Fine-tune BERT for Extractive Summarization

Yang Liu
Institute for Language, Cognition and Computation
School of Informatics, University of Edinburgh
10 Crichton Street, Edinburgh EH8 9AB
yang.liu2@ed.ac.uk

Abstract

BERT (Devlin et al., 2018), a pre-trained Transformer (Vaswani et al., 2017) model, has achieved ground-breaking performance on multiple NLP tasks. In this paper, we describe BERTSUM, a simple variant of BERT, for extractive summarization. Our system is the state of the art on the CNN/Dailymail dataset, outperforming the previous best-performed system by 1.65 on ROUGE-L. The codes to reproduce our results are available at https://github.com/nlpyang/BertSum

1 Introduction

Single-document summarization is the task of automatically generating a shorter version of a document while retaining its most important information. The task has received much attention in the natural language processing community due to its potential for various information access applications. Examples include tools which digest textual content (e.g., news, social media, reviews), answer questions, or provide recommendations.

The task is often divided into two paradigms, *abstractive* summarization and *extractive* summarization. In abstractive summarization, target summaries contains words or phrases that were not in the original text and usually require various text rewriting operations to generate, while extractive approaches form summaries by copying and concatenating the most important spans (usually sentences) in a document. In this paper, we focus on extractive summarization.

Although many neural models have been proposed for extractive summarization recently (Cheng and Lapata, 2016; Nallapati et al., 2017; Narayan et al., 2018; Dong et al., 2018; Zhang et al., 2018; Zhou et al., 2018), the improvement on automatic metrics like ROUGE has reached a bottleneck due to the complexity of the task. In this paper, we argue that, BERT (Devlin et al., 2018), with its pre-training on a huge dataset and the powerful architecture for learning complex features, can further boost the performance of extractive summarization .

In this paper, we focus on designing different variants of using BERT on the extractive summarization task and showing their results on CNN/Dailymail and NYT datasets. We found that a flat architecture with inter-sentence Transformer layers performs the best, achieving the state-of-the-art results on this task.

2 Methodology

Let d denote a document containing several sentences $[sent_1, sent_2, \cdots, sent_m]$, where $sent_i$ is the i-th sentence in the document. Extractive summarization can be defined as the task of assigning a label $y_i \in \{0, 1\}$ to each $sent_i$, indicating whether the sentence should be included in the summary. It is assumed that summary sentences represent the most important content of the document.

2.1 Extractive Summarization with BERT

To use BERT for extractive summarization, we require it to output the representation for each sentence. However, since BERT is trained as a masked-language model, the output vectors are grounded to tokens instead of sentences. Meanwhile, although BERT has segmentation embeddings for indicating different sentences, it only has two labels (sentence A or sentence B), instead of multiple sentences as in extractive summarization. Therefore, we modify the input sequence and embeddings of BERT to make it possible for extracting summaries.

Encoding Multiple Sentences As illustrated in Figure 1, we insert a [CLS] token before each sen-

[†]Please see https://arxiv.org/abs/1908.08345 for the full and most current version of this paper

