Text Summarization with Pretrained Encoders

Yang Liu and **Mirella Lapata**
Institute for Language, Cognition and Computation
School of Informatics, University of Edinburgh
yang.liu2@ed.ac.uk, mlap@inf.ed.ac.uk

Abstract

Bidirectional Encoder Representations from Transformers (BERT; Devlin et al. 2019) represents the latest incarnation of pretrained language models which have recently advanced a wide range of natural language processing tasks. In this paper, we showcase how BERT can be usefully applied in text summarization and propose a general framework for both extractive and abstractive models. We introduce a novel document-level encoder based on BERT which is able to express the semantics of a document and obtain representations for its sentences. Our extractive model is built on top of this encoder by stacking several inter-sentence Transformer layers. For abstractive summarization, we propose a new fine-tuning schedule which adopts different optimizers for the encoder and the decoder as a means of alleviating the mismatch between the two (the former is pretrained while the latter is not). We also demonstrate that a two-staged fine-tuning approach can further boost the quality of the generated summaries. Experiments on three datasets show that our model achieves state-of-the-art results across the board in both extractive and abstractive settings.[1]

1 Introduction

Language model pretraining has advanced the state of the art in many NLP tasks ranging from sentiment analysis, to question answering, natural language inference, named entity recognition, and textual similarity. State-of-the-art pretrained models include ELMo (Peters et al., 2018), GPT (Radford et al., 2018), and more recently Bidirectional Encoder Representations from Transformers (BERT; Devlin et al. 2019). BERT combines both word and sentence representations in a single very large Transformer (Vaswani et al., 2017); it is pretrained on vast amounts of text, with an unsupervised objective of masked language modeling and next-sentence prediction and can be fine-tuned with various task-specific objectives.

In most cases, pretrained language models have been employed as encoders for sentence- and paragraph-level natural language understanding problems (Devlin et al., 2019) involving various classification tasks (e.g., predicting whether any two sentences are in an entailment relationship; or determining the completion of a sentence among four alternative sentences). In this paper, we examine the influence of language model pretraining on text summarization. Different from previous tasks, summarization requires wide-coverage natural language understanding going beyond the meaning of individual words and sentences. The aim is to condense a *document* into a shorter version while preserving most of its meaning. Furthermore, under abstractive modeling formulations, the task requires language generation capabilities in order to create summaries containing novel words and phrases not featured in the source text, while extractive summarization is often defined as a binary classification task with labels indicating whether a text span (typically a sentence) should be included in the summary.

We explore the potential of BERT for text summarization under a general framework encompassing both extractive and abstractive modeling paradigms. We propose a novel document-level encoder based on BERT which is able to encode a document and obtain representations for its sentences. Our extractive model is built on top of this encoder by stacking several inter-sentence Transformer layers to capture document-level features for extracting sentences. Our abstractive model adopts an encoder-decoder architecture, combining the same pretrained BERT encoder with a randomly-initialized Transformer de-

[1] Our code is available at https://github.com/nlpyang/PreSumm.

coder (Vaswani et al., 2017). We design a new training schedule which separates the optimizers of the encoder and the decoder in order to accommodate the fact that the former is pretrained while the latter must be trained from scratch. Finally, motivated by previous work showing that the combination of extractive and abstractive objectives can help generate better summaries (Gehrmann et al., 2018), we present a two-stage approach where the encoder is fine-tuned twice, first with an extractive objective and subsequently on the abstractive summarization task.

We evaluate the proposed approach on three single-document news summarization datasets representative of different writing conventions (e.g., important information is concentrated at the beginning of the document or distributed more evenly throughout) and summary styles (e.g., verbose vs. more telegraphic; extractive vs. abstractive). Across datasets, we experimentally show that the proposed models achieve state-of-the-art results under both extractive and abstractive settings. Our contributions in this work are threefold: a) we highlight the importance of document encoding for the summarization task; a variety of recently proposed techniques aim to enhance summarization performance via copying mechanisms (Gu et al., 2016; See et al., 2017; Nallapati et al., 2017), reinforcement learning (Narayan et al., 2018b; Paulus et al., 2018; Dong et al., 2018), and multiple communicating encoders (Celikyilmaz et al., 2018). We achieve better results with a minimum-requirement model without using any of these mechanisms; b) we showcase ways to effectively employ pretrained language models in summarization under both extractive and abstractive settings; we would expect any improvements in model pretraining to translate in better summarization in the future; and c) the proposed models can be used as a stepping stone to further improve summarization performance as well as baselines against which new proposals are tested.

