Text Summarization with Pre-Trained Encoders (ACL 2019)

Paper link: https://www.aclweb.org/anthology/D19-1387.pdf

Objective and Contribution:

The main task of this paper is to showcase how Bidirectional Encoder Representations from Transformers (BERT) can be usefully applied in text summarization and also to propose a general framework for both extractive and abstractive models. The authors of the paper have proposed a document-level encoder which is based on BERT. With the help of which it will be able to express the contextual semantics of a document and get the representations for its sentences. The name of the proposed architecture is BERTSUM, which is BERT architecture for summarisation.

The extractive text summarisation module is built on top of the document-level encoder. This is done by applying several intersentence Transformer layers on it. Whereas, for the task of abstractive text summarization, they have proposed a new fine-tuning schedule. This can take on different optimizers for the encoder and the decoder. This can help in reducing the mismatch between the encoder which is pre-trained and a decoder. In order to improve the quality of the summarisation, authors have experimented with two-stage fine-tuning. Here the encoder is fine-tuned first with an extractive summarisation as objective and fine-tuned second time on the abstractive summarization task.

Language model pre-training has been used in many different NLP tasks like sentiment analysis, question answering (QnA), natural language inference, named entity recognition (NER),and textual similarity. Some of the State-of-the-art pretrained models include ELMo, GPT, and BERT. The advantage of using BERT is that it combines both word as well as sentence representations in a single very large Transformer.

So, this paper has a main contribution in observing the impact of Language model pre-training for text summarization tasks. The aim of this paper is to compress a document into a shorter version and at the same time it should be able to preserve most of its meaning. For the task of Abstractive summarization, in order to generate the summaries which contain the most important words and phrases from the document which are not present in the source document itself requires capabilities of language generation. That is, it should be able to recreate complemetely new words as well as phrases. Extractive summarization can be done by using the strategy of binary classification.

Datasets:

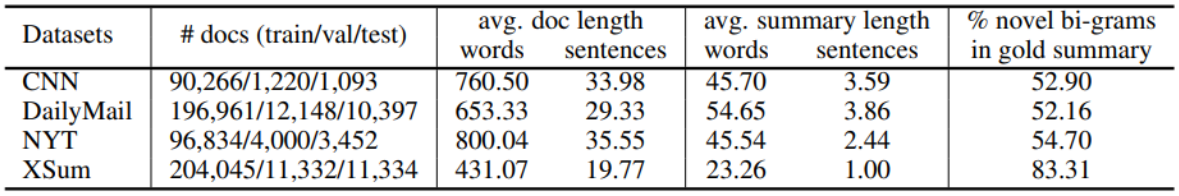
There are three different datasets which are single-document news summarization datasets. These datasets fairly represent various summary styles along with conventions for writing. The authors of this paper have experimentally shown that the proposed models achieve state-of-the-art results under both extractive and abstractive settings for the datasets mentioned below.

The datasets used are:

1. CNN/DailyMail:
   This is an English-language dataset containing news articles and associated highlights. SO it gives a few bullet points giving a brief overview of the article. It has over 300k news articles which are unique. The dataset has id, article and highlights columns.

2. NYT:
   This is New York Times Annotated Corpus which contains 110,540 articles which were published by New York Times from January, 1987 to June, 2007. This dataset contains the abstractive summaries of news articles.

3. XSum:
   This is an extreme summarisation dataset and hence the name xsum. This dataset contains 226,711 news articles along with a one-sentence summary, answering the question “What is this article about?”. This dataset has three features which are id, document and summary.



Datasets	# docs (train/val/test)	avg. doc length		avg. summary length		% novel bi-grams in gold summary
		words	sentences	words	sentences	
CNN	90,266/1,220/1,093	760.50	33.98	45.70	3.59	52.90
DailyMail	196,961/12,148/10,397	653.33	29.33	54.65	3.86	52.16
NYT	96,834/4,000/3,452	800.04	35.55	45.54	2.44	54.70
XSum	204,045/11,332/11,334	431.07	19.77	23.26	1.00	83.31

Table 1: Comparison of summarization datasets: size of training, validation, and test sets and average document and summary length (in terms of words and sentences). The proportion of novel bi-grams that do not appear in source documents but do appear in the gold summaries quantifies corpus bias towards extractive methods.