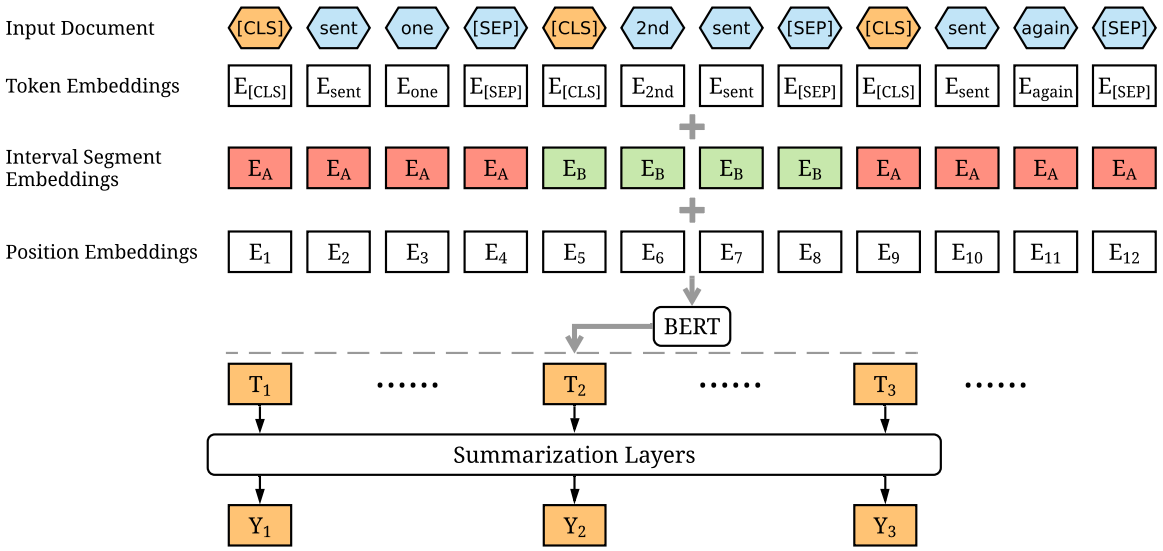

Figure 1: The overview architecture of the BERTSUM model.

tence and a [SEP] token after each sentence. In vanilla BERT, The [CLS] is used as a symbol to aggregate features from one sentence or a pair of sentences. We modify the model by using multiple [CLS] symbols to get features for sentences ascending the symbol.

Interval Segment Embeddings We use interval segment embeddings to distinguish multiple sentences within a document. For $sent_i$ we will assign a segment embedding E_A or E_B conditioned on i is odd or even. For example, for $[sent_1, sent_2, sent_3, sent_4, sent_5]$ we will assign $[E_A, E_B, E_A, E_B, E_A]$.

The vector T_i which is the vector of the i-th [CLS] symbol from the top BERT layer will be used as the representation for $sent_i$.

2.2 Fine-tuning with Summarization Layers

After obtaining the sentence vectors from BERT, we build several summarization-specific layers stacked on top of the BERT outputs, to capture document-level features for extracting summaries. For each sentence $sent_i$, we will calculate the final predicted score $\hat{Y}_i$. The loss of the whole model is the Binary Classification Entropy of $\hat{Y}_i$ against gold label Y_i. These summarization layers are jointly fine-tuned with BERT.

Simple Classifier Like in the original BERT paper, the Simple Classifier only adds a linear layer on the BERT outputs and use a sigmoid function to get the predicted score:

$$\hat{Y}_i = \sigma(W_o T_i + b_o) \quad (1)$$

where σ is the Sigmoid function.

Inter-sentence Transformer Instead of a simple sigmoid classifier, Inter-sentence Transformer applies more Transformer layers only on sentence representations, extracting document-level features focusing on summarization tasks from the BERT outputs:

$$\tilde{h}^l = \mathrm{LN}(h^{l-1} + \mathrm{MHAtt}(h^{l-1})) \quad (2)$$

$$h^l = \mathrm{LN}(\tilde{h}^l + \mathrm{FFN}(\tilde{h}^l)) \quad (3)$$

where $h^0 = \mathrm{PosEmb}(T)$ and T are the sentence vectors output by BERT, PosEmb is the function of adding positional embeddings (indicating the position of each sentence) to T; LN is the layer normalization operation (Ba et al., 2016); MHAtt is the multi-head attention operation (Vaswani et al., 2017); the superscript l indicates the depth of the stacked layer.

The final output layer is still a sigmoid classifier:

$$\hat{Y}_i = \sigma(W_o h_i^L + b_o) \quad (4)$$

where h^L is the vector for $sent_i$ from the top layer (the L-th layer) of the Transformer. In experiments, we implemented Transformers with $L = 1, 2, 3$ and found Transformer with 2 layers performs the best.

Recurrent Neural Network Although the Transformer model achieved great results on several tasks, there are evidence that Recurrent Neural Networks still have their advantages, especially when combining with techniques in Transformer (Chen et al., 2018). Therefore, we apply an LSTM layer over the BERT outputs to learn summarization-specific features.

To stabilize the training, pergate layer normalization (Ba et al., 2016) is applied within each LSTM cell. At time step i, the input to the LSTM layer is the BERT output T_i, and the output is calculated as:

$$\begin{pmatrix} F_i \\ I_i \\ O_i \\ G_i \end{pmatrix} = \mathrm{LN}_h(W_h h_{i-1}) + \mathrm{LN}_x(W_x T_i) \quad (5)$$

$$C_i = \sigma(F_i) \odot C_{i-1} + \sigma(I_i) \odot \tanh(G_{i-1}) \quad (6)$$

$$h_i = \sigma(O_t) \odot \tanh(\mathrm{LN}_c(C_t)) \quad (7)$$

where F_i, I_i, O_i are forget gates, input gates, output gates; G_i is the hidden vector and C_i is the memory vector; h_i is the output vector; $\mathrm{LN}_h, \mathrm{LN}_x, \mathrm{LN}_c$ are there difference layer normalization operations; Bias terms are not shown.