2 Background

2.1 Pretrained Language Models

Pretrained language models (Peters et al., 2018; Radford et al., 2018; Devlin et al., 2019; Dong et al., 2019; Zhang et al., 2019) have recently emerged as a key technology for achieving impressive gains in a wide variety of natural language tasks. These models extend the idea of word embeddings by learning contextual representations from large-scale corpora using a language modeling objective. Bidirectional Encoder Representations from Transformers (BERT; Devlin et al. 2019) is a new language representation model which is trained with a masked language modeling and a "next sentence prediction" task on a corpus of 3,300M words.

The general architecture of BERT is shown in the left part of Figure 1. Input text is first preprocessed by inserting two special tokens. [CLS] is appended to the beginning of the text; the output representation of this token is used to aggregate information from the whole sequence (e.g., for classification tasks). And token [SEP] is inserted after each sentence as an indicator of sentence boundaries. The modified text is then represented as a sequence of tokens $X = [w_1, w_2, \cdots, w_n]$. Each token w_i is assigned three kinds of embeddings: *token embeddings* indicate the meaning of each token, *segmentation embeddings* are used to discriminate between two sentences (e.g., during a sentence-pair classification task) and *position embeddings* indicate the position of each token within the text sequence. These three embeddings are summed to a single input vector x_i and fed to a bidirectional Transformer with multiple layers:

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

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

where $h^0 = x$ are the input vectors; LN is the layer normalization operation (Ba et al., 2016); MHAtt is the multi-head attention operation (Vaswani et al., 2017); superscript l indicates the depth of the stacked layer. On the top layer, BERT will generate an output vector t_i for each token with rich contextual information.

Pretrained language models are usually used to enhance performance in language understanding tasks. Very recently, there have been attempts to apply pretrained models to various generation problems (Edunov et al., 2019; Rothe et al., 2019). When fine-tuning for a specific task, unlike ELMo whose parameters are usually fixed, parameters in BERT are *jointly* fine-tuned with additional task-specific parameters.

2.2 Extractive Summarization

Extractive summarization systems create a summary by identifying (and subsequently concatenating) the most important sentences in a document. Neural models consider extractive sum-

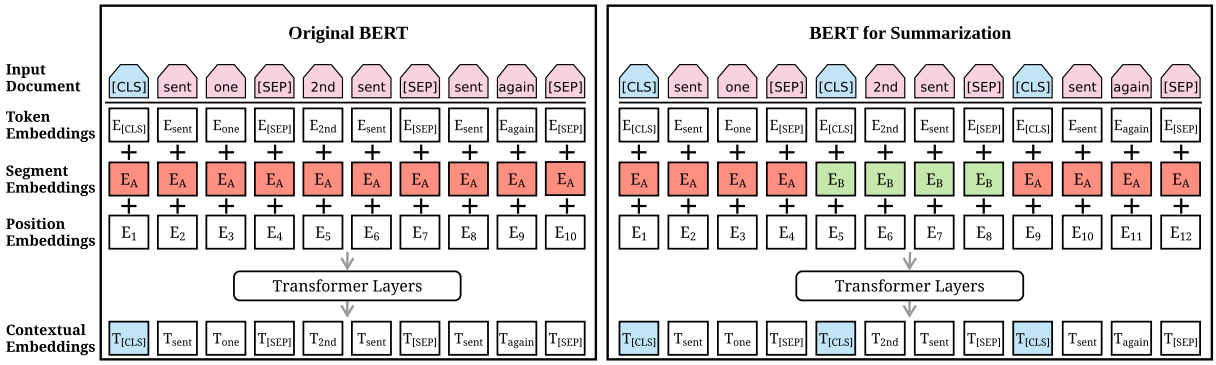


Figure 1: Architecture of the original BERT model (left) and BERTSUM (right). The sequence on top is the input document, followed by the summation of three kinds of embeddings for each token. The summed vectors are used as input embeddings to several bidirectional Transformer layers, generating contextual 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.

marization as a sentence classification problem: a neural encoder creates sentence representations and a classifier predicts which sentences should be selected as summaries. SUMMARUNNER (Nallapati et al., 2017) is one of the earliest neural approaches adopting an encoder based on Recurrent Neural Networks. REFRESH (Narayan et al., 2018b) is a reinforcement learning-based system trained by globally optimizing the ROUGE metric. More recent work achieves higher performance with more sophisticated model structures. LATENT (Zhang et al., 2018) frames extractive summarization as a latent variable inference problem; instead of maximizing the likelihood of "gold" standard labels, their latent model directly maximizes the likelihood of human summaries given selected sentences. SUMO (Liu et al., 2019) capitalizes on the notion of structured attention to induce a multi-root dependency tree representation of the document while predicting the output summary. NEUSUM (Zhou et al., 2018) scores and selects sentences jointly and represents the state of the art in extractive summarization.

