Sequence Level Contrastive Learning for Text Summarization

Shusheng Xu, [1*] **Xingxing Zhang,** [2] **Yi Wu,** [1,3] **Furu Wei** [2]

[1] IIIS, Tsinghua University, Beijing, China
[2] Microsoft Research Asia, Beijing, China
[3] Shanghai Qi Zhi Institute, Shanghai China
xuss20@mails.tsinghua.edu.cn
xizhang@microsoft.com
jxwuyi@gmail.com
fuwei@microsoft.com

Abstract

Contrastive learning models have achieved great success in unsupervised visual representation learning, which maximize the similarities between feature representations of different views of the same image, while minimize the similarities between feature representations of views of different images. In text summarization, the output summary is a shorter form of the input document and they have similar meanings. In this paper, we propose a contrastive learning model for supervised abstractive text summarization, where we view a document, its gold summary and its model generated summaries as different views of the same mean representation and maximize the similarities between them during training. We improve over a strong sequence-to-sequence text generation model (i.e., BART) on three different summarization datasets. Human evaluation also shows that our model achieves better faithfulness ratings compared to its counterpart without contrastive objectives. We release our code at https://github.com/xssstory/SeqCo.

Introduction

Document summarization is the task of rewriting a long document into a shorter form while still preserving its important content, which requires the model to understand the entire document. Many approaches for summarization has been explored in the literature and the most popular ones are *extractive summarization* and *abstractive summarization* (Nenkova and McKeown 2011). Summaries in their nature are abstractive. The summaries generated by extractive summarization methods are usually long and redundant, which bring bad reading experience. Therefore, we focus on abstractive summarization in this paper. Abstractive summarization is usually modeled as a sequence-to-sequence (Seq2Seq) learning problem (Sutskever, Vinyals, and Le 2014), where a document is viewed as a sequence of words and its summary another sequence of words (Nallapati et al. 2016).

Although abstractive models have been more and more powerful due to recent introduction of large pre-trained Transformers (Liu and Lapata 2019; Raffel et al. 2020; Dong et al. 2019; Lewis et al. 2020), the training paradigm for abstractive models is still not changed, which is to minimize the negative log-likelihood (NLL) between the model predicted word distributions and the gold summary. One great property of the summarization task is that a document and its summary should convey the same meaning, which is not modeled *explicitly* by the NLL loss.

In computer vision, contrastive learning methods for unsupervised image representation learning advanced the state-of-the-art in object detection and image segmentation (He et al. 2020b). The key idea is to minimize distances (or maximize similarities) between feature representations of different views of the same image (positive examples), while to maximize the distances between feature representations of views of different images (negative examples) (He et al. 2020b; Chen et al. 2020). As mentioned earlier, in summarization a document and its summary should convey the same meaning. Therefore, we view a document, its gold summary and its model generated summaries as different views of the same meaning representation and during training, we maximize the similarities between them. To achieve that, we propose SeqCo (as shorthand for **Seq**uence Level **Co**ntrastive Learning), which is based on contrastive learning. In addition to the gold summaries, we also use the dynamically generated summaries from our model during training to increase the diversity of inputs to SeqCo. In text summarization, an abstractive summarization model needs to first encode the document and then generate the summary. The contrastive objective in SeqCo tries to map representations of a document and its summary (or generated summary) to the same vector space, which intuitively helps the generation of summaries. Specifically, a document may contain distinct (or unnecessary) information from its summary. During training time, the contrastive objective between the document and summary actually encourages the model to encode important (and necessary) information from the document, otherwise the distance between the representations of document and summary will be large (the objective updates model parameters to make it small). Intuitively, the capability of encoding important information from documents would help to generate better summaries.

In experiments, we find our proposed contrastive learning based model SeqCo consistently improves upon a strong abstractive summarization model based on BART (Lewis et al.

[*] Work done during the first author's internship at Microsoft Research Asia.

2020) across three different summarization datasets (i.e., CNN/DailyMail (Hermann et al. 2015), New York Times (Sandhaus 2008) and XSum (Narayan, Cohen, and Lapata 2018)). Human evaluation also shows that our model SeqCo achieves better faithfulness ratings compared to its counterpart without contrastive objectives.

Related Work

The most popular paradigms for summarization are *extractive* and *abstractive* based approaches. We focus on abstractive summarization. Abstractive summarization may add new words or phrases when generating summaries, which is usually viewed as a sequence to sequence learning problem (Nallapati et al. 2016; See, Liu, and Manning 2017; Paulus, Xiong, and Socher 2018; Gehrmann, Deng, and Rush 2018). Probably because small and shallow LSTM (Hochreiter and Schmidhuber 1997) based attentive seq2seq models (Sutskever, Vinyals, and Le 2014; Bahdanau, Cho, and Bengio 2015) without pre-training are not powerful enough to model documents. Quality of summaries produced by these mdoels are not satisfactory (Liu and Lapata 2019). As the recent introduction of large pre-trained transformer models (Liu and Lapata 2019; Dong et al. 2019; Zou et al. 2020; Lewis et al. 2020; Zhang et al. 2020; Raffel et al. 2020), abstractive models are greatly improved. Best results for summarization are achieved by finetuning large models pre-trained with generation (or summarization) tailored objectives on huge amount of unlabeled text ($\geq$160G). Dong et al. (2019) pre-train jointly designed Transformer encoder and decoder with language model and masked language model objectives. Zhang et al. (2020) predict gapped sentences from a document removing these sentences and Lewis et al. (2020) propose sentence permutation and text infilling tasks to pre-train seq2seq transformers. There is also some work on combining extractive and abstractive summarization models (He et al. 2020a; Dou et al. 2021) or multiple summarization models (Liu, Dou, and Liu 2021). Unfortunately, pre-training transformers from scratch or combining multiple summarization systems are expensive, while our model can be applied to the light-weighted finetuning stage.