The final output layer is also a sigmoid classifier:

$$\hat{Y}_i = \sigma(W_o h_i + b_o) \quad (8)$$

3 Experiments

In this section we present our implementation, describe the summarization datasets and our evaluation protocol, and analyze our results.

3.1 Implementation Details

We use PyTorch, OpenNMT (Klein et al., 2017) and the 'bert-base-uncased'* version of BERT to implement the model. BERT and summarization layers are jointly fine-tuned. Adam with $\beta_1 = 0.9$, $\beta_2 = 0.999$ is used for fine-tuning. Learning rate schedule is following (Vaswani et al., 2017) with warming-up on first 10,000 steps:

$$lr = 2e^{-3} \cdot min(step^{-0.5}, step \cdot warmup^{-1.5})$$

All models are trained for 50,000 steps on 3 GPUs (GTX 1080 Ti) with gradient accumulation per two steps, which makes the batch size approximately equal to 36. Model checkpoints are saved and evaluated on the validation set every 1,000 steps. We select the top-3 checkpoints based on their evaluation losses on the validations set, and report the averaged results on the test set.

When predicting summaries for a new document, we first use the models to obtain the score for each sentence. We then rank these sentences by the scores from higher to lower, and select the top-3 sentences as the summary.

Trigram Blocking During the predicting process, Trigram Blocking is used to reduce redundancy. Given selected summary S and a candidate sentence c, we will skip c is there exists a trigram overlapping between c and S. This is similar to the Maximal Marginal Relevance (MMR) (Carbonell and Goldstein, 1998) but much simpler.

3.2 Summarization Datasets

We evaluated on two benchmark datasets, namely the CNN/DailyMail news highlights dataset (Hermann et al., 2015) and the New York Times Annotated Corpus (NYT; Sandhaus 2008). The CNN/DailyMail dataset contains news articles and associated highlights, i.e., a few bullet points giving a brief overview of the article. We used the standard splits of Hermann et al. (2015) for training, validation, and testing (90,266/1,220/1,093 CNN documents and 196,961/12,148/10,397 DailyMail documents). We did not anonymize entities. We first split sentences by CoreNLP and pre-process the dataset following methods in See et al. (2017).

The NYT dataset contains 110,540 articles with abstractive summaries. Following Durrett et al. (2016), we split these into 100,834 training and 9,706 test examples, based on date of publication (test is all articles published on January 1, 2007 or later). We took 4,000 examples from the training set as the validation set. We also followed their filtering procedure, documents with summaries that are shorter than 50 words were removed from the raw dataset. The filtered test set (NYT50) includes 3,452 test examples. We first split sentences by CoreNLP and pre-process the dataset following methods in Durrett et al. (2016).

Both datasets contain abstractive gold summaries, which are not readily suited to training extractive summarization models. A greedy algorithm was used to generate an oracle summary for

*https://github.com/huggingface/pytorch-pretrained-BERT

Model	ROUGE-1	ROUGE-2	ROUGE-L
PGN*	39.53	17.28	37.98
DCA*	41.69	19.47	37.92
LEAD	40.42	17.62	36.67
ORACLE	52.59	31.24	48.87
REFRESH*	41.0	18.8	37.7
NEUSUM*	41.59	19.01	37.98
Transformer	40.90	18.02	37.17
BERTSUM+Classifier	43.23	20.22	39.60
BERTSUM+Transformer	**43.25**	**20.24**	**39.63**
BERTSUM+LSTM	43.22	20.17	39.59

Table 1: Test set results on the CNN/DailyMail dataset using ROUGE F_1. Results with $*$ mark are taken from the corresponding papers.

each document. The algorithm greedily select sentences which can maximize the ROUGE scores as the oracle sentences. We assigned label 1 to sentences selected in the oracle summary and 0 otherwise.

4 Experimental Results

The experimental results on CNN/Dailymail datasets are shown in Table 1. For comparison, we implement a non-pretrained Transformer baseline which uses the same architecture as BERT, but with smaller parameters. It is randomly initialized and only trained on the summarization task. The Transformer baseline has 6 layers, the hidden size is 512 and the feed-forward filter size is 2048. The model is trained with same settings following Vaswani et al. (2017). We also compare our model with several previously proposed systems.

