Learning to Encode Text as Human-Readable Summaries using Generative Adversarial Networks

Yau-Shian Wang
National Taiwan University
king6101@gmail.com

Hung-Yi Lee
National Taiwan University
tlkagkb93901106@gmail.com

Abstract

Auto-encoders compress input data into a latent-space representation and reconstruct the original data from the representation. This latent representation is not easily interpreted by humans. In this paper, we propose training an auto-encoder that encodes input text into human-readable sentences, and unpaired abstractive summarization is thereby achieved. The auto-encoder is composed of a generator and a reconstructor. The generator encodes the input text into a shorter word sequence, and the reconstructor recovers the generator input from the generator output. To make the generator output human-readable, a discriminator restricts the output of the generator to resemble human-written sentences. By taking the generator output as the summary of the input text, abstractive summarization is achieved without document-summary pairs as training data. Promising results are shown on both English and Chinese corpora.

1 Introduction

When it comes to learning data representations, a popular approach involves the auto-encoder architecture, which compresses the data into a latent representation without supervision. In this paper we focus on learning text representations. Because text is a sequence of words, to encode a sequence, a sequence-to-sequence (seq2seq) auto-encoder (Li et al., 2015; Kiros et al., 2015) is usually used, in which a RNN is used to encode the input sequence into a fixed-length representation, after which another RNN is used to decode the original input sequence given this representation.

Although the latent representation learned by the seq2seq auto-encoder can be used in downstream applications, it is usually not human-readable. A human-readable representation should comply the rule of human grammar and can be comprehended by human. Therefore, in this work, we use comprehensible natural language as a latent representation of the input source text in an auto-encoder architecture. This human-readable latent representation is shorter than the source text; in order to reconstruct the source text, it must reflect the core idea of the source text. Intuitively, the latent representation can be considered a summary of the text, so unpaired abstractive summarization is thereby achieved.

The idea that using human comprehensible language as a latent representation has been explored on text summarization, but only in a semi-supervised scenario. Previous work (Miao and Blunsom, 2016) uses a prior distribution from a pre-trained language model to constrain the generated sequence to natural language. However, to teach the compressor network to generate text summaries, the model is trained using labeled data. In contrast, in this work we need no labeled data to learn the representation.

As shown in Fig. 1, the proposed model is composed of three components: a generator, a discriminator, and a reconstructor. Together, the generator and reconstructor form a text auto-encoder. The generator acts as an encoder in generating the latent representation from the input text. Instead of using a vector as latent representation, however, the generator generates a word sequence much shorter than the input text. From the shorter text, the reconstructor reconstructs the original input of the generator. By minimizing the reconstruction loss, the generator learns to generate short text segments that contain the main information in the original input. We use the seq2seq model in modeling the generator and reconstructor because both have input and output sequences with different lengths.

However, it is very possible that the generator's output word sequence can only be processed and recognized by the reconstructor but is

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4187–4195
Brussels, Belgium, October 31 - November 4, 2018.

not readable by humans. Here, instead of regularizing the generator output with a pre-trained language model (Miao and Blunsom, 2016), we borrow from adversarial auto-encoders (Makhzani et al., 2015) and cycle GAN (Zhu et al., 2017) and introduce a third component – the discriminator – to regularize the generator's output word sequence. The discriminator and the generator form a generative adversarial network (GAN) (Goodfellow et al., 2014). The discriminator discriminates between the generator output and human-written sentences, and the generator produces output as similar as possible to human-written sentences to confuse the discriminator. With the GAN framework, the discriminator teaches the generator how to create human-like summary sentences as a latent representation. However, due to the non-differential property of discrete distributions, generating discrete distributions by GAN is challenging. To tackle this problem, in this work, we proposed a new kind of method on language generation by GAN.

By achieving unpaired abstractive text summarization, machine is able to unsupervisedly extract the core idea of the documents. This approach has many potential applications. For example, the output of the generator can be used for the downstream tasks like document classification and sentiment classification. In this study, we evaluate the results on an abstractive text summarization task. The output word sequence of the generator is regarded as the summaries of the input text. The model is learned from a set of documents without summaries. As most documents are not paired with summaries, for example the movie reviews or lecture recordings, this technique makes it possible to learn summarizer to generate summaries for these documents. The results show that the generator generates summaries with reasonable quality on both English and Chinese corpora.