Convoluational neural networks pre-trained with contrastive learning methods advance the state-of-the-art in object detection and image segmentation in computer vision (He et al. 2020b). The idea is to minimize the distances between feature representations of different views of the same image (positive examples), while to maximize the distances between feature representations of views of different images (negative examples). To discriminate positive examples from negative examples, He et al. (2020b) maintain a queue of negative sample representations and utilize momentum updates for encoder of the queue to stabilize these representations. Chen et al. (2020) use other examples from the same batch as negative examples and as a result, they need a large batch size. These works above suggest that using a large number of negative examples is crucial to obtain good performance, which also increases the complexity for implementation. There is also an interesting line of work without using negative examples. Caron et al. (2020) employ online clustering to assign codes for two views of the same image and then use representation of one view to predict the cluster codes of the other. During the training of BYOL (Grill et al. 2020), they only minimized the distance between representations of two views of the same image and they use a momentum encoder for the *target* view to stabilize the training. Chen and He (2020) find that even the momentum encoder can be removed, although there might be a small drop in performance. The contrastive learning method used in our model is most related to BYOL (Grill et al. 2020) in the sense that we do not use negative examples either and we also employ a momentum encoder. In the models above, contrastive learning is applied in the unsupervised pre-training stage, which create different views of the same image by using effective data argumentation methods. In this paper, we take advantage of the nature of the summarization task and use the document, gold summary, and generated summary as different views of the same meaning representation (note that a summary is a shorter form of the original document). To fit sequence-to-sequence learning models for text generation, we handles two sequence of embeddings of discrete words, while the vision models handle two single embeddings of fixed dimensions. In addition, the generated summary are created dynamically during training with a model, which are more diverse than using non-model-based approaches in vision tasks.

In NLP, previously contrastive learning methods are mostly used in pre-training or natural language understanding tasks. For example, `word2vec` (Mikolov et al. 2013) learns the word embeddings by distinguishing words in a windows (positive examples) w.r.t. the current word and words randomly sampled (negative examples) using negative sampling. (Iter et al. 2020) propose a contrastive learning based method for language model pre-training, which predicts the relative distance between sentences using randomly sampled sentences as negative examples. More recently, MatchSum (Zhong et al. 2020) formulates extractive summarization as a semantic text matching problem using contrastive learning. Wu et al. (2020) measures the summary qualities without reference summaries by contrasting the document with the summaries using a ranking model. GSum (Dou et al. 2021) takes different kinds of external guidance as additional input to the document and advances summarization performance significantly. SimCLS (Liu and Liu 2021) proposes a contrastive based framework for abstractive summarization, which trains a model to rerank the candidate summaries of an abstractive model. We add constrastive learning to the training of an abstractive model by enforcing similarities between document, summary and generated summary, which does not need negative examples.

Model

In this section, we describe our contrastive learning model **SeqCo** (as shorthand for **Seq**uence Level **Co**ntrastive Learning) for abstractive text summarization. We first introduce abstractive text summarization models (i.e., Seq2Seq model), on which our model is based. Then we present SeqCo, which adapts contrastive learning to the sequence-to-sequence learning setting.

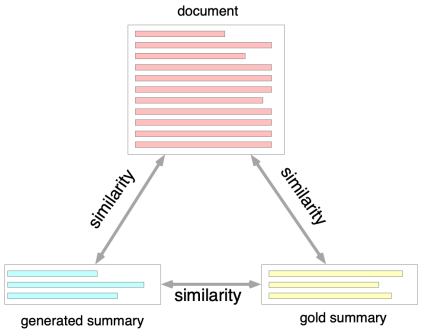

Figure 1: We enforce the similarities between the document, gold summary and model generated summary.

Abstractive Text Summarization

For text summarization, we can view the document as a long sequence of tokens[1] and the summary as a short sequence of tokens. Let $X = (x_0 = \texttt{<s>}, x_1, x_2, \ldots, x_{|X|} = \texttt{</s>})$ denote a document (i.e., the long sequence of tokens) and $Y = (y_0 = \texttt{<s>}, y_1, y_2, \ldots, y_{|Y|} = \texttt{</s>})$ its summary (i.e., the short sequence of tokens), where <s> and </s> are begin and end of sequence tokens. We predict Y one token at a time given X. We adopt the Transformer model (Vaswani et al. 2017), which is composed of an encoder Transformer and a decoder Transformer. Specifically, the encoder Transformer maps X into a sequence of hidden states $\mathbf{E} = (\mathbf{e}_0, \mathbf{e}_1, \ldots, \mathbf{e}_{|X|})$.

$$\mathbf{E} = \text{Trans}^{\text{E}}(X) \tag{1}$$

Supposing that the first $t-1$ tokens $y_{1:t-1}$ have been generated and we are generating y_t. The decoder Transformer computes the current hidden state $\mathbf{o}_t$ by *self* attending to the encoder hidden states $\mathbf{E}$ and proceeding tokens $y_{0:t-1}$.

$$\mathbf{o}_t = \text{Trans}^{\text{D}}(y_{0:t-1}, \mathbf{E}) \tag{2}$$

Note that during training, we can obtain $\mathbf{O} = (\mathbf{o}_1, ..., \mathbf{o}_{|Y|})$ in parallel.

$$\mathbf{O} = \text{Trans}^{\text{D}}(Y, \mathbf{E}) \tag{3}$$

The probability of y_t can be estimated using a linear projection and a *softmax* function

$$p(y_t|y_{0:t-1}, X) = \text{softmax}(\mathbf{W}^o\, \mathbf{o}_t) \tag{4}$$

$$\mathcal{L}^{\text{NLL}} = -\frac{1}{|Y|}\sum_{t=1}^{|Y|} \log p(y_t|y_{0:t-1}, X) \tag{5}$$

SeqCo: Sequence Level Contrastive Learning for Text Summarization

In text summarization, the summary Y is a shorter form of the input document X and they should convey the same meaning. Therefore, X and Y should be close in the semantic space at least after certain types of transformations. However, a Seq2Seq model is trained using the negative log-likelihood loss (see Equation (5)) and there is no explicit modeling for the similarity between X and Y. Further, during the training phase, given X as input, the model can also generate output sequences from its distribution by either beam search or sampling. Let $\hat{Y}$ denote one sample the model generated from X. Intuitively, $\hat{Y}$ should also be similar to both X and Y. As shown in figure 1, we enforce the similarities between X, Y and $\hat{Y}$ during model training. To do this, we propose SeqCo, which is a contrastive learning based model for text summarization.

Contrastive learning methods are proposed in the context of self-supervised learning for image representations (Wu et al. 2018; He et al. 2020b; Caron et al. 2020; Grill et al. 2020; Chen and He 2020). The training objective tries to make representations of different views of the same image closer (positive examples) while representations of views of different images apart from each other (negative examples). Inspired by Grill et al. (2020) and Chen and He (2020), we propose a model that does not need negative examples. In the following, we first define similarity measures between sequences and then we present how to equip the similarity measures into our training objective.