- LEAD is an extractive baseline which uses the first-3 sentences of the document as a summary.
- REFRESH (Narayan et al., 2018) is an extractive summarization system trained by globally optimizing the ROUGE metric with reinforcement learning.
- NEUSUM (Zhou et al., 2018) is the state-of-the-art extractive system that jontly score and select sentences.
- PGN (See et al., 2017), is the Pointer Generator Network, an abstractive summarization system based on an encoder-decoder architecture.
- DCA (Celikyilmaz et al., 2018) is the Deep Communicating Agents, a state-of-the-art abstractive summarization system with multiple agents to represent the document as well as hierarchical attention mechanism over the agents for decoding.

As illustrated in the table, all BERT-based models outperformed previous state-of-the-art models by a large margin. BERTSUM with Transformer achieved the best performance on all three metrics. The BERTSUM with LSTM model does not have an obvious influence on the summarization performance compared to the Classifier model.

Ablation studies are conducted to show the contribution of different components of BERTSUM. The results are shown in in Table 2. Interval segments increase the performance of base model. Trigram blocking is able to greatly improve the summarization results. This is consistent to previous conclusions that a sequential extractive decoder is helpful to generate more informative summaries. However, here we use the trigram blocking as a simple but robust alternative.

Model	R-1	R-2	R-L
BERTSUM+Classifier	43.23	20.22	39.60
-interval segments	43.21	20.17	39.57
-trigram blocking	42.57	19.96	39.04

Table 2: Results of ablation studies of BERTSUM on CNN/Dailymail test set using ROUGE F_1 (R-1 and R-2 are shorthands for unigram and bigram overlap, R-L is the longest common subsequence).

The experimental results on NYT datasets are shown in Table 3. Different from CNN/Dailymail, we use the limited-length recall evaluation, fol-

lowing Durrett et al. (2016). We truncate the predicted summaries to the lengths of the gold summaries and evaluate summarization quality with ROUGE Recall. Compared baselines are (1) First-k words, which is a simple baseline by extracting first k words of the input article; (2) Full is the best-performed extractive model in Durrett et al. (2016); (3) Deep Reinforced (Paulus et al., 2018) is an abstractive model, using reinforce learning and encoder-decoder structure. The BERTSUM+Classifier can achieve the state-of-the-art results on this dataset.

Model	R-1	R-2	R-L
First-k words	39.58	20.11	35.78
Full*	42.2	24.9	-
Deep Reinforced*	42.94	26.02	-
BERTSUM+Classifier	**46.66**	**26.35**	**42.62**

Table 3: Test set results on the NYT50 dataset using ROUGE Recall. The predicted summary are truncated to the length of the gold-standard summary. Results with $*$ mark are taken from the corresponding papers.

5 Conclusion

In this paper, we explored how to use BERT for extractive summarization. We proposed the BERTSUM model and tried several summarization layers can be applied with BERT. We did experiments on two large-scale datasets and found the BERTSUM with inter-sentence Transformer layers can achieve the best performance.

References

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

Jaime G Carbonell and Jade Goldstein. 1998. The use of mmr and diversity-based reranking for reodering documents and producing summaries.

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

Mia Xu Chen, Orhan Firat, Ankur Bapna, Melvin Johnson, Wolfgang Macherey, George Foster, Llion Jones, Niki Parmar, Mike Schuster, Zhifeng Chen, et al. 2018. The best of both worlds: Combining recent advances in neural machine translation. In *Proceedings of the ACL Conference*.

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

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding.

Yue Dong, Yikang Shen, Eric Crawford, Herke van Hoof, and Jackie Chi Kit Cheung. 2018. Banditsum: Extractive summarization as a contextual bandit. In *Proceedings of the EMNLP Conference*.

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

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pages 1693–1701.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M Rush. 2017. Opennmt: Open-source toolkit for neural machine translation. In *arXiv preprint arXiv:1701.02810*.

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

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

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

Evan Sandhaus. 2008. The New York Times Annotated Corpus. *Linguistic Data Consortium, Philadelphia*, 6(12).

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

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 *Advances in Neural Information Processing Systems*, pages 5998–6008.

Xingxing Zhang, Mirella Lapata, Furu Wei, and Ming Zhou. 2018. Neural latent extractive document summarization. In *Proceedings of the EMNLP Conference*.

Qingyu Zhou, Nan Yang, Furu Wei, Shaohan Huang, Ming Zhou, and Tiejun Zhao. 2018. Neural document summarization by jointly learning to score and select sentences. In *Proceedings of the ACL Conference*.