2 Related Work

Abstractive Text Summarization

Recent model architectures for abstractive text summarization basically use the sequence-to-sequence (Sutskever et al., 2014) framework in combination with various novel mechanisms. One popular mechanism is attention (Bahdanau et al., 2015), which has been shown helpful for summarization (Nallapati et al., 2016; Rush et al., 2015; Chopra et al., 2016). It is also possible to directly optimize evaluation metrics such as ROUGE (Lin, 2004) with reinforcement learning (Ranzato et al., 2016; Paulus et al., 2017; Bahdanau et al., 2016). The hybrid pointer-generator network (See et al., 2017) selects words from the original text with a pointer (Vinyals et al., 2015) or from the whole vocabulary with a trained weight. In order to eliminate repetition, a coverage vector (Tu et al., 2016) can be used to keep track of attended words, and coverage loss (See et al., 2017) can be used to encourage model focus on diverse words. While most papers focus on supervised learning with novel mechanisms, in this paper, we explore unsupervised training models.

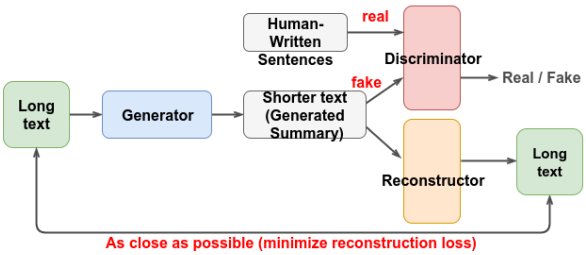

Figure 1: Proposed model. Given long text, the generator produces a shorter text as a summary. The generator is learned by minimizing the reconstruction loss together with the reconstructor and making discriminator regard its output as human-written text.

GAN for Language Generation

In this paper, we borrow the idea of GAN to make the generator output human-readable. The major challenge in applying GAN to sentence generation is the discrete nature of natural language. To generate a word sequence, the generator usually has non-differential parts such as *argmax* or other sample functions which cause the original GAN to fail.

In (Gulrajani et al., 2017), instead of feeding a discrete word sequence, the authors directly feed the generator output layer to the discriminator. This method works because they use the earth mover's distance on GAN as proposed in (Arjovsky et al., 2017), which is able to evaluate the distance between a discrete and a continuous distribution. SeqGAN (Yu et al., 2017) tackles the sequence generation problem with reinforcement learning. Here, we refer to this approach as adversarial REINFORCE. However, the discriminator only measures the quality of whole sequence, and thus the rewards are extremely sparse and the rewards assigned to all the generation steps are all the same. MC search (Yu et al., 2017) is proposed to evaluate the approximate reward at each time

step, but this method suffers from high time complexity. Following this idea, (Li et al., 2017) proposes partial evaluation approach to evaluate the expected reward at each time step. In this paper, we propose the self-critical adversarial REINFORCE algorithm as another way to evaluate the expected reward at each time step. The performance between original WGAN and proposed adversarial REINFORCE is compared in experiment.

3 Proposed Method

The overview of the proposed model is shown in Fig. 2. The model is composed of three components: generator G, discriminator D, and reconstructor R. Both G and R are seq2seq hybrid pointer-generator networks (See et al., 2017) which can decide to copy words from encoder input text via pointing or generate from vocabulary. They both take a word sequence as input and output a sequence of word distributions. Discriminator D, on the other hand, takes a sequence as input and outputs a scalar. The model is learned from a set of documents x and human-written sentences y^{real}.

To train the model, a training document $x = \{x_1, x_2, ..., x_t, ..., x_T\}$, where x_t represents a word, is fed to G, which outputs a sequence of word distributions $G(x) = \{y_1, y_2, ..., y_n, ..., y_N\}$, where y_n is a distribution over all words in the lexicon. Then we randomly sample a word y_n^s from each distribution y_n, and a word sequence $y^s = \{y_1^s, y_2^s, ..., y_N^s\}$ is obtained according to $G(x)$. We feed the sampled word sequence y^s to reconstructor R, which outputs another sequence of word distributions $\hat{x}$. The reconstructor R reconstructs the original text x from y^s. That is, we seek an output of reconstructor $\hat{x}$ that is as close to the original text x as possible; hence the loss for training the reconstructor, R_{loss}, is defined as