BERTSUM Architecture:

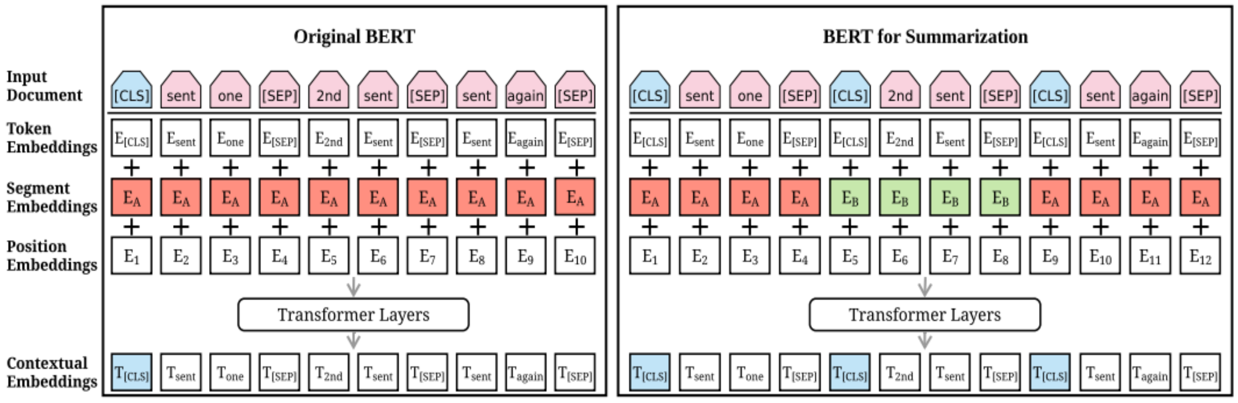

Original BERT
BERT for Summarization
Input Document
Token Embeddings
Segment Embeddings
Position Embeddings
Contextual Embeddings
[CLS] sent one [SEP] 2nd sent [SEP] sent again [SEP]
[CLS] sent one [SEP] [CLS] 2nd sent [SEP] [CLS] sent again [SEP]
$E_{[CLS]}$ E_{sent} E_{one} $E_{[SEP]}$ E_{2nd} E_{again}
E_A E_B
E_1 … E_{12}
Transformer Layers
$T_{[CLS]}$ T_{sent} T_{one} $T_{[SEP]}$ T_{2nd} T_{again}

Figure 1: Architecture of the original BERT model (left) and BERTSUM (right). The sequence on top is the input document, followed by the summation of three kinds of embeddings for each token. The summed vectors are used as input embeddings to several bidirectional Transformer layers, generating contextual vectors for each token. BERTSUM extends BERT by inserting multiple [CLS] symbols to learn sentence representations and using interval segmentation embeddings (illustrated in red and green color) to distinguish multiple sentences.

Summarization Encoder:

In the proposed model in order to represent individual sentences, they have inserted external [CSL] tokens at the start of each sentence. Each [CLS] symbol collects features for the sentence preceding it. They used interval segment embeddings for distinguishing multiple sentences within a document. For each senti in a given document they have assigned a segment embedding EA or EB depending on whether ‘i’ is odd or even.

For example,
Given a document: $[sent1;\ sent2;\ sent3;\ sent4;\ sent5]$
assign embeddings: $[EA;\ EB;\ EA;\ EB;\ EA]$

This way, document representations are learned hierarchically where lower Transformer layers represent adjacent sentences, while higher layers, in combination with self-attention, represent multi-sentence discourse. Position embeddings in the original BERT model have a maximum length of 512; we overcome this limitation by adding more position embeddings that are initialized randomly and fine tuned with other parameters in the encoder.

Extractive Summarization:

Let d denote a document containing sentences $[sent1;\ sent2;\ \cdots;\ sentm]$ where $senti$ is the $i'\ th$sentence in the document. Then the Extractive summarization task can be defined as the task of assigning a label $yi \in \{0,\ 1\}$to each sent_i, indicating whether the sentence should be included in the summary or not.

$$yi\ =\ 1;\ Include\ the\ sentence\ in\ the\ summary$$
$$yi\ =\ 0;\ Do\ not\ include\ sentence\ in\ the\ summary$$

Also, it is assumed that summary sentences represent the most important content of the document.

Abstractive Summarization:

For the task of performing abstractive summarisation, authors have used standard encoder-decoder framework. The encoder is the pretrained BERTSUM and the decoder is a 6-layered Transformer which is initialized randomly. As the encoder is pre-trained while the decoder is trained from scratch there is a mismatch between the encoder and the decoder. This can make fine-tuning unstable; for example, the encoder might overfit the data while the decoder underfits, or vice versa. To overcome this challenge, authors have used a new fine-tuning schedule which separates the optimizers of the encoder and the decoder.