Sequence Representation Suppose that we have two sequences $S_i = (w_0^i, w_1^i, w_2^i, ..., w_{|S_i|}^i)$ and $S_j = (w_0^j, w_1^j, w_2^j, ..., w_{|S_j|}^j)$. S_i and S_j are two sequences, which we will maximize their similarity in Eq. 15. For example, S_i and S_j can be a document X and its gold summary Y, or document and generated summary, or gold summary and generated summary, just like Fig. 2. Before going to the similarity computation, we first convert them into sequences of hidden representations. We designed two mapping functions here. The first one (f_θ^{E}) is unconditional, which *reuses* the encoder of our Seq2Seq model (see Equation (1)):

$$f_\theta^{\text{E}}(S_i) = g(\text{Trans}^{\text{E}}(S_i)) \tag{6}$$

where $\text{Trans}^{\text{E}}(\cdot)$ is the Transformer encoder described in Equation (1) and $g(\cdot)$ is a feed-forward network that is used to give more freedom for encoding S_i. Here we use θ to denote the parameters in $f_\theta^{\text{E}}(\cdot)$.

The second mapping function (f_θ^{D}) is conditional, which takes of the input sequence into account.[2] Let X denote the input sequence and S_i is its *gold* output sequence or a sequence generated by the Seq2Seq model. In this mapping function, we employ both the encoder and the decoder of the Seq2Seq model (see Equation (1) and (2)):

$$f_\theta^{\text{D}}(S_i) = g(\text{Trans}^{\text{D}}(S_i, \text{Trans}^{\text{E}}(X))) \tag{7}$$

where $\text{Trans}^{\text{E}}(\cdot)$ and $\text{Trans}^{\text{D}}(\cdot)$ are the Transformer encoder and decoder described in Equation (1) and (3). As mentioned earlier, $g(\cdot)$ is a feed-forward network to give more freedom for encoding S_i. In $f_\theta^{\text{D}}(\cdot)$, we intend to use X as additional input to encode S_i more accurately in vector space. During

[1] We use *tokens* instead of *words*, because the sequence might be a sequence of sub-words.

[2] Note that in f_θ^{D} we only consider that S_i and S_i as the gold summary and the generated summary

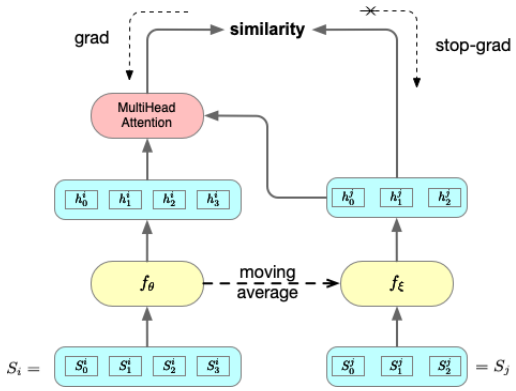

Figure 2: The contrastive objective. S_i and S_j are two sequences to contrast, f_θ and f_ξ have the same architecture, θ in f_θ is updated by gradient decent while ξ in f_ξ is the moving average of θ.

contrastive training, using $f_\theta^{\text{D}}(\cdot)$ can force the objective to optimize both the encoder and the decoder of the summarization model.

Sequence Similarity After defining the mapping functions, we are ready to compute sequence similarities. Without losing generality, let f_θ denote the mapping function, where θ is the parameter of the function. Note that f_θ can be either f_θ^{E} or f_θ^{D} (see Eq. (6) and (7) for details). We additionally employ another mapping function f_ξ, which has the same architecture as f_θ, but with parameter ξ. We obtain the representations of S_i and S_j by applying f_θ and f_ξ to them:

$$\begin{aligned}\mathbf{H}^i &= (\mathbf{h}_0^i, \mathbf{h}_1^i, \ldots, \mathbf{h}_{|S_i|}^i) = f_\theta(S_i) \\ \mathbf{H}^j &= (\mathbf{h}_0^j, \mathbf{h}_1^j, \ldots, \mathbf{h}_{|S_j|}^j) = f_\xi(S_j)\end{aligned} \tag{8}$$

To fully utilize the word-to-word interactions between the two sequences S_i and S_j, we apply a cross attention between $\mathbf{H}^i$ and $\mathbf{H}^j$:

$$\widetilde{\mathbf{H}^i} = \text{MultiHeadAttn}(\mathbf{H}^j, \mathbf{H}^i, \mathbf{H}^i) \tag{9}$$

where $\text{MultiHeadAttn}(\cdot,\cdot,\cdot)$ is the multi-head attention module (Vaswani et al. 2017) and $\mathbf{H}^j$, $\mathbf{H}^i$ and $\mathbf{H}^i$ are the query, key and value matrices, respectively. Note that the resulting $\widetilde{\mathbf{H}^i}$ and $\mathbf{H}^j$ have the same size. The similarity between S_i and S_j is the averaged cosine similarities of all vectors with the same index:

$$sim(S_i, S_j) = \frac{1}{|S_j|+1}\sum_{k=0}^{|S_j|}\cos(\widetilde{\mathbf{h}_k^i}, \mathbf{h}_k^j) \tag{10}$$

We adopt multi-head attention (MHA) for similarity computation for two reasons. 1) The sequences (esp. documents) are long and MHA takes all pairs of tokens across two sequences into account, which is intuitively more powerful than `[CLS]` pooling based methods (will introduce below). 2) The two sequences we compare may have different lengths (e.g., a document v.s. a summary). MHA can convert the hidden states of one sequence to the same length as the hidden states of another sequence (see Equation 9), which are easier to use for the similarity computation.

Note that we can also define a simpler similarity function using the `[CLS]` pooling as in BERT (Devlin et al. 2019):

$$sim(S_i, S_j) = \cos(q(\mathbf{h}_0^i), \mathbf{h}_0^j) \tag{11}$$

where q is a feed-forword network to project $\mathbf{h}_0^i$ following Grill et al. (2020). We obtained worse results using the similarity measure above and the measure also sometimes leads to numerical errors during training.