$$R_{loss} = \sum_{k=1}^{K} l_s(x, \hat{x}), \tag{1}$$

where the reconstruction loss $l_s(x, \hat{x})$ is the cross-entropy loss computed between the reconstructor output sequence $\hat{x}$ and the source text x, or the negative conditional log-likelihood of source text x given word sequence y^s sampled from $G(x)$. The reconstructor output sequence $\hat{x}$ is teacher-forced by source text x. The subscript $_s$ in $l_s(x, \hat{x})$ indicates that $\hat{x}$ is reconstructed from y^s. K is the number of training documents, and (1) is the summation of the cross-entropy loss over all the training documents x.

In the proposed model, the generator G and reconstructor R form an auto-encoder. However, the reconstructor R does not directly take the generator output distribution $G(x)$ as input [1]. Instead, the reconstructor takes a sampled discrete sequence y^s as input. Due to the non-differentiable property of discrete sequences, we apply the REINFORCE algorithm, which is described in Section 4.

In addition to reconstruction, we need the discriminator D to discriminate between the real sequence y^{real} and the generated sequence y^s to regularize the generated sequence satisfying the summary distribution. D learns to give y^{real} higher scores while giving y^s lower scores. The loss for training the discriminator D is denoted as D_{loss}; this is further described in Section 5.

G learns to minimize the reconstruction loss R_{loss}, while maximizing the loss of the discriminator D by generating a summary sequence y^s that cannot be differentiated by D from the real thing. The loss for the generator G_{loss} is

$$G_{loss} = \alpha R_{loss} - D'_{loss} \tag{2}$$

where D'_{loss} is highly related to D_{loss} – but not necessary the same[2] – and α is a hyper-parameter. After obtaining the optimal generator by minimizing (2), we use it to generate summaries.

Generator G and discriminator D together form a GAN. We use two different adversarial training methods to train D and G; as shown in Fig. 2, these two methods have their own discriminators 1 and 2. Discriminator 1 takes the generator output layer $G(x)$ as input, whereas discriminator 2 takes the sampled discrete word sequence y^s as input. The two methods are described respectively in Sections 5.1 and 5.2.

4 Minimizing Reconstruction Loss

Because discrete sequences are non-differentiable, we use the REINFORCE algorithm. The generator is seen as an agent whose reward given the source text x is $-l_s(x, \hat{x})$. Maximizing the reward is equivalent to minimizing the reconstruction loss R_{loss} in (1). However, the reconstruction

[1] We found that if the reconstructor R directly takes $G(x)$ as input, the generator G learns to put the information about the input text in the distribution of $G(x)$, making it difficult to sample meaningful sentences from $G(x)$.

[2] D'_{loss} has different formulations in different approaches. This will be clear in Sections 5.1 and 5.2.

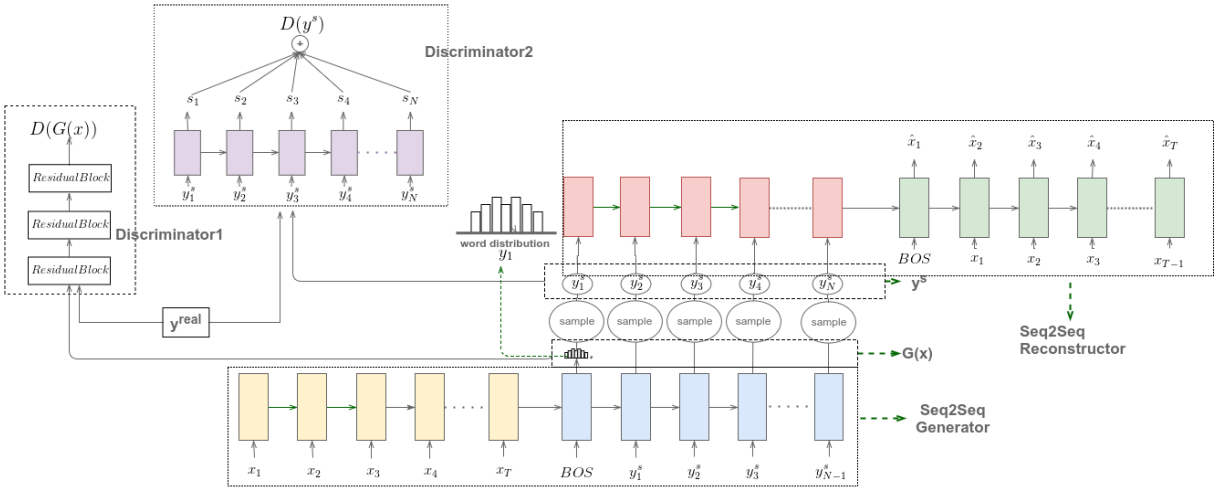

Figure 2: Architecture of proposed model. The generator network and reconstructor network are a seq2seq hybrid pointer-generator network, but for simplicity, we omit the pointer and the attention parts.