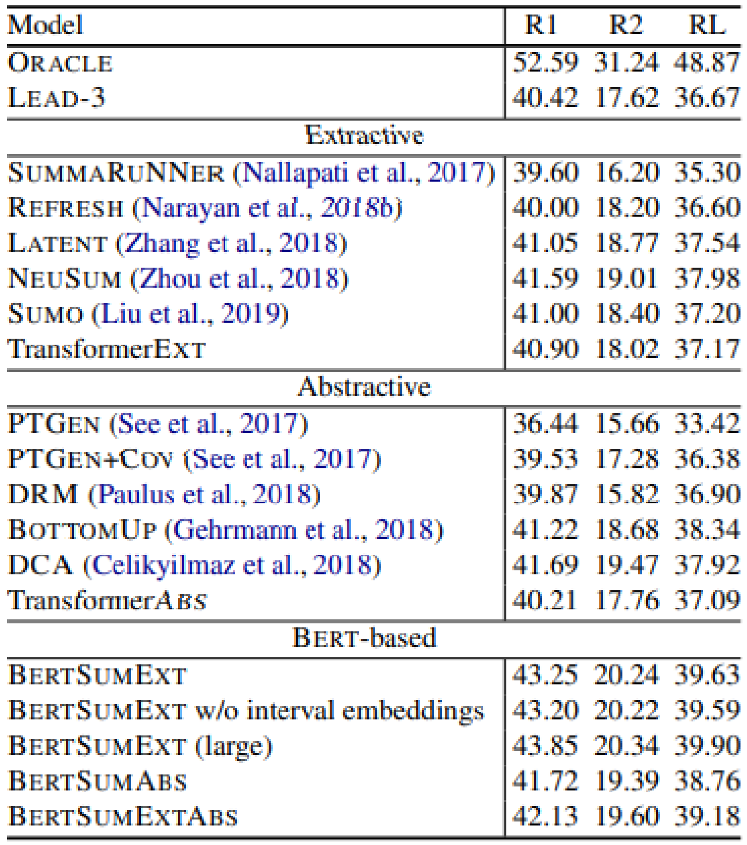
Also, a two-stage fine-tuning is proposed in which the encoder is first fine-tuned on the extractive summarization task and then fine-tuned on the abstractive summarization task. This two stage approach is beneficial because we can take advantage of the information shared between the tasks without changing the architecture. The default abstractive model is named as BertSumAbs and the two-stage fine-tuned model is named as BertSumExtAbs.

Evaluation Metric:

The evaluation metric used in this paper is ROUGE and more specifically ROUGE-1 (R1), ROUGE-2 (R2) and ROUGE-L.

Experiments And Results:

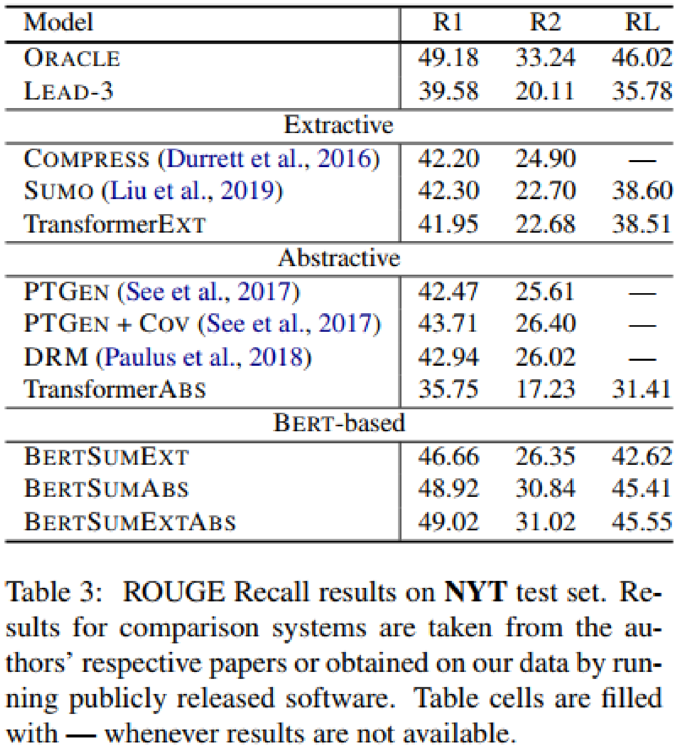
- **Results on CNN/Daily mail dataset:**