Training To make S_i and S_j closer, we can minimize the following loss:

$$\mathcal{L}_{\theta,\xi}(S_i, S_j) = 1 - sim(S_i, S_j) \tag{12}$$

As mentioned earlier, f_θ (the encoding function for S_i) and f_ξ (the encoding function for S_j) use different set of parameters (i.e., θ and ξ). If we update the parameters in both f_θ and f_ξ simultaneously, the optimization maybe too easy, which may lead to collapsed solutions (Grill et al. 2020). So we use f_ξ to produce regression *targets* for f_θ. Specifically, we do not update the parameters in f_ξ during the optimization of the loss above and ξ is a moving average of θ:

$$\xi = \tau\xi + (1-\tau)\theta \tag{13}$$

where $\tau \in [0,1]$ is a hyper-parameter to control the extend of retaining ξ. This contrastive objective is demonstrated in figure 2. Note that $\mathcal{L}_{\theta,\xi}(S_i, S_j)$ is not symmetric and we make the loss symmetric as follows:

$$\mathcal{L}_{\text{sim}}(S_i, S_j) = \mathcal{L}_{\theta,\xi}(S_i, S_j) + \mathcal{L}_{\theta,\xi}(S_j, S_i) \tag{14}$$

Hence, θ in f_θ will have more chances to be updated. As mentioned earlier, the encoding function f_θ can be either f_θ^{E} or f_θ^{D}. We use $\mathcal{L}_{\text{sim}}^{\text{E}}$ to denote the loss function using f_θ^{E} and $\mathcal{L}_{\text{sim}}^{\text{D}}$ to denote the loss function using f_θ^{D}.

To enforce the similarities between the document X, its gold summary Y and one of the model generated summary $\hat{Y}$, we employ the following loss function as our final training loss[3]:

$$\begin{aligned}\mathcal{L} =& \mathcal{L}^{\text{NLL}} + \lambda_{x-y}\mathcal{L}_{\text{sim}}^{\text{E}}(X,Y) + \lambda_{x-\hat{y}}\mathcal{L}_{\text{sim}}^{\text{E}}(X,\hat{Y}) \\ &+ \lambda_{y-\hat{y}}\mathcal{L}_{\text{sim}}^{\text{E}}(Y,\hat{Y}) + \lambda_{y-\hat{y}}^{\text{D}}\mathcal{L}_{\text{sim}}^{\text{D}}(Y,\hat{Y})\end{aligned} \tag{15}$$

This objective contains five terms. $\mathcal{L}^{\text{NLL}}$ is the negative log-likelihood; $\mathcal{L}_{sim}^{D}$ is the similarity loss w.r.t. $(Y,\hat{Y})$ with f_θ^{D}; $\mathcal{L}_{sim}^{\text{E}}$ terms are the similarity losses with f_θ^{E} w.r.t. (X,Y), $(X,\hat{Y})$ and $(Y,\hat{Y})$. λ_{x-y}, $\lambda_{x-\hat{y}}$, $\lambda_{y-\hat{y}}$ and $\lambda_{y-\hat{y}}^{\text{D}}$ are weight hyper-parameters for the last four terms. We completely train the model end-to-end following this loss function and empirically find that using a *single* similarity loss works better than using multiple ones, which is also more efficient for training. For example, we can set $\lambda_{x-\hat{y}} = 1.0$ and $\lambda_{x-y} = \lambda_{y-\hat{y}} = \lambda_{y-\hat{y}}^{\text{D}} = 0$. When $\hat{Y}$ is adopted, The model iteratively generates $\hat{Y}$ by using the loss to update parameters

[3]We can also use multiple generated summaries in training, we refrained to do so for efficiency reasons.

and generating new $\hat{Y}$. Since $\hat{Y}$ can not be perfect, iteratively generating $\hat{Y}$ makes it change toward ground-truth summary and make the positive examples for contrastive learning more accurate and diverse. Since SeqCo is designed for the fine-tuning stage, and the model SeqCo based on (i.e., BART) is pre-trained with a denoising auto-encoding objective, it can naturally generate the sequence with the same meaning as the input even before fine-tuning in a specific dataset. In addition, enforcing the similarity of y and $\hat{y}$ does not equals optimizing NLL, since the similarity loss is on sequence level while the NLL loss is on token level.

Experiments

In this section, we assess the performance of our contrastive learning model on the task of text summarization. We will first introduce the datasets we used. Then we present our implementation details. Finally, we compare our model with multiple previous models.

Datasets

CNNDM We conduct our experiments on three summarization datasets. The CNN/DailyMail dataset (CNNDM; Hermann et al. 2015) contains news articles and their associated highlights (i.e., reference summaries) from the CNN and Daily Mail websites. We follow the standard pre-processing steps in (See, Liu, and Manning 2017)[4] and the resulting dataset contains 287,226 articles for training, 13,368 for validation and 11,490 for test.

NYT The New York Times dataset (NYT; Sandhaus 2008) is composed of articles published by the New York Times with summaries written by library scientists. Following the pre-processing procedures in (Durrett, Berg-Kirkpatrick, and Klein 2016; Liu and Lapata 2019), we first obtain 110,540 articles with abstractive summaries. The test set is constructed from the 9,706 articles published after January 1, 2007. After removing articles whose summaries are shorter than 50 words, the final test set contains 3,452 articles. The remaining 100,834 articles are filtered and splitted into 38,264 articles for training and 4,000 articles for validation.

XSum The articles in the XSum dataset (Narayan, Cohen, and Lapata 2018) are from the BBC website with accompanying single sentence summaries, which are professionally written. We use the official splits of (Narayan, Cohen, and Lapata 2018) (i.e., 204,045 articles for training, 11,332 articles for validation and 11,334 articles for test).

All datasets are tokenized with the byte-pair encoding of GPT2 (Radford et al. 2019).

Implementation Details

Our model is initialized from $\text{BART}_{\text{Large}}$ (Lewis et al. 2020). Therefore, the size is identical with $\text{BART}_{\text{Large}}$ (Lewis et al. 2020). Specifically, the encoder and decoder are all 12-layer transformers with 16 attention heads, hidden size 1,024 and feed-forward filter size 4,096, which amounts to 406M trainable parameters. We also have additional component for contrastive learning. The feedforward network g (see Equation (6) and (7)) for projecting sequence features contains one hidden layer of 4,096 neurons with ReLU activation function. The multi-head attention module (see Equation (9)) used to compute cross attention between sequences also has 16 heads. These two components above contribute to an extra 13M trainable parameters.