loss varies widely from sample to sample, and thus the rewards to the generator are not stable either. Hence we add a baseline to reduce their difference. We apply self-critical sequence training (Rennie et al., 2017); the modified reward $r^R(x, \hat{x})$ from reconstructor R with the baseline for the generator is

$$r^R(x, \hat{x}) = -l_s(x, \hat{x}) - (-l_a(x, \hat{x}) - b) \quad (3)$$

where $-l_a(x, \hat{x}) - b$ is the baseline. $l_a(x, \hat{x})$ is also the same cross-entropy reconstruction loss as $l_s(x, \hat{x})$, except that $\hat{x}$ is obtained from y^a instead of y^s. y^a is a word sequence $\{y_1^a, y_2^a, ..., y_n^a, ..., y_N^a\}$, where y_n^a is selected using the *argmax* function from the output distribution of generator y_n. As in the early training stage, the sequence y^s barely yields higher reward than sequence y^a, to encourage exploration we introduce the second baseline score b, which gradually decreases to zero. Then, the generator is updated using the REINFORCE algorithm with reward $r^R(x, \hat{x})$ to minimize R_{loss}.

5 GAN Training

With adversarial training, the generator learns to produce sentences as similar to the human-written sentences as possible. Here, we conduct experiments on two kinds of methods of language generation with GAN. In Section 5.1 we directly feed the generator output probability distributions to the discriminator and use a Wasserstein GAN (WGAN) with a gradient penalty. In Section 5.2, we explore adversarial REINFORCE, which feeds sampled discrete word sequences to the discriminator and evaluates the quality of the sequence from the discriminator for use as a reward signal to the generator.

5.1 Method 1: Wasserstein GAN

In the lower left of Fig. 2, the discriminator model of this method is shown as **discriminator1** D_1. D_1 is a deep CNN with residual blocks, which takes a sequence of word distributions as input and outputs a score. The discriminator loss D_{loss} is

$$D_{loss} = \frac{1}{K}\sum_{k=1}^{K} D_1(G(x^{(k)})) - \frac{1}{K}\sum_{k=1}^{K} D_1(y^{real(k)}) + \beta_1 \frac{1}{K}\sum_{k=1}^{K}(\Delta_{y^{i(k)}} D_1(y^{i(k)}) - 1)^2,$$

where K denotes the number of training examples in a batch, and k denotes the k-th example. The last term is the gradient penalty (Gulrajani et al., 2017). We interpolate the generator output layer $G(x)$ and the real sample y^{real}, and apply the gradient penalty to the interpolated sequence y^i. β_1 determines the gradient penalty scale. In Equation (2), for WGAN, the generator maximizes D'_{loss}:

$$D'_{loss} = \frac{1}{K}\sum_{k=1}^{K} D_1(G(x^{(k)})). \quad (4)$$

5.2 Method 2: Self-Critic Adversarial REINFORCE

In this section, we describe in detail the proposed adversarial REINFORCE method. The core idea is we use the LSTM discriminator to evaluate the current quality of the generated sequence

$\{y_1^s, y_2^s, ..., y_i^s\}$ at each time step i. The generator knows that compared to the last time step, as the generated sentence either improves or worsens, it can easily find the problematic generation step in a long sequence, and thus fix the problem easily.

5.2.1 Discriminator 2

As shown in Fig. 2, the **discriminator2** D_2 is a unidirectional LSTM network which takes a discrete word sequence as input. At time step i, given input word y_i^s it predicts the current score s_i based on the sequence $\{y_1, y_2, ..., y_i\}$. The score is viewed as the quality of the current sequence. An example of discriminator regularized by weight clipping(Arjovsky et al., 2017) is shown in Fig. 3.

