforce- ment objective helps the model effectively. Then we analyze the effect of refine process by removing it from the Two-Stage model, we observe that without the refine process the average ROUGE score drops by 1.69. The ablation study proves that each module is necessary for our full model, and the improve- ments are statistically significant on all metrics.

### Effects of Summary Length

To evaluate the impact of summary length on model per- formance, we compare the average rouge score improve- ments of our model with different length of ground-truth sum- maries. As the above sub-figure of Figure [2](#_bookmark10) shows, compared to Pointer-Generator with Coverage, on length interval 40- 80(occupies about 70% of test set) the improvements of our model are higher than shorter samples, confirms that with bet- ter context representations, in longer documents our model can achieve higher performance.

As shown in the below sub-figure of Figure [2,](#_bookmark10) compared

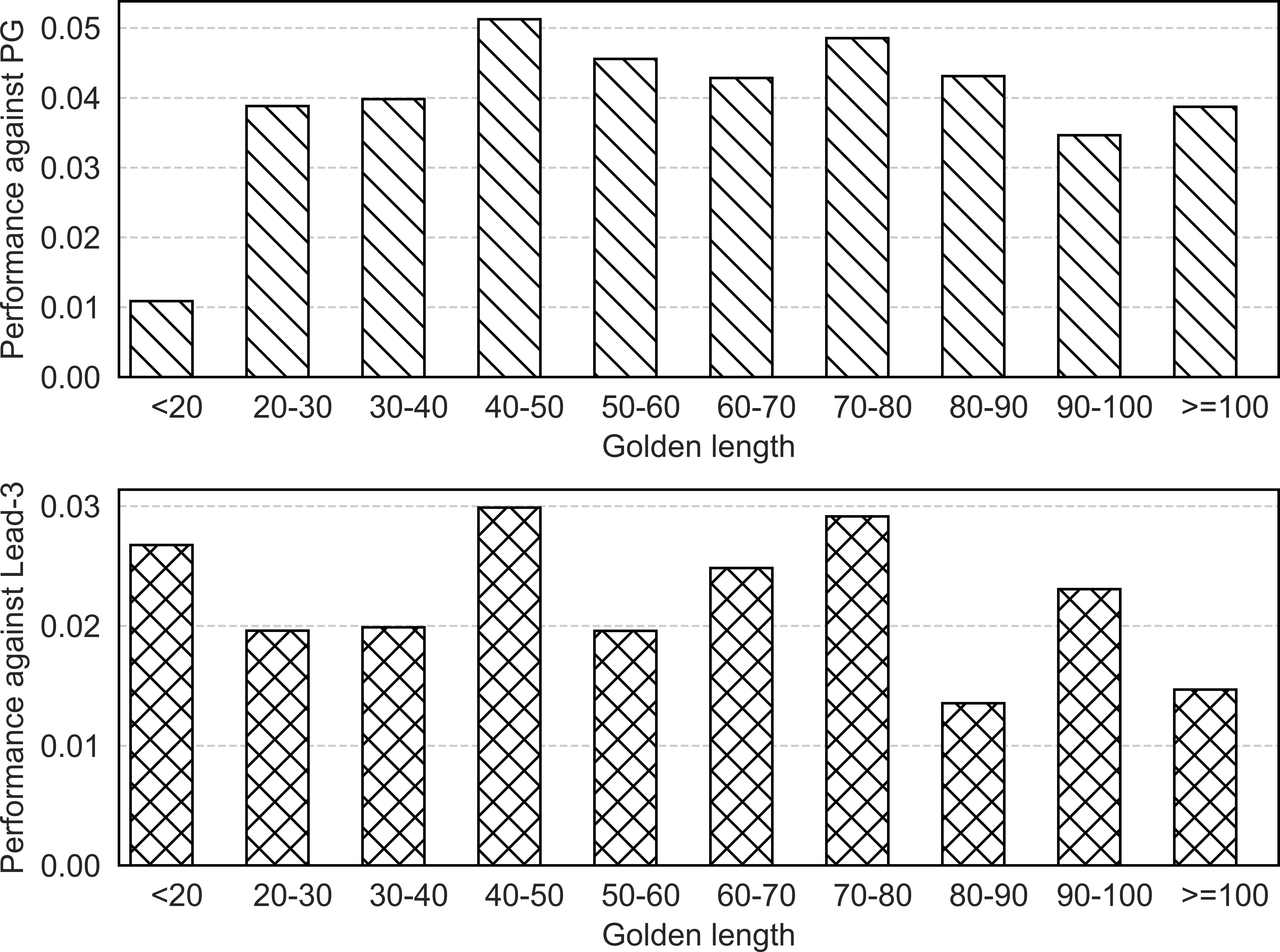


Figure 2: Average ROUGE-L improvement on CNN/Daily mail test set samples with different golden summary length.