2.3 Abstractive Summarization

Neural approaches to abstractive summarization conceptualize the task as a sequence-to-sequence problem, where an encoder maps a sequence of tokens in the source document $\boldsymbol{x} = [x_1, ..., x_n]$ to a sequence of continuous representations $\boldsymbol{z} = [z_1, ..., z_n]$, and a decoder then generates the target summary $\boldsymbol{y} = [y_1, ..., y_m]$ token-by-token, in an auto-regressive manner, hence modeling the conditional probability: $p(y_1, ..., y_m|x_1, ..., x_n)$.

Rush et al. (2015) and Nallapati et al. (2016) were among the first to apply the neural encoder-decoder architecture to text summarization. See et al. (2017) enhance this model with a pointer-generator network (PTGEN) which allows it to copy words from the source text, and a coverage mechanism (COV) which keeps track of words that have been summarized. Celikyilmaz et al. (2018) propose an abstractive system where multiple agents (encoders) represent the document together with a hierarchical attention mechanism (over the agents) for decoding. Their Deep Communicating Agents (DCA) model is trained end-to-end with reinforcement learning. Paulus et al. (2018) also present a deep reinforced model (DRM) for abstractive summarization which handles the coverage problem with an intra-attention mechanism where the decoder attends over previously generated words. Gehrmann et al. (2018) follow a bottom-up approach (BOTTOMUP); a content selector first determines which phrases in the source document should be part of the summary, and a copy mechanism is applied only to preselected phrases during decoding. Narayan et al. (2018a) propose an abstractive model which is particularly suited to extreme summarization (i.e., single sentence summaries), based on convolutional neural networks and additionally conditioned on topic distributions (TCONVS2S).

3 Fine-tuning BERT for Summarization

3.1 Summarization Encoder

Although BERT has been used to fine-tune various NLP tasks, its application to summarization

is not as straightforward. Since BERT is trained as a masked-language model, the output vectors are grounded to tokens instead of sentences, while in extractive summarization, most models manipulate sentence-level representations. Although segmentation embeddings represent different sentences in BERT, they only apply to sentence-pair inputs, while in summarization we must encode and manipulate multi-sentential inputs. Figure 1 illustrates our proposed BERT architecture for SUMmarization (which we call BERTSUM).

In order to represent *individual* sentences, we insert external [CLS] tokens at the start of each sentence, and each [CLS] symbol collects features for the sentence preceding it. We also use *interval segment* embeddings to distinguish multiple sentences within a document. For $sent_i$ we assign segment embedding E_A or E_B depending on whether i is odd or even. For example, for document $[sent_1, sent_2, sent_3, sent_4, sent_5]$, we would assign embeddings $[E_A, E_B, E_A, E_B, E_A]$. 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.

3.2 Extractive Summarization

Let d denote a document containing 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.

With BERTSUM, vector t_i which is the vector of the i-th [CLS] symbol from the top layer can be used as the representation for $sent_i$. Several inter-sentence Transformer layers are then stacked on top of BERT outputs, to capture document-level features for extracting summaries:

$$\tilde{h}^l = \mathrm{LN}(h^{l-1} + \mathrm{MHAtt}(h^{l-1})) \quad (3)$$

$$h^l = \mathrm{LN}(\tilde{h}^l + \mathrm{FFN}(\tilde{h}^l)) \quad (4)$$

where $h^0 = \mathrm{PosEmb}(T)$; T denotes the sentence vectors output by BERTSUM, and function PosEmb adds sinusoid positional embeddings (Vaswani et al., 2017) to T, indicating the position of each sentence.

The final output layer is a sigmoid classifier:

$$\hat{y}_i = \sigma(W_o h_i^L + b_o) \quad (5)$$

where h_i^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 that a Transformer with $L = 2$ performed best. We name this model BERTSUMEXT.