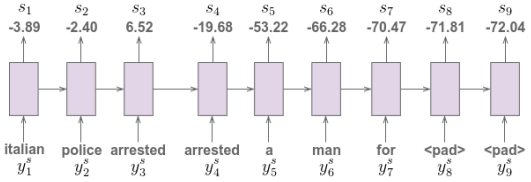

Figure 3: When the second *arrested* appears, as the sentence becomes ungrammatical, the discriminator determines that this example comes from the generator. Hence, after this time-step, it outputs low scores.

In order to compute the discriminator loss D_{loss}, we sum the scores $\{s_1, s_2, ..., s_N\}$ of the whole sequence y^s to yield

$$D_2(y^s) = \frac{1}{N}\sum_{n=1}^{N} s_n.$$

where N denotes the generated sequence length. Then, the loss of discriminator is

$$D_{loss} = \frac{1}{K}\sum_{k=1}^{K} D_2(y^{s(k)}) - \frac{1}{K}\sum_{k=1}^{K} D_2(y^{real(k)}) + \beta_2 \frac{1}{K}\sum_{k=1}^{K}(\Delta_{y^{i(k)}} D_2(y^{i(k)}) - 1)^2,$$

Similar to previous section, the last term is gradient penalty term. With the loss mentioned above, the discriminator attempts to quickly determine whether the current sequence is real or fake. The earlier the timestep discriminator determines whether the current sequence is real or fake, the lower its loss.

5.2.2 Self-Critical Generator

Since we feed a discrete sequence y^s to the discriminator, the gradient from the discriminator cannot directly back-propagate to the generator. Here, we use the policy gradient method. At timestep i, we use the $i-1$ timestep score s_{i-1} from the discriminator as its self-critical baseline. The reward r_i^D evaluates whether the quality of sequence in timestep i is better or worse than that in timestep $i-1$. The generator reward r_i^D from D_2 is

$$r_i^D = \begin{cases} s_i & \text{if i = 1} \\ s_i - s_{i-1} & \text{otherwise.} \end{cases}$$

However, some sentences may be judged as bad sentences at the previous timestep, but at later timesteps judged as good sentences, and vice versa. Hence we use the discounted expected reward d with discount factor γ to calculate the discounted reward d_i at time step i as

$$d_i = \sum_{j=i}^{N} \gamma^{j-i} r_j^D.$$

To maximize the expected discounted reward d_i, the loss of generator is:

$$G'_{loss} = -E_{y_i^s \sim p_G(y_i^s | y_1^s, ..., y_{i-1}^s, x)}[d_i]. \quad (5)$$

We use the likelihood ratio trick to approximate the gradient to minimize (5).

6 Experiment

Our model was evaluated on the English/Chinese Gigaword datasets and CNN/Daily Mail dataset. In Section 6.1,6.2 and 6.4, the experiments were conducted on English Gigaword, while the experiments were conducted on CNN/Daily Mail dataset and Chinese Gigaword dataset respectively in Sections 6.3 and 6.6. We used ROUGE(Lin, 2004) as our evaluation metric.[3] During testing, when using the generator to generate summaries, we used beam search with beam size=5, and we eliminated repetition. We provide the details of the implementation and corpus re-processing respectively in Appendix A and B.

Before jointly training the whole model, we pre-trained the three major components – generator, discriminator, and reconstructor – separately. First, we pre-trained the generator in an unsupervised manner so that the generator would be able to somewhat grasp the semantic meaning of the source text. The details of the pre-training are in Appendix C. We pre-trained the discriminator and reconstructor respectively with the pre-trained generator's output to ensure that these two critic networks provide good feedback to the generator.

[3]We used pyrouge package with option -m -n 2 -w 1.2 to compute ROUGE score for all experiments.