We optimize the model using Adam with $\beta_1 = 0.9, \beta_2 = 0.999$. Following (Lewis et al. 2020), we employ a linear schedule for the learning rate. We firstly warmup the model by increasing the learning rate linearly to a peak learning rate and then decrease the learning rate linearly to zero. The peak learning rate, warmup steps, total number of updates and batch size are tuned on validation sets and are different across datasets, which are 1000, 20000, $4e-5$, 128 on CNNDM, 500, 5000, $2e-5$, 64 on NYT, 500, and 15000, $6e-5$, 64 on XSum. In all datasets, the number of training epochs are between 5 to 10. During the optimization, parameters ξ in the online encoding function f_ξ (see Equation (6) and (7)) are not updated. Parameters ξ in f_ξ are updated following Equation (13) with $\tau = 0.99$. We employ label smoothing of 0.1 (Szegedy et al. 2016; Vaswani et al. 2017). The models for CNNDM are trained on 8 Tesla V100 GPUs, and the models for the other datasets are trained on 4 Tesla V100 GPUs. During decoding, we select minimum generated length and length penalty according to ROUGE scores on the validation set. Following (Paulus, Xiong, and Socher 2018), we also blocked repeated trigrams during beam search. Following (Lewis et al. 2020), the articles are truncated to 1024 tokens in both training and decoding.

Evaluations

We use ROUGE (Lin 2004) to measure the quality of generated summaries. We reported full-length F1 based ROUGE-1, ROUGE-2 and ROUGE-L scores on CNNDM and XSum datasets. Following (Durrett, Berg-Kirkpatrick, and Klein 2016), we use the limited-length recall based ROUGE-1, ROUGE-2 and ROUGE-L on NYT, where generated summaries are truncated to the length of gold summaries. ROUGE scores are computed with the `ROUGE-1.5.5.pl` script[5].

Results

We present our main results on the CNNDM dataset in Table 1. We compare our model against both extractive and abstractive systems. The first block summarizes the results for extractive systems. Lead3 is a baseline which simply takes the leading three sentences in a document as its summary. BERTEXT (Liu and Lapata 2019) employs BERT as encoder and predicts whether a sentence is a summary. MatchSum (Zhong et al. 2020) is the best performing extractive models, which formulates summarization as a semantic text matching problem using contrastive learning. The abstractive models are in the second block. PTGen (See, Liu, and Manning 2017) is a LSTM-based Seq2Seq model augmented with copy and

[4] Available at https://github.com/abisee/cnn-dailymail

[5] with -c 95 -r 1000 -n 2 -a -m arguments

coverage models. Large pre-trained language models mostly dominate summarization. BERTSUMEXTABS (Liu and Lapata 2019) is an abstractive model with encoder initialized with BERT and decoder randomly initialized. UniLM (Dong et al. 2019) is trained using language modeling and masked language modeling objectives. T5 (Raffel et al. 2020), PEGASUS (Zhang et al. 2020), BART (Lewis et al. 2020) and STEP (Zou et al. 2020) pre-train Seq2Seq transformers using different unsupervised text-to-text tasks. PEGASUS (Zhang et al. 2020) is trained by predicting gapped sentences (selected by some heuristics) in a document given the document with these sentences masked. Similar to BERTSUMEXTABS, the encoder of STEP is initialized from RoBERTa (Liu et al. 2019). BART + R3F (Aghajanyan et al. 2021) applies a trust region theory based fine-tuning method to BART. Our model is based on BART and therefore we also re-implement BART (BART⋆). These models above are single models. We also present the results of recent combination models in the third block. CTRLsum (He et al. 2020a) and GSum (Dou et al. 2021) combine a keywords extraction model (or an extractive model) with an abstractive model by taking the resulting keywords (or sentences) as additional input. SimCLS(Chen et al. 2020) and Refsum (Liu, Dou, and Liu 2021) train re-ranking models to rank multiple candidate summaries.

The fourth block includes results of our model SeqCo. As mentioned in Equation (15), we can do contrastive learning between document and gold summary (i.e., SeqCo (λ_{x-y})), document and generated summary (i.e., SeqCo ($\lambda_{x-\hat{y}}$)) as well as gold summary and generated summary (i.e., SeqCo ($\lambda_{y-\hat{y}}$)). Note SeqCo (λ_{*-*}) means that $\lambda_{*-*} > 0$ and all the other λs equal to zero in Equation (15)[6]. We can see that SeqCo (λ_{x-y}), SeqCo ($\lambda_{x-\hat{y}}$) and SeqCo ($\lambda_{y-\hat{y}}$) all outperform BART⋆ significantly ($p < 0.05$) measured by the ROUGE script, which demonstrates the effectiveness of our proposed contrastive methods. SeqCo ($\lambda_{y-\hat{y}}$) outperforms all *single* models in comparison (first two blocks) and differences between them are significant w.r.t. the ROUGE script. We also observe that using generated summaries in contrastive learning leads to better performance (i.e., results of SeqCo ($\lambda_{x-\hat{y}}$) and SeqCo ($\lambda_{y-\hat{y}}$) are better), which is not surprising. Generated summaries are created dynamically during training and they might be more diverse than gold summaries.

It is also possible to employ multiple pairs of text for contrastive learning. Results on validation set with different combinations of text pairs are shown in Table 2. We obtain worse results with more than one pair of text in contrastive learning. Perhaps because the information learned using different pair of text is a bit redundant. We compared the results on the validation and test sets of the other two datasets and observed similar trends.[7] We find best results are achieved by using a single similarity loss on all datasets except for the validation set of XSum, where SeqCo (λ_{x-y} + $\lambda_{y-\hat{y}}$) and SeqCo (λ_{x-y} + $\lambda_{x-\hat{y}}$ + $\lambda_{y-\hat{y}}$) outperform SeqCo(x-y) slightly.

[6]We tune $\lambda_{x-y}, \lambda_{x-\hat{y}}, \lambda_{y-\hat{y}} \in \{0.5, 1.0\}$ on the validation set when > 0

[7]Please refer to https://arxiv.org/abs/2109.03481 for Appedix, detailed numbers are shown in Appendix.