Model	R1	R2	RL
Oracle	52.59	31.24	48.87
Lead-3	40.42	17.62	36.67
Extractive			
SummaRuNNer (Nallapati et al., 2017)	39.60	16.20	35.30
Refresh (Narayan et al., 2018b)	40.00	18.20	36.60
Latent (Zhang et al., 2018)	41.05	18.77	37.54
NeuSum (Zhou et al., 2018)	41.59	19.01	37.98
Sumo (Liu et al., 2019)	41.00	18.40	37.20
TransformerExt	40.90	18.02	37.17
Abstractive			
PTGen (See et al., 2017)	36.44	15.66	33.42
PTGen+Cov (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
TransformerAbs	40.21	17.76	37.09
Bert-based			
BertSumExt	43.25	20.24	39.63
BertSumExt w/o interval embeddings	43.20	20.22	39.59
BertSumExt (large)	43.85	20.34	39.90
BertSumAbs	41.72	19.39	38.76
BertSumExtAbs	42.13	19.60	39.18

Table 2: ROUGE F1 results on **CNN/DailyMail** test set (R1 and R2 are shorthands for unigram and bigram overlap; RL is the longest common subsequence). Results for comparison systems are taken from the authors' respective papers or obtained on our data by running publicly released software.

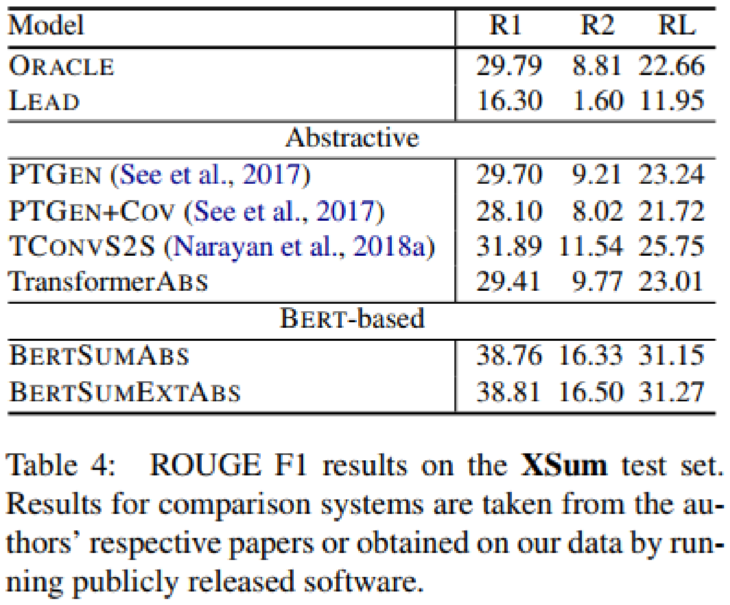
- **Results on NYT dataset:**



Model	R1	R2	RL
Oracle	49.18	33.24	46.02
Lead-3	39.58	20.11	35.78
Extractive			
Compress (Durrett et al., 2016)	42.20	24.90	—
Sumo (Liu et al., 2019)	42.30	22.70	38.60
TransformerExt	41.95	22.68	38.51
Abstractive			
PTGen (See et al., 2017)	42.47	25.61	—
PTGen + Cov (See et al., 2017)	43.71	26.40	—
DRM (Paulus et al., 2018)	42.94	26.02	—
TransformerAbs	35.75	17.23	31.41
Bert-based			
BertSumExt	46.66	26.35	42.62
BertSumAbs	48.92	30.84	45.41
BertSumExtAbs	49.02	31.02	45.55

Table 3: ROUGE Recall results on **NYT** test set. Results for comparison systems are taken from the authors' respective papers or obtained on our data by running publicly released software. Table cells are filled with — whenever results are not available.

- **Results on XSum dataset:**



Model	R1	R2	RL
Oracle	29.79	8.81	22.66
Lead	16.30	1.60	11.95
Abstractive			
PTGen (See et al., 2017)	29.70	9.21	23.24
PTGen+Cov (See et al., 2017)	28.10	8.02	21.72
TConvS2S (Narayan et al., 2018a)	31.89	11.54	25.75
TransformerAbs	29.41	9.77	23.01
Bert-based			
BertSumAbs	38.76	16.33	31.15
BertSumExtAbs	38.81	16.50	31.27

Table 4: ROUGE F1 results on the **XSum** test set. Results for comparison systems are taken from the authors' respective papers or obtained on our data by running publicly released software.

Conclusion:

In this paper, the author showcased how pretrained BERT can be usefully applied in text summarization and introduced a novel document-level encoder and proposed a general framework for both abstractive and extractive summarization. Experimental results across three datasets show that the model achieves state-of-the-art results across the board under automatic and human-based evaluation protocols. Although the author mainly focused on document encoding for summarization, in the future, we would like to take advantage of the capabilities of BERT for language generation.