The loss of the model is the binary classification entropy of prediction $\hat{y}_i$ against gold label y_i. Inter-sentence Transformer layers are jointly fine-tuned with BERTSUM. We use the Adam optimizer with $\beta_1 = 0.9$, and $\beta_2 = 0.999$). Our learning rate schedule follows (Vaswani et al., 2017) with warming-up ($\mathrm{warmup} = 10,000$):

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

3.3 Abstractive Summarization

We use a standard encoder-decoder framework for abstractive summarization (See et al., 2017). The encoder is the pretrained BERTSUM and the decoder is a 6-layered Transformer initialized randomly. It is conceivable that there is a mismatch between the encoder and the decoder, since the former is pretrained while the latter must be trained from scratch. This can make fine-tuning unstable; for example, the encoder might overfit the data while the decoder underfits, or vice versa. To circumvent this, we design a new fine-tuning schedule which separates the optimizers of the encoder and the decoder.

We use two Adam optimizers with $\beta_1 = 0.9$ and $\beta_2 = 0.999$ for the encoder and the decoder, respectively, each with different warmup-steps and learning rates:

$$lr_{\mathcal{E}} = \tilde{lr}_{\mathcal{E}} \cdot \min(step^{-0.5}, \mathrm{step} \cdot \mathrm{warmup}_{\mathcal{E}}^{-1.5}) \quad (6)$$

$$lr_{\mathcal{D}} = \tilde{lr}_{\mathcal{D}} \cdot \min(step^{-0.5}, \mathrm{step} \cdot \mathrm{warmup}_{\mathcal{D}}^{-1.5}) \quad (7)$$

where $\tilde{lr}_{\mathcal{E}} = 2e^{-3}$, and $\mathrm{warmup}_{\mathcal{E}} = 20,000$ for the encoder and $\tilde{lr}_{\mathcal{D}} = 0.1$, and $\mathrm{warmup}_{\mathcal{D}} = 10,000$ for the decoder. This is based on the assumption that the pretrained encoder should be fine-tuned with a smaller learning rate and smoother decay (so that the encoder can be trained with more accurate gradients when the decoder is becoming stable).

Datasets	# docs (train/val/test)	avg. doc length		avg. summary length		% novel bi-grams
		words	sentences	words	sentences	in gold summary
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.

In addition, we propose a two-stage fine-tuning approach, where we first fine-tune the encoder on the extractive summarization task (Section 3.2) and then fine-tune it on the abstractive summarization task (Section 3.3). Previous work (Gehrmann et al., 2018; Li et al., 2018) suggests that using extractive objectives can boost the performance of abstractive summarization. Also notice that this two-stage approach is conceptually very simple, the model can take advantage of information shared between these two tasks, without fundamentally changing its architecture. We name the default abstractive model BERTSUMABS and the two-stage fine-tuned model BERTSUMEXTABS.

4 Experimental Setup

In this section, we describe the summarization datasets used in our experiments and discuss various implementation details.

4.1 Summarization Datasets

We evaluated our model on three benchmark datasets, namely the CNN/DailyMail news highlights dataset (Hermann et al., 2015), the New York Times Annotated Corpus (NYT; Sandhaus 2008), and XSum (Narayan et al., 2018a). These datasets represent different summary styles ranging from highlights to very brief one sentence summaries. The summaries also vary with respect to the type of rewriting operations they exemplify (e.g., some showcase more cut and paste operations while others are genuinely abstractive). Table 1 presents statistics on these datasets (test set); example (gold-standard) summaries are provided in the supplementary material.

CNN/DailyMail 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 with the Stanford CoreNLP toolkit (Manning et al., 2014) and pre-processed the dataset following See et al. (2017). Input documents were truncated to 512 tokens.

NYT contains 110,540 articles with abstractive summaries. Following Durrett et al. (2016), we split these into 100,834/9,706 training/test examples, based on the date of publication (the test set contains all articles published from January 1, 2007 onward). We used 4,000 examples from the training as validation set. We also followed their filtering procedure, documents with summaries less than 50 words were removed from the dataset. The filtered test set (NYT50) includes 3,452 examples. Sentences were split with the Stanford CoreNLP toolkit (Manning et al., 2014) and pre-processed following Durrett et al. (2016). Input documents were truncated to 800 tokens.

XSum contains 226,711 news articles accompanied with a one-sentence summary, answering the question "What is this article about?". We used the splits of Narayan et al. (2018a) for training, validation, and testing (204,045/11,332/11,334) and followed the pre-processing introduced in their work. Input documents were truncated to 512 tokens.