Model	R-1	R-2	R-L
Extractive			
Lead3	40.34	17.70	36.57
BERTEXT (2019)	43.85	20.34	39.90
MATCHSUM (2020)	**44.41**	**20.86**	**40.55**
Abstractive			
PTGen (2017)	39.53	17.28	36.38
BERTSUMEXTABS (2019)	42.13	19.60	39.18
UniLM (2019)	43.47	20.30	40.63
T5 (2020)	43.52	**21.55**	40.69
PEGASUS (C4)	43.90	21.20	40.76
PEGASUS (HugeNews)	44.17	21.47	41.11
STEP (2020)	44.03	21.13	41.20
BART (2020)	44.16	21.28	40.90
BART⋆ (2020)	44.10	21.31	40.91
BART + R3F (2021)	**44.38**	21.53	**41.17**
Combination Methods			
CTRLsum (2020a)	45.65	22.35	42.50
GSum (2021)	45.94	22.32	42.48
simCLS (2021)	**46.67**	22.15	43.54
Refsum (2021)	46.12	**22.46**	**42.92**
Ours			
SeqCo (λ_{x-y})	44.66†	21.57*	41.38*
SeqCo ($\lambda_{x-\hat{y}}$)	44.94†	**21.82**†	41.68†
SeqCo ($\lambda_{y-\hat{y}}$)	**45.02**†	21.80†	**41.75**†

Table 1: Results on the test split of CNNDM using full length F1 based ROUGE-1/2/L. ⋆ means our own re-implementation. SeqCo (λ_{x-y}), SeqCo ($\lambda_{x-\hat{y}}$) and SeqCo ($\lambda_{y-\hat{y}}$) stand for contrastive learning between document and gold summary, document and generated summary as well as gold and generated summary, respectively. * means outperforms BART⋆ significantly, † means outperforms best performing single model "BART+R3F" significantly ($p < 0.05$). Models in "Combination Methods" employ multiple summarization models.

Given the fact that adding one more similarity loss increases around 30% training time and the observations above, we recommend using a single similarity loss. We probably need to encourage the "disagreement" between them (we leave this for future work). As mentioned in Equation (7), we can also use decoder based encoding function f_θ^{D} (see the SeqCo ($\lambda^D_{y-\hat{y}}$) and SeqCo ($\lambda_{y-\hat{y}}$) rows in Table 2) and we obtain worse results. It may because influencing the decoding during contrastive training is too aggressive. Therefore, we only report results of contrastive models on single pair of text (i.e., SeqCo (λ_{x-y}), SeqCo ($\lambda_{x-\hat{y}}$) and SeqCo ($\lambda_{y-\hat{y}}$)) on NYT and XSum. We also propose to employ multi-head attention based similarity modeling (see Equation (9) and (10)) rather than `[CLS]` based method (see Equation (11)). It also shows attention based similarity, which takes associations across two sequences into account, is better (see SeqCo ($\lambda_{y-\hat{y}}$) and SeqCo ($\lambda_{y-\hat{y}}$) w/ `[CLS]` rows in Table 2).

Results on NYT are shown in Table 3 and the trend is similar. ROBERTA-S2S is a transformer based Seq2Seq model with encoder initialized from RoBERTa (Liu et al. 2019) and its results are reported in (Zou et al. 2020). SeqCo ($\lambda_{x-\hat{y}}$)

Model	R-1	R-2	R-L
BART⋆	45.24	22.10	42.01
SeqCo (λ_{x-y})	45.60	22.30	42.36
SeqCo ($\lambda_{x-\hat{y}}$)	45.80	22.39	42.57
SeqCo ($\lambda_{y-\hat{y}}$)	**45.88**	**22.46**	**42.66**
SeqCo ($\lambda_{y-\hat{y}}$) w/ `[CLS]`	45.72	22.42	42.48
SeqCo (λ_{x-y} + $\lambda_{x-\hat{y}}$)	45.68	22.38	42.45
SeqCo (λ_{x-y} + $\lambda_{y-\hat{y}}$)	45.62	22.29	42.37
SeqCo ($\lambda_{x-\hat{y}}$ + $\lambda_{y-\hat{y}}$)	45.72	22.35	42.45
SeqCo (λ_{x-y} + $\lambda_{x-\hat{y}}$ + $\lambda_{y-\hat{y}}$)	45.72	22.38	42.46
SeqCo ($\lambda^{D}_{y-\hat{y}}$)	45.74	22.39	41.55

Table 2: Results on the validation split of CNNDM using full length F1 based ROUGE-1/2/L. "w/ [CLS]" means we replace MHA with [CLS] pooling defined in Eq. 11

.

Model	R-1	R-2	R-L
Extractive			
Lead3	39.58	20.11	35.78
BERTEXT (2019)	**46.66**	**26.35**	**42.62**
Abstractive			
PTGen (2017)	43.71	26.40	-
BERTSUMEXTABS (2019)	49.02	31.02	45.55
ROBERTA-S2S (2019)	45.92	29.48	42.73
STEP (2020)	50.03	32.12	46.25
BART⋆ (2020)	**53.20**	**35.04**	**49.23**
Combination Methods			
GSum (2021)	54.27	35.37	47.63
Ours			
SeqCo (λ_{x-y})	53.79	35.43	49.84
SeqCo ($\lambda_{x-\hat{y}}$)	**54.25***	**35.82***	**50.24***
SeqCo ($\lambda_{y-\hat{y}}$)	54.14	35.69	50.11

Table 3: Results on the test split of NYT using limited-length recall based ROUGE. ⋆ means our own re-implementation. * means outperforms BART⋆ significantly ($p < 0.05$).

outperforms BART⋆ by +1.0 ROUGE-1, +0.8 ROUGE-2 and +1.0 ROUGE-L and the differences between them are significant measured by the ROUGE script. SeqCo ($\lambda_{x-\hat{y}}$) obtains better results than all models in comparison. We again observe that using generated summaries in SeqCo are better than using gold summaries only.

Table 4 summarizes our results on the XSum dataset. BART⋆ (our reimplementation) are better at ROUGE-1, but worse at ROUGE-2 and ROUGE-L compared to BART. SeqCo (λ_{x-y}) outperforms BART⋆ significantly measured with the ROUGE script. Results of SeqCo (λ_{x-y}) are better than all previously published models except for PEGASUS (HugeNews) and Refsum. It is not entirely surprising, because PEGASUS (HugeNews) is trained on 3,800 GB news data (the same genre as the XSum dataset), while PEGASUS(C4) is pre-trained on the C4 dataset consist of text from 350M Web pages (750GB) and performs worse than PEGASUS (HugeNews). Refsum reranks outputs of PEGASUS (HugeNews). Note that the pre-trained transformer (i.e., BART) in SeqCo is trained on only 160 GB data, which also contains

Model	R-1	R-2	R-L
Extractive			
Lead3	16.30	1.60	11.95
MATCHSUM (2020)	**24.86**	**4.66**	**18.41**
Abstractive			
PTGen (2017)	28.10	8.02	21.72
BERTSUMEXTABS (2019)	38.81	16.50	31.27
ROBERTA-S2S (2019)	43.54	20.49	35.75
STEP (2020)	43.02	20.11	35.34
PEGASUS (C4)	45.20	22.06	36.99
PEGASUS (HugeNews)	**47.21**	**24.56**	**39.25**
BART (2020)	45.14	22.27	37.25
BART⋆ (2020)	45.35	22.01	36.76
Combination Methods			
GSum (2021)	45.40	21.89	36.67
simCLS (2021)	**47.61**	**24.57**	**39.44**
Refsum (2021)	47.45	24.55	39.41
Ours			
SeqCo (λ_{x-y})	**45.65***	**22.41***	**37.04***
SeqCo ($\lambda_{x-\hat{y}}$)	45.6	22.36	36.94
SeqCo ($\lambda_{y-\hat{y}}$)	45.52	22.24	36.90

Table 4: Results on the test split of XSum using full length F1 based ROUGE. ⋆ means our own re-implementation. * means outperforms BART⋆ significantly ($p < 0.05$).

data in other domains rather than news data.