to extractive baseline: Lead-3 [[See *et al.*, 2017](#_bookmark31)], the advan- tage of our model will fall when golden summary length is greater than 80. This probably because that we truncate the long documents and golden summaries and cannot get full in- formation, it could also because that the training data in these intervals is too few to train an abstractive model, so simple extractive method will not fall too far behind.

|  |  |  |
| --- | --- | --- |
| Model | R-1 | R-2 |
| First sentences | 28.6 | 17.3 |
| First *k* words | 35.7 | 21.6 |
| Full [[Durrett *et al.*, 2016](#_bookmark17)] | 42.2 | 24.9 |
| ML+RL+intra-attn [[Paulus *et al.*, 2018](#_bookmark26)] | 42.94 | 26.02 |
| Two-Stage + RL (Ours) | **45.33** | **26.53** |

Table 2: Limited length ROUGE recall results on the NYT50 test set.

## Additional Results on NYT50

Table [2](#_bookmark11) reports experiment results on the NYT50 corpus. Since the short summary samples are filtered, NYT50 has av- erage longer summaries than CNN/Daily Mail. So the model needs to catch long-term dependency of the sequences to gen- erate good summaries.

The first two lines of Table [2](#_bookmark11) show results of the two baselines introduced by [[Durrett *et al.*, 2016](#_bookmark17)]: these base- lines select first n sentences, or select the first k words from the original document. Also we compare performance of our model with two recent models, we see 2.39 ROUGE-1 improvements compared to the ML+RL with intra-attn ap- proach(previous SOTA) carries over to this dataset, which is a large margin. On ROUGE-2, our model also get an im- provement of 0.51. The experiment proves that our approach can outperform competitive methods on different data distri- butions.

# Related work

## Text Summarization

Text summarization models are usually classified to ab- stractive and extractive ones. Recently, extractive models like DeepChannel [[Shi *et al.*, 2018](#_bookmark32)], rnn-ext+RL [[Chen and](#_bookmark15) [Bansal, 2018](#_bookmark15)] and NeuSUM [[Zhou *et al.*, 2018](#_bookmark36)] achieve higher performances using well-designed structures. For ex- ample, DeepChannel propose a salience estimation network and iteratively extract salient sentences. [[Zhang *et al.*, 2018](#_bookmark35)] train a sentence compression model to teach another latent variable extractive model.

Also, several recent works focus on improving abstractive methods. [[Gehrmann *et al.*, 2018](#_bookmark18)] design a content selector to over-determine phrases in a source document that should be part of the summary. [[Hsu *et al.*, 2018](#_bookmark20)] introduce inconsis- tency loss to force words in less attended sentences(which de- termined by extractive model) to have lower generation prob- abilities. [[Li *et al.*, 2018](#_bookmark22)] extend seq2seq model with an information selection network to generate more informative summaries.

## Pre-trained language models

Pre-trained word vectors [[Mikolov *et al.*, 2013;](#_bookmark23) [Pennington *et*](#_bookmark27)[*al.*, 2014;](#_bookmark27) [Bojanowski *et al.*, 2017](#_bookmark12)] have been widely used in many NLP tasks. More recently, pre-trained language models (ELMo, GPT and BERT), have also achieved great success on several NLP problems such as textual entailment, semantic similarity, reading comprehension, and question answering [[Peters *et al.*, 2018;](#_bookmark28) [Radford *et al.*, 2018;](#_bookmark30) [Devlin *et al.*, 2018](#_bookmark16)].

Some recent works also focus on leveraging pre-trained language models in summarization. [[Radford *et al.*, 2017](#_bookmark29)] pretrain a language model and use it as the sentiment anal- yser when generating reviews of goods. [[Krys´cin´ski *et al.*,](#_bookmark21) [2018](#_bookmark21)] train a language model on golden summaries, and then use it on the decoder side to incorporate prior knowledge.

In this work, we use BERT(which is a pre-trained language model using large scale unlabeled data) on the encoder and decoder of a seq2seq model, and by designing a two stage de- coding structure we build a competitive model for abstractive text summarization.

# Conclusion and Future Work

In this work, we propose a two-stage model based on sequence-to-sequence paradigm. Our model utilize BERT on both encoder and decoder sides, and introduce reinforce ob- jective in learning process. We evaluate our model on two benchmark datasets CNN/Daily Mail and New York Times, the experimental results show that compared to previous sys- tems our approach effectively improves performance.

Although our experiments are conducted on summariza- tion task, our model can be used in most natural language generation tasks, such as machine translation, question gener- ation and paraphrasing. The refine decoder and mixed objec- tive can also be applied on other sequence generation tasks, and we will investigate on them in future work.

# References

[Bojanowski *et al.*, 2017] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vec- tors with subword information. *Transactions of the Asso- ciation for Computational Linguistics*, 5:135–146, 2017.