Aside from various statistics on the three datasets, Table 1 also reports the proportion of novel bi-grams in gold summaries as a measure of their abstractiveness. We would expect models with extractive biases to perform better on datasets with (mostly) extractive summaries, and abstractive models to perform more rewrite operations on datasets with abstractive summaries. CNN/DailyMail and NYT are somewhat extractive, while XSum is highly abstractive.

4.2 Implementation Details

For both extractive and abstractive settings, we used PyTorch, OpenNMT (Klein et al., 2017) and the 'bert-base-uncased'[2] version of BERT to implement BERTSUM. Both source and target texts

[2] https://git.io/fhbJQ

were tokenized with BERT's subwords tokenizer.

Extractive Summarization All extractive models were trained for 50,000 steps on 3 GPUs (GTX 1080 Ti) with gradient accumulation every two steps. Model checkpoints were saved and evaluated on the validation set every 1,000 steps. We selected the top-3 checkpoints based on the evaluation loss on the validation set, and report the averaged results on the test set. We used a greedy algorithm similar to Nallapati et al. (2017) to obtain an oracle summary for each document to train extractive models. The algorithm generates an oracle consisting of multiple sentences which maximize the ROUGE-2 score against the gold summary.

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

During sentence selection we use *Trigram Blocking* to reduce redundancy (Paulus et al., 2018). Given summary S and candidate sentence c, we skip c if there exists a trigram overlapping between c and S. The intuition is similar to Maximal Marginal Relevance (MMR; Carbonell and Goldstein 1998); we wish to minimize the similarity between the sentence being considered and sentences which have been already selected as part of the summary.

Abstractive Summarization In all abstractive models, we applied dropout (with probability 0.1) before all linear layers; label smoothing (Szegedy et al., 2016) with smoothing factor 0.1 was also used. Our Transformer decoder has 768 hidden units and the hidden size for all feed-forward layers is 2,048. All models were trained for 200,000 steps on 4 GPUs (GTX 1080 Ti) with gradient accumulation every five steps. Model checkpoints were saved and evaluated on the validation set every 2,500 steps. We selected the top-3 checkpoints based on their evaluation loss on the validation set, and report the averaged results on the test set.

During decoding we used beam search (size 5), and tuned the α for the length penalty (Wu et al., 2016) between 0.6 and 1 on the validation set; we decode until an end-of-sequence token is emitted and repeated trigrams are blocked (Paulus et al., 2018). It is worth noting that our decoder applies neither a copy nor a coverage mechanism (See et al., 2017), despite their popularity in abstractive summarization. This is mainly because we focus on building a minimum-requirements model and these mechanisms may introduce additional hyper-parameters to tune. Thanks to the subwords tokenizer, we also rarely observe issues with out-of-vocabulary words in the output; moreover, trigram-blocking produces diverse summaries managing to reduce repetitions.

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.

5 Results

5.1 Automatic Evaluation

We evaluated summarization quality automatically using ROUGE (Lin, 2004). We report unigram and bigram overlap (ROUGE-1 and ROUGE-2) as a means of assessing informativeness and the longest common subsequence (ROUGE-L) as a means of assessing fluency.

Table 2 summarizes our results on the CNN/DailyMail dataset. The first block in the table includes the results of an extractive ORACLE system as an upper bound. We also present the LEAD-3 baseline (which simply selects the first three sentences in a document).

The second block in the table includes various extractive models trained on the CNN/DailyMail dataset (see Section 2.2 for an overview). For

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.

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.

comparison to our own model, we also implemented a non-pretrained Transformer baseline (TransformerEXT) which uses the same architecture as BERTSUMEXT, but with fewer parameters. It is randomly initialized and only trained on the summarization task. TransformerEXT has 6 layers, the hidden size is 512, and the feed-forward filter size is 2,048. The model was trained with same settings as in Vaswani et al. (2017).

The third block in Table 2 highlights the performance of several abstractive models on the CNN/DailyMail dataset (see Section 2.3 for an overview). We also include an abstractive Transformer baseline (TransformerABS) which has the same decoder as our abstractive BERTSUM models; the encoder is a 6-layer Transformer with 768 hidden size and 2,048 feed-forward filter size.

The fourth block reports results with fine-tuned BERT models: BERTSUMEXT and its two variants (one without interval embeddings, and one with the large version of BERT), BERTSUMABS, and BERTSUMEXTABS. BERT-based models outperform the LEAD-3 baseline which is not a strawman; on the CNN/DailyMail corpus it is indeed superior to several extractive (Nallapati et al., 2017; Narayan et al., 2018b; Zhou et al., 2018) and abstractive models (See et al., 2017). BERT models collectively outperform all previously proposed extractive and abstractive systems, only falling behind the ORACLE upper bound. Among BERT variants, BERTSUMEXT performs best which is not entirely surprising; CNN/DailyMail summaries are somewhat extractive and even abstractive models are prone to copying sentences from the source document when trained on this dataset (See et al., 2017). Perhaps unsurprisingly we observe that larger versions of BERT lead to performance improvements and that interval embeddings bring only slight gains.