Task	Labeled	Methods	R-1	R-2	R-L
(A)Supervised	3.8M	(A-1)Supervised training on generator	33.19	14.21	30.50
		(A-2) (Rush et al., 2015)†	29.76	11.88	26.96
		(A-3) (Chopra et al., 2016)†	33.78	15.97	31.15
		(A-4) (Zhou et al., 2017)†	**36.15**	**17.54**	**33.63**
(B) Trivial baseline	0	(B-1) Lead-8	21.86	7.66	20.45
(C) Unpaired	0	(C-1) Pre-trained generator	21.26	5.60	18.89
		(C-2) WGAN	28.09	9.88	25.06
		(C-3) **Adversarial REINFORCE**	**28.11**	**9.97**	**25.41**
(D) Semi-supervised	10K	(D-1) WGAN	29.17	10.54	26.72
		(D-2) **Adversarial REINFORCE**	**30.01**	**11.57**	**27.61**
	500K	(D-3)(Miao and Blunsom, 2016)†	30.14	12.05	27.99
		(D-4) WGAN	32.50	13.65	29.67
		(D-5) **Adversarial REINFORCE**	**33.33**	**14.18**	**30.48**
	1M	(D-6)(Miao and Blunsom, 2016)†	31.09	12.79	28.97
		(D-7) WGAN	33.18	14.19	30.69
		(D-8) **Adversarial REINFORCE**	**34.21**	**15.16**	**31.64**
(E) Transfer learning	0	(E-1) Pre-trained generator	21.49	6.28	19.34
		(E-2) **WGAN**	25.11	7.94	23.05
		(E-3) **Adversarial REINFORCE**	**27.15**	**9.09**	**24.11**

Table 1: Average F1 ROUGE scores on English Gigaword. R-1, R-2 and R-L refers to ROUGE 1, ROUGE 2 and ROUGE L respectively. Results marked with † are obtained from corresponding papers. In part (A), the model was trained supervisedly. In row (B-1), we select the article's first eight words as its summary. Part (C) are the results obtained without paired data. In part (D), we trained our model with few labeled data. In part (E), we pre-trained generator on CNN/Diary and used the summaries from CNN/Diary as real data for the discriminator.

6.1 English Gigaword

The English Gigaword is a sentence summarization dataset which contains the first sentence of each article and its corresponding headlines. The preprocessed corpus contains 3.8M training pairs and 400K validation pairs. We trained our model on part of or fully unparalleled data on 3.8M training set. To have fair comparison with previous works, the following experiments were evaluated on the 2K testing set same as (Rush et al., 2015; Miao and Blunsom, 2016). We used the sentences in article headlines as real data for discriminator[4]. As shown in the following experiments, the headlines can even come from another set of documents not related to the training documents.

The results on English Gigaword are shown in Table 1. WGAN and adversarial REINFORCE refer to the adversarial training methods mentioned in Sections 5.1 and 5.2 respectively. Results trained by full labeled data are in part (A). In row (A-1), We trained our generator by supervised training. Compared with the previous work (Zhou et al., 2017), we used simpler model and smaller vocabulary size. We did not try to achieve the state-of-the-art results because the focus of this work is unsupervised learning, and the proposed approach is independent to the summarization models used. In row (B-1), we simply took the first eight words in a document as its summary.

The results for the pre-trained generator with method mentioned in Appendix.C is shown in row (C-1). In part (C), we directly took the sentences in the summaries of Gigaword as the training data of discriminator. Compared with the pre-trained generator and the trivial baseline , the proposed approach (rows (C-2) and (C-3)) showed good improvement. In Fig. 4, we provide a real example. More examples can be found in the Appendix.D.

6.2 Semi-Supervised Learning

In semi-supervised training, generator was pre-trained with few available labeled data. During training, we conducted teacher-forcing with labeled data on generator after several updates without labeled data. With 10K, 500K and 1M la-

[4]Instead of using general sentences as real data for discriminator, we chose sentences from headlines because they have their own unique distribution.

beled data, the teacher-forcing was conducted every 25, 5 and 3 updates without paired data, respectively. In teacher-forcing, given source text as input, the generator was teacher-forced to predict the human-written summary of source text. Teacher-forcing can be regarded as regularization of unpaired training that prevents generator from producing unreasonable summaries of source text. We found that if we teacher-forced generator too frequently, generator would overfit on training data since we only used very few labeled data on semi-supervised training.

The performance of semi-supervised model in English Gigaword regarding available labeled data is shown in Table 1 part (D). We compared our results with (Miao and Blunsom, 2016) which was the previous state-of-the-art method on semi-supervised summarization task under the same amount of labeled data. With both 500K and 1M labeled data, our method performed better. Furthermore, with only 1M labeled data, using adversarial REINFORCE even outperformed supervised training in Table 1 (A-1) with the whole 3.8M labeled data.