Human Evaluation We do human evaluations on CNNDM, NYT and XSum with 100 documents each. We asked the participants to rank the outputs of different systems according to their faithfulness and the mean rank scores (lower is better) are shown in table 5. We employed (self-reported) native speakers to annotate our output summaries on Amazon Mechanical Turk. To further guarantee the annotation quality, we filter out the annotated assignments which were done less than two minutes (average time spent per assignment is 6 minutes). After the filtering process, we guarantee each document is annotated by three annotators. In CNNDM and NYT datasets, Seqco outperforms BART significantly. In XSum dataset, there are no significant differences among these systems. It may be because generated summaries in XSum are shorter, which are difficult for annotators to tell the differences. We calculate the ratios of agreement between annotators (i.e., ratio of all three annotators' agreement and ratios of at least two annotators' agreement) to measure the agreement for human evaluation. As shown in table 6, there are around 30% of summaries that all of 3 participants give the same annotations, and more than 90% of summaries obtained the same annotations by at least 2 annotators. In addition, the Fleiss' Kappa scores are 0.329 on CNNDM, 0.313 on NYT and 0.364 on XSum, which demonstrate a fair degree of agreement. We believe the agreement between annotators is reasonable.

Analysis *Different from CNNDM and NYT, why does using generated summaries in contrastive learning perform worse on XSum?* As shown in Table 7, it may because XSum is more abstractive (see the novel $n$gram statistics of Gold on the

systems	BART	$x-y$	$x-\hat{y}$	$y-\hat{y}$
CNNDM	2.62	2.51	2.45*	2.42*
NYT	2.68	2.46*	2.39*	2.46*
XSum	2.47	2.44	2.58	2.50

Table 5: Human evaluation on faithfulness with mean rank (lower is better). We randomly sample 100 documents for each dataset and asked the participants to rank the outputs of different systems according to their faithfulness. * means this result is significantly different ($p < 0.05$) from BART.

Datasets	CNNDM	NYT	Xsum
3 agree	26.50%	31.00%	29.50%
$\geq$ 2 agree	96.25%	95.75%	94.75%

Table 6: The ratios of agreement between annotators.

Model	1-gram	2-gram	3-gram
CNNDM			
Gold	0.1360	0.4871	0.6908
BART	0.0157	0.1140	0.2161
SeqCo	0.0228	0.1524	0.2769
NYT			
Gold	0.1064	0.4260	0.6189
BART	0.0350	0.2231	0.3896
SeqCo	0.0368	0.2284	0.3961
XSum			
Gold	0.3752	0.8328	0.9551
BART	0.2821	0.7341	0.8924
SeqCo	0.2929	0.7465	0.9015

Table 7: Proportions of novel n-grams w.r.t. original documents in gold and model generated summaries on the validation sets of CNNDM, NYT and XSum.

three datasets) and more difficult. As a result, the generated summaries are easier to have different meanings from their documents and gold summaries (at least in the early stage of training). Maybe that is the reason why the $x-\hat{y}$ and $y-\hat{y}$ objective is worse than the $x-y$ objective. CNNDM and NYT are less abstractive and the generated summaries could retain the main meanings more easily and are also more diverse (compared to gold summaries), which leads to the $x-\hat{y}$ and $y-\hat{y}$ objectives work better.

We can also see from Table 7 that SeqCo can either be more abstractive than BART or almost as abstractive as BART. To choose the contrastive objective, our suggestion is 1) for the datasets whose summaries are highly abstractive, choose the $x-y$ pair as the contrastive objective; 2) for less abstractive datasets (the case for most datasets), choose either $x-\hat{y}$ or $y-\hat{y}$ as the contrastive objective. As far as we observed, the performance of $x-\hat{y}$ and $y-\hat{y}$ are similar.

Ablation Study We list the ablation results on three datasets in the appendix A.[8] We compared single similarity loss v.s. multiple similarity losses on the validation and test sets and observed the similar trends. We find best results are achieved by using a single similarity loss on all datasets except for the validation set of XSum, where SeqCo (λ_{x-y} + $\lambda_{y-\hat{y}}$) and SeqCo (λ_{x-y} + $\lambda_{x-\hat{y}}$ + $\lambda_{y-\hat{y}}$) outperform SeqCo(x-y) slightly. Given the fact that adding one more similarity loss increases around 30% training time and the observations above, we recommend using a single similarity loss.

Example Outputs Some example outputs of SeqCo and BART$\star$ are also listed in appendix B. In conclusion, BART sometimes miss some important points, while SeqCo can do better.

Conclusions

In text summarization, a document, its gold summary and model generated summaries can be viewed as different views of the same meaning representation. We propose SeqCo, a sequence level contrastive learning model for text summarization, which intends to minimize distances between the document, its summary and its generated summaries during training. Experiments on three summarization datasets (CNNDM, NYT and XSum) show that SeqCo consistantly improves a strong Seq2Seq text generation model. In the future, we plan to extend SeqCo in the multi-lingual or cross-lingual text generation tasks. We observed in experiments that using multiple contrastive objectives did not improve the results. We are interested in developing methods for regularizing different contrastive objectives.

[8]Please refer to https://arxiv.org/abs/2109.03481 for Appedix.

References

Aghajanyan, A.; Shrivastava, A.; Gupta, A.; Goyal, N.; Zettlemoyer, L.; and Gupta, S. 2021. Better Fine-Tuning by Reducing Representational Collapse. In *International Conference on Learning Representations*.

Bahdanau, D.; Cho, K.; and Bengio, Y. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. *CoRR*, abs/1409.0473.

Caron, M.; Misra, I.; Mairal, J.; Goyal, P.; Bojanowski, P.; and Joulin, A. 2020. Unsupervised Learning of Visual Features by Contrasting Cluster Assignments. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*.