Table 3 presents results on the NYT dataset. Following the evaluation protocol in Durrett et al. (2016), we use limited-length ROUGE Recall, where predicted summaries are truncated to the length of the gold summaries. Again, we report the performance of the ORACLE upper bound and LEAD-3 baseline. The second block in the table contains previously proposed extractive models as well as our own Transformer baseline. COMPRESS (Durrett et al., 2016) is an ILP-based model which combines compression and anaphoricity constraints. The third block includes abstractive models from the literature, and our Transformer baseline. BERT-based models are shown in the fourth block. Again, we observe that they outperform previously proposed approaches. On this dataset, abstractive BERT models generally perform better compared to BERTSUMEXT, almost approaching ORACLE performance.

Table 4 summarizes our results on the XSum dataset. Recall that summaries in this dataset are highly abstractive (see Table 1) consisting of a single sentence conveying the gist of the document. Extractive models here perform poorly as corroborated by the low performance of the LEAD baseline (which simply selects the leading sentence from the document), and the ORACLE (which selects a single-best sentence in each document) in Table 4. As a result, we do not report results for extractive models on this dataset. The second

$\tilde{lr}_{\mathcal{E}}$ \ $\tilde{lr}_{\mathcal{D}}$	1	0.1	0.01	0.001
2e-2	50.69	9.33	10.13	19.26
2e-3	37.21	8.73	9.52	16.88

Table 5: Model perplexity (CNN/DailyMail; validation set) under different combinations of encoder and decoder learning rates.

block in Table 4 presents the results of various abstractive models taken from Narayan et al. (2018a) and also includes our own abstractive Transformer baseline. In the third block we show the results of our BERT summarizers which again are superior to all previously reported models (by a wide margin).

5.2 Model Analysis

Learning Rates Recall that our abstractive model uses separate optimizers for the encoder and decoder. In Table 5 we examine whether the combination of different learning rates ($\tilde{lr}_{\mathcal{E}}$ and $\tilde{lr}_{\mathcal{D}}$) is indeed beneficial. Specifically, we report model perplexity on the CNN/DailyMail validation set for varying encoder/decoder learning rates. We can see that the model performs best with $\tilde{lr}_{\mathcal{E}} = 2e-3$ and $\tilde{lr}_{\mathcal{D}} = 0.1$.

Position of Extracted Sentences In addition to the evaluation based on ROUGE, we also analyzed in more detail the summaries produced by our model. For the extractive setting, we looked at the position (in the source document) of the sentences which were selected to appear in the summary. Figure 2 shows the proportion of selected summary sentences which appear in the source document at positions 1, 2, and so on. The analysis was conducted on the CNN/DailyMail dataset for Oracle summaries, and those produced by BERTSUMEXT and the TransformerEXT. We can see that Oracle summary sentences are fairly smoothly distributed across documents, while summaries created by TransformerEXT mostly concentrate on the first document sentences. BERTSUMEXT outputs are more similar to Oracle summaries, indicating that with the pretrained encoder, the model relies less on shallow position features, and learns deeper document representations.

Novel N-grams We also analyzed the output of abstractive systems by calculating the proportion of novel n-grams that appear in the summaries but not in the source texts. The results are shown in Figure 3. In the CNN/DailyMail dataset, the proportion of novel n-grams in automatically generated summaries is much lower compared to reference summaries, but in XSum, this gap is much smaller. We also observe that on CNN/DailyMail, BERTEXTABS produces less novel n-ngrams than BERTABS, which is not surprising. BERTEXTABS is more biased towards selecting sentences from the source document since it is initially trained as an extractive model.