Source Text: global one communications will launch its global intranet <unk> -lrb- virtual private networks -rrb- service in new zealand by the end of the year , the new zealand infotech weekly reported monday .	
Ground Truth: international intranet service destined for new zealand	**(A-1)Supervised Result:** global one communications to launch global networks in new zealand
(C-2)WGAN: global communications launch global service in new zealand	**(C-3)Adversarial REINFORCE:** global communications launch global virtual private networks in new zealand
(E-2)WGAN: global one communications will launch its global networks -rrb- service	**(E-3)Adversarial REINFORCE:** global one communications will launch its global virtual networks

Figure 4: Real examples with methods referred in Table 1. The proposed methods generated summaries that grasped the core idea of the articles.

6.3 CNN/Daily Mail dataset

The CNN/Daily Mail dataset is a long text summarization dataset which is composed of news articles paired with summaries. We evaluated our model on this dataset because it's a popular benchmark dataset, and we want to know whether the proposed model works on long input and long output sequences. The details of corpus preprocessing can be found in Appendix.B . In unpaired training, to prevent the model from directly matching the input articles to its corresponding summaries, we split the training pairs into two equal sets, one set only supplied articles and the other set only supplied summaries.

The results are shown in Table 2. For supervised approaches in part (A), although our seq2seq model was similar to (See et al., 2017), due to the smaller vocabulary size (we didn't tackle out-of-vocabulary words), simpler model architecture, shorter output length of generated summaries, there was a performance gap between our model and the scores reported in (See et al., 2017). Compared to the lead-3 baseline in part (B) which took the first three sentences of articles as summaries, the seq2seq models fell behind. That was because news writers often put the most important information in the first few sentences, and thus even the best abstractive summarization model only slightly beat the lead-3 baseline on ROUGE scores. However, during pre-training or training we didn't make assumption that the most important sentences are in first few sentences.

We observed that our unpaired model yielded decent ROUGE-1 score, but it yielded lower ROUGE-2 and ROUGE-L score. That was probably because the length of our generated sequence was shorter than ground truth, and our vocabulary size was small. Another reason was that the generator was good at selecting the most important words from the articles, but sometimes failed to combine them into reasonable sentences because it's still difficult for GAN to generate long sequence. In addition, since the reconstructor only evaluated the reconstruction loss of whole sequence, as the generated sequence became long, the reconstruction reward for generator became extremely sparse. However, compared to pre-trained generator (rows (C-2), (C-3) v.s. (C-1)), our model still enhanced the ROUGE score. An real example of generated summary can be found at Appendix.D Fig.11 .

6.4 Transfer Learning

The experiments conducted up to this point required headlines unpaired to the documents but in the same domain to train discriminator. In this subsection, we generated the summaries from English Gigaword (target domain), but the summaries for discriminator were from CNN/Daily Mail dataset (source domain).

The results of transfer learning are shown in Table. 1 part (E). Table 1 (E-1) is the result of pre-

Methods		R-1	R-2	R-L
(A)Supervised	(A-1)Supervised training on our generator	38.89	13.74	29.42
	(A-2) (See et al., 2017)†	39.53	17.28	36.38
(B)Lead-3 baseline (See et al., 2017)†		40.34	17.70	36.57
(C) Unpaired	(C-1) Pre-trained generator	29.86	5.14	14.66
	(C-2) WGAN	35.14	**9.43**	**21.04**
	(C-3) **Adversarial REINFORCE**	**35.51**	9.38	20.98

Table 2: F1 ROUGE scores on CNN/Diary Mail dataset. In row (B), the first three sentences were taken as summaries. Part (C) are the results obtained without paired data. The results with symbol † are directly obtained from corresponding papers.

Methods		R-1	R-2	R-L
(A) Training with paired data (supervised)		49.62	34.10	46.42
(B)Lead-15 baseline		30.08	18.24	27.74
(C) Unpaired	(C-1) Pre-trained generator	28.36	16.73	26.48
	(C-2) WGAN	38.15	24.60	35.27
	(C-3) **Adversarial REINFORCE**	**41.25**	**26.54**	**37.76**

Table 3: F1 ROUGE scores on Chinese Gigaword. In row (B), we selected the article's first fifteen words as its summary. Part (C) are the results obtained without paired data.

trained generator and the poor pre-training result indicates that the data distributions of two datasets are quite different. We find that using sentences from another dataset yields lower ROUGE scores on the target testing set (parts (E) v.s. (C)) due to the mismatch word distributions between the summaries of the source and target domains. However, the discriminator still regularizes the generated word sequence. After unpaired training, the model enhanced the ROUGE scores of the pre-trained model (rows (E-2), (E-3) v.s. (E-1)) and it also surpassed the trivial baselines in part (B).