[Celikyilmaz *et al.*, 2018] Asli Celikyilmaz, Antoine Bosse- lut, Xiaodong He, and Yejin Choi. Deep Communicat- ing Agents for Abstractive Summarization. *arXiv preprint arXiv:1803.10357*, 2018.

[Chang *et al.*, 2019] Ming-Wei Chang, Kristina Toutanova, Kenton Lee, and Jacob Devlin. Language Model Pre-training for Hierarchical Document Representations. *arXiv preprint arXiv:1901.09128*, 2019.

[Chen and Bansal, 2018] Yen-Chun Chen and Mohit Bansal. Fast abstractive summarization with reinforce-selected sentence rewriting. *arXiv preprint arXiv:1805.11080*, 2018.

[Devlin *et al.*, 2018] Jacob Devlin, Ming-Wei Chang, Ken- ton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understand- ing. *arXiv preprint arXiv:1810.04805*, 2018.

[Durrett *et al.*, 2016] Greg Durrett, Taylor Berg-Kirkpatrick, and Dan Klein. Learning-based single-document summa- rization with compression and anaphoricity constraints. In *Proceedings of the 54th Annual Meeting of the Associa- tion for Computational Linguistics, ACL 2016, August 7- 12, 2016, Berlin, Germany, Volume 1: Long Papers*, 2016.

[Gehrmann *et al.*, 2018] Sebastian Gehrmann, Yuntian Deng, and Alexander M Rush. Bottom-up abstractive summarization. *arXiv preprint arXiv:1808.10792*, 2018.

[Gu *et al.*, 2016] Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O. K. Li. Incorporating Copying Mechanism in Sequence-to-Sequence Learning. In *ACL*, 2016.

[Hsu *et al.*, 2018] Wan-Ting Hsu, Chieh-Kai Lin, Ming- Ying Lee, Kerui Min, Jing Tang, and Min Sun. A unified model for extractive and abstractive summarization us- ing inconsistency loss. *arXiv preprint arXiv:1805.06266*, 2018.

[Krys´cin´ski *et al.*, 2018] Wojciech Krys´cin´ski, Romain Paulus, Caiming Xiong, and Richard Socher. Improv- ing abstraction in text summarization. *arXiv preprint arXiv:1808.07913*, 2018.

[Li *et al.*, 2018] Wei Li, Xinyan Xiao, Yajuan Lyu, and Yuanzhuo Wang. Improving Neural Abstractive Document Summarization with Explicit Information Selection Mod- eling. In *EMNLP*, pages 1787–1796, 2018.

[Mikolov *et al.*, 2013] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed repre- sentations of words and phrases and their compositional- ity. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[Nallapati *et al.*, 2017] Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. Summarunner: A recurrent neural net- work based sequence model for extractive summarization

of documents. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, pages 3075–3081, 2017.

[Narayan *et al.*, 2018] Shashi Narayan, Shay B Cohen, and Mirella Lapata. Ranking Sentences for Extractive Sum- marization with Reinforcement Learning. *arXiv preprint arXiv:1802.08636*, 2018.

[Paulus *et al.*, 2018] Romain Paulus, Caiming Xiong, Richard Socher, and Palo Alto. A deep reinforced model for abstractive summarization. *ICLR*, pages 1–13, 2018.

[Pennington *et al.*, 2014] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[Peters *et al.*, 2018] Matthew E Peters, Mark Neumann, Mo- hit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word repre- sentations. *arXiv preprint arXiv:1802.05365*, 2018.

[Radford *et al.*, 2017] Alec Radford, Rafal Jo´zefowicz, and Ilya Sutskever. Learning to generate reviews and discov- ering sentiment. *CoRR*, abs/1704.01444, 2017.

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

[See *et al.*, 2017] Abigail See, Peter J. Liu, and Christo- pher D. Manning. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Lin- guistics, ACL*, pages 1073–1083, 2017.

[Shi *et al.*, 2018] Jiaxin Shi, Chen Liang, Lei Hou, Juanzi Li, Zhiyuan Liu, and Hanwang Zhang. DeepChannel: Salience Estimation by Contrastive Learning for Extrac- tive Document Summarization. *CoRR*, 2018.

[Srivastava *et al.*, 2014] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdi- nov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[Szegedy *et al.*, 2016] Christian Szegedy, Vincent Van- houcke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016*, pages 2818–2826, 2016.

[Zhang *et al.*, 2018] Xingxing Zhang, Mirella Lapata, Furu Wei, and Ming Zhou. Neural latent extractive document summarization. *arXiv preprint arXiv:1808.07187*, 2018.

[Zhou *et al.*, 2018] Qingyu Zhou, Nan Yang, Furu Wei, Shaohan Huang, Ming Zhou, and Tiejun Zhao. Neural document summarization by jointly learning to score and select sentences. In *Proceedings of the 56th Annual Meet- ing of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1:*

*Long Papers*, pages 654–663, 2018.