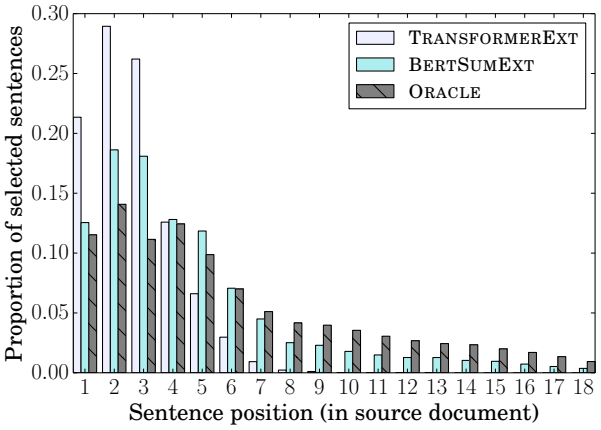


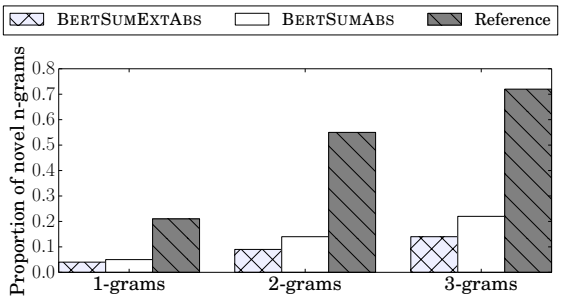
Figure 2: Proportion of extracted sentences according to their position in the original document.

The supplementary material includes examples of system output and additional ablation studies.

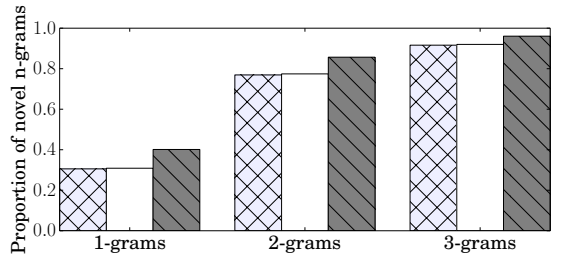
5.3 Human Evaluation

In addition to automatic evaluation, we also evaluated system output by eliciting human judgments. We report experiments following a question-answering (QA) paradigm (Clarke and Lapata, 2010; Narayan et al., 2018b) which quantifies the degree to which summarization models retain key information from the document. Under this paradigm, a set of questions is created based on the gold summary under the assumption that it highlights the most important document content. Participants are then asked to answer these questions by reading system summaries alone without access to the article. The more questions a system can answer, the better it is at summarizing the document as a whole.

Moreover, we also assessed the overall quality of the summaries produced by abstractive systems which due to their ability to rewrite content may produce disfluent or ungrammatical output. Specifically, we followed the Best-Worst Scaling (Kiritchenko and Mohammad, 2017) method where participants were presented with the output of two systems (and the original document) and



(a) CNN/DailyMail Dataset



(b) XSum dataset

Figure 3: Proportion of novel n-grams in model generated summaries.

Extractive	CNN/DM	NYT
LEAD	42.5†	36.2†
NEUSUM	42.2†	—
SUMO	41.7†	38.1†
Transformer	37.8†	32.5†
BERTSUM	58.9	41.9

Table 6: QA-based evaluation. Models with † are significantly different from BERTSUM (using a paired student t-test; $p < 0.05$). Table cells are filled with — whenever system output is not available.

asked to decide which one was better according to the criteria of *Informativeness*, *Fluency*, and *Succinctness*.

Both types of evaluation were conducted on the Amazon Mechanical Turk platform. For the CNN/DailyMail and NYT datasets we used the same documents (20 in total) and questions from previous work (Narayan et al., 2018b; Liu et al., 2019). For XSum, we randomly selected 20 documents (and their questions) from the release of Narayan et al. (2018a). We elicited 3 responses per HIT. With regard to QA evaluation, we adopted the scoring mechanism from Clarke and Lapata (2010); correct answers were marked with a score of one, partially correct answers with 0.5, and zero otherwise. For quality-based evaluation, the rating of each system was computed as the percentage of times it was chosen as better minus the times it was selected as worse. Ratings thus range from -1 (worst) to 1 (best).

	CNN/DM		NYT		XSum	
Abstractive	QA	Rank	QA	Rank	QA	Rank
LEAD	42.5†	—	36.2†	—	9.20†	—
PTGEN	33.3†	-0.24†	30.5†	-0.27†	23.7†	-0.36†
BOTTOMUP	40.6†	-0.16†	—	—	—	—
TCONVS2S	—	—	—	—	52.1	-0.20†
GOLD	—	0.22†	—	0.33†	—	0.38†
BERTSUM	56.1	0.17	41.8	-0.07	57.5	0.19

Table 7: QA-based and ranking-based evaluation. Models with † are significantly different from BERTSUM (using a paired student t-test; $p < 0.05$). Table cells are filled with — whenever system output is not available. GOLD is not used in QA setting, and LEAD is not used in Rank evaluation.