Chen, T.; Kornblith, S.; Norouzi, M.; and Hinton, G. 2020. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, 1597–1607. PMLR.

Chen, X.; and He, K. 2020. Exploring Simple Siamese Representation Learning. *arXiv preprint arXiv:2011.10566*.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL-HLT*.

Dong, L.; Yang, N.; Wang, W.; Wei, F.; Liu, X.; Wang, Y.; Gao, J.; Zhou, M.; and Hon, H.-W. 2019. Unified language model pre-training for natural language understanding and generation. In *Advances in Neural Information Processing Systems*, 13063–13075.

Dou, Z.-Y.; Liu, P.; Hayashi, H.; Jiang, Z.; and Neubig, G. 2021. GSum: A General Framework for Guided Neural Abstractive Summarization. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 4830–4842. Online: Association for Computational Linguistics.

Durrett, G.; Berg-Kirkpatrick, T.; and Klein, D. 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)*, 1998–2008. Berlin, Germany: Association for Computational Linguistics.

Gehrmann, S.; Deng, Y.; and Rush, A. 2018. Bottom-Up Abstractive Summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 4098–4109. Brussels, Belgium: Association for Computational Linguistics.

Grill, J.-B.; Strub, F.; Altché, F.; Tallec, C.; Richemond, P. H.; Buchatskaya, E.; Doersch, C.; Pires, B. A.; Guo, Z. D.; Azar, M. G.; et al. 2020. Bootstrap your own latent: A new approach to self-supervised learning. *arXiv preprint arXiv:2006.07733*.

He, J.; Kryściński, W.; McCann, B.; Rajani, N.; and Xiong, C. 2020a. Ctrlsum: Towards generic controllable text summarization. *arXiv preprint arXiv:2012.04281*.

He, K.; Fan, H.; Wu, Y.; Xie, S.; and Girshick, R. 2020b. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 9729–9738.

Hermann, K. M.; Kocisky, T.; Grefenstette, E.; Espeholt, L.; Kay, W.; Suleyman, M.; and Blunsom, P. 2015. Teaching Machines to Read and Comprehend. In Cortes, C.; Lawrence, N.; Lee, D.; Sugiyama, M.; and Garnett, R., eds., *Advances in Neural Information Processing Systems*, volume 28, 1693–1701. Curran Associates, Inc.

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

Iter, D.; Guu, K.; Lansing, L.; and Jurafsky, D. 2020. Pretraining with Contrastive Sentence Objectives Improves Discourse Performance of Language Models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 4859–4870. Online: Association for Computational Linguistics.

Lewis, M.; Liu, Y.; Goyal, N.; Ghazvininejad, M.; Mohamed, A.; Levy, O.; Stoyanov, V.; and Zettlemoyer, L. 2020. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 7871–7880.

Lin, C.-Y. 2004. ROUGE: A Package for Automatic Evaluation of Summaries. In *Text Summarization Branches Out*, 74–81. Barcelona, Spain: Association for Computational Linguistics.

Liu, Y.; Dou, Z.-Y.; and Liu, P. 2021. RefSum: Refactoring Neural Summarization. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 1437–1448. Online: Association for Computational Linguistics.

Liu, Y.; and Lapata, M. 2019. Text Summarization with Pretrained Encoders. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 3730–3740. Hong Kong, China: Association for Computational Linguistics.

Liu, Y.; and Liu, P. 2021. SimCLS: A Simple Framework for Contrastive Learning of Abstractive Summarization. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, 1065–1072. Online: Association for Computational Linguistics.

Liu, Y.; Ott, M.; Goyal, N.; Du, J.; Joshi, M.; Chen, D.; Levy, O.; Lewis, M.; Zettlemoyer, L.; and Stoyanov, V. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Mikolov, T.; Chen, K.; Corrado, G. S.; and Dean, J. 2013. Efficient Estimation of Word Representations in Vector Space. In *ICLR*.

Nallapati, R.; Zhou, B.; dos Santos, C.; GuÌ‡lçehre, Ç.; and Xiang, B. 2016. Abstractive Text Summarization using Sequence-to-sequence RNNs and Beyond. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, 280–290. Berlin, Germany: Association for Computational Linguistics.

Narayan, S.; Cohen, S. B.; and Lapata, M. 2018. 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*. Brussels, Belgium.

Nenkova, A.; and McKeown, K. 2011. *Automatic summarization*. Now Publishers Inc.

Paulus, R.; Xiong, C.; and Socher, R. 2018. A Deep Reinforced Model for Abstractive Summarization. In *International Conference on Learning Representations*.

Radford, A.; Wu, J.; Child, R.; Luan, D.; Amodei, D.; and Sutskever, I. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8): 9.

Raffel, C.; Shazeer, N.; Roberts, A.; Lee, K.; Narang, S.; Matena, M.; Zhou, Y.; Li, W.; and Liu, P. J. 2020. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *J. Mach. Learn. Res.*, 21: 140:1–140:67.

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

See, A.; Liu, P. J.; and Manning, C. D. 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)*, 1073–1083. Vancouver, Canada: Association for Computational Linguistics.

Sutskever, I.; Vinyals, O.; and Le, Q. V. 2014. Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27: 3104–3112.

Szegedy, C.; Vanhoucke, V.; Ioffe, S.; Shlens, J.; and Wojna, Z. 2016. Rethinking the Inception Architecture for Computer Vision. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2818–2826.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. In *Advances in neural information processing systems*, 5998–6008.

Wu, H.; Ma, T.; Wu, L.; Manyumwa, T.; and Ji, S. 2020. Unsupervised Reference-Free Summary Quality Evaluation via Contrastive Learning. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 3612–3621. Online: Association for Computational Linguistics.

Wu, Z.; Xiong, Y.; Yu, S. X.; and Lin, D. 2018. Unsupervised feature learning via non-parametric instance discrimination. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 3733–3742.

Zhang, J.; Zhao, Y.; Saleh, M.; and Liu, P. 2020. Pegasus: Pre-training with extracted gap-sentences for abstractive summarization. In *International Conference on Machine Learning*, 11328–11339. PMLR.

Zhong, M.; Liu, P.; Chen, Y.; Wang, D.; Qiu, X.; and Huang, X. 2020. Extractive Summarization as Text Matching. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 6197–6208. Online: Association for Computational Linguistics.

Zou, Y.; Zhang, X.; Lu, W.; Wei, F.; and Zhou, M. 2020. Pre-training for Abstractive Document Summarization by Reinstating Source Text. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 3646–3660.