Results for extractive and abstractive systems are shown in Tables 6 and 7, respectively. We compared the best performing BERTSUM model in each setting (extractive or abstractive) against various state-of-the-art systems (whose output is publicly available), the LEAD baseline, and the GOLD standard as an upper bound. As shown in both tables participants overwhelmingly prefer the output of our model against comparison systems across datasets and evaluation paradigms. All differences between BERTSUM and comparison models are statistically significant ($p < 0.05$), with the exception of TCONVS2S (see Table 7; XSum) in the QA evaluation setting.

6 Conclusions

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

Acknowledgments

This research is supported by a Google PhD Fellowship to the first author. We gratefully acknowledge the support of the European Research Council (Lapata, award number 681760, "Translating Multiple Modalities into Text"). We would also like to thank Shashi Narayan for providing us with the XSum dataset.

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 reordering documents and producing summaries. In *Proceedings of the 21st Annual International ACL SIGIR Conference on Research and Development in Information Retrieval*, pages 335–336, Melbourne, Australia.

Asli Celikyilmaz, Antoine Bosselut, Xiaodong He, and Yejin Choi. 2018. Deep communicating agents for abstractive summarization. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1662–1675, New Orleans, Louisiana.

James Clarke and Mirella Lapata. 2010. Discourse constraints for document compression. *Computational Linguistics*, 36(3):411–441.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota.

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. *arXiv preprint arXiv:1905.03197*.

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 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3739–3748, Brussels, Belgium.

Greg Durrett, Taylor Berg-Kirkpatrick, and Dan Klein. 2016. Learning-based single-document summarization with compression and anaphoricity constraints. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1998–2008, Berlin, Germany.

Sergey Edunov, Alexei Baevski, and Michael Auli. 2019. Pre-trained language model representations for language generation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4052–4059, Minneapolis, Minnesota.

Sebastian Gehrmann, Yuntian Deng, and Alexander Rush. 2018. Bottom-up abstractive summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4098–4109, Brussels, Belgium.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O.K. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1631–1640, Berlin, Germany. Association for Computational Linguistics.

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.

Svetlana Kiritchenko and Saif Mohammad. 2017. Best-worst scaling more reliable than rating scales: A case study on sentiment intensity annotation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 465–470, Vancouver, Canada.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander Rush. 2017. OpenNMT: Open-source toolkit for neural machine translation. In *Proceedings of ACL 2017, System Demonstrations*, pages 67–72, Vancouver, Canada.

Wei Li, Xinyan Xiao, Yajuan Lyu, and Yuanzhuo Wang. 2018. Improving neural abstractive document summarization with explicit information selection modeling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1787–1796, Brussels, Belgium.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain.

Yang Liu, Ivan Titov, and Mirella Lapata. 2019. Single document summarization as tree induction. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1745–1755, Minneapolis, Minnesota.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60, Baltimore, Maryland.

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 31st AAAI Conference on Artificial Intelligence*, pages 3075–3081, San Francisco, California.

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Çağlar Gülçehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence RNNs and beyond. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 280–290, Berlin, Germany.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018a. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1797–1807, Brussels, Belgium.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018b. Ranking sentences for extractive summarization with reinforcement learning. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1747–1759, New Orleans, Louisiana.

Romain Paulus, Caiming Xiong, and Richard Socher. 2018. A deep reinforced model for abstractive summarization. In *Proceedings of the 6th International Conference on Learning Representations*, Vancouver, Canada.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. In *CoRR, abs/1704.01444, 2017*.

Sascha Rothe, Shashi Narayan, and Aliaksei Severyn. 2019. Leveraging pre-trained checkpoints for sequence generation tasks. *arXiv preprint arXiv:1907.12461*.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 379–389, Lisbon, Portugal.

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 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083, Vancouver, Canada.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2818–2826, Las Vegas, Nevada.

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.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. In *arXiv preprint arXiv:1609.08144*.

Xingxing Zhang, Mirella Lapata, Furu Wei, and Ming Zhou. 2018. Neural latent extractive document summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 779–784, Brussels, Belgium.

Xingxing Zhang, Furu Wei, and Ming Zhou. 2019. HIBERT: Document level pre-training of hierarchical bidirectional transformers for document summarization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5059–5069, Florence, Italy. Association for Computational Linguistics.

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 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 654–663, Melbourne, Australia.