6.5 GAN Training

In this section, we discuss the performance of two GAN training methods. As shown in the Table 1, in English Gigaword, our proposed adversarial REINFORCE method performed better than WGAN. However, in Table 2, our proposed method slightly outperformed by WGAN. In addition, we find that when training with WGAN, convergence is faster. Because WGAN directly evaluates the distance between the continuous distribution from generator and the discrete distribution from real data, the distribution was sharpened at an early stage in training. This caused generator to converge to a relatively poor place. On the other hand, when training with REINFORCE, generator keeps seeking the network parameters that can better fool discriminator. We believe that training GAN on language generation with this method is worth exploring.

6.6 Chinese Gigaword

The Chinese Gigaword is a long text summarization dataset composed of paired headlines and news. Unlike the input news in English Gigaword, the news in Chinese Gigaword consists of several sentences. The results are shown in Table 3. Row (A) lists the results using 1.1M document-summary pairs to directly train the generator without the reconstructor and discriminator: this is the upper bound of the proposed approach. In row (B), we simply took the first fifteen words in a document as its summary. The number of words was chosen to optimize the evaluation metrics. Part (C) are the results obtained in the scenario without paired data. The discriminator took the summaries in the training set as real data. We show the results of the pre-trained generator in row (C-1); rows (C-2) and (C-3) are the results for the two GAN training methods respectively. We find that despite the performance gap between the unpaired and supervised methods (rows (C-2), (C-3) v.s. (A)), the proposed method yielded much better performance than the trivial baselines (rows (C-2), (C-3) v.s. (B)).

7 Conclusion and Future Work

Using GAN, we propose a model that encodes text as a human-readable summary, learned without document-summary pairs. In future work, we hope to use extra discriminators to control the style and sentiment of the generated summaries.

References

Martin Arjovsky, Soumith Chintala, and Lon Bottou. 2017. Wasserstein gan. *arXiv preprint arXiv:1701.07875*.

Dzmitry Bahdanau, Philemon Brakel, Kelvin Xu, Anirudh Goyal, Ryan Lowe, Joelle Pineau, Aaron Courville, and Yoshua Bengio. 2016. An actor-critic algorithm for sequence prediction. *arXiv preprint arXiv:1607.07086*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. *ICLR*.

Sumit Chopra, Michael Auli, and Alexander M. Rush. 2016. Abstractive sentence summarization with attentive recurrent neural networks. *HLT-NAAC*.

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial networks. *arXiv preprint arXiv:1406.2661*.

Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. 2017. Improved training of wasserstein gans. *arXiv preprint arXiv:1704.00028*.

Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S. Zemel, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. 2015. Skip-thought vectors. *NIPS*.

Jiwei Li, Minh-Thang Luong, and Dan Jurafsky. 2015. A hierarchical neural autoencoder for paragraphs and documents. *ACL*.

Jiwei Li, Will Monroe, Tianlin Shi, Sbastien Jean, Alan Ritter, and Dan Jurafsky. 2017. Adversarial learning for neural dialogue generation. *arXiv preprint arXiv:1701.06547*.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. *In Text summarization branches out: ACL workshop*.

Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. 2015. Adversarial autoencoders. *arXiv preprint arXiv:1511.05644*.

Yishu Miao and Phil Blunsom. 2016. Language as a latent variable: Discrete generative models for sentence compression. *EMNLP*.

Ramesh Nallapati, Bowen Zhou, Cicero Nogueira dos santos, Caglar Gulcehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. *EMNLP*.

Romain Paulus, Caiming Xiong, and Richard Socher. 2017. A deep reinforced model for abstractive summarization. *arXiv preprint arXiv:1705.04304*.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2016. Sequence level training with recurrent neural networks. *ICLR*.

Steven J. Rennie, Etienne Marcheret, Youssef Mroueh, Jarret Ross, and Vaibhava Goel. 2017. Self-critical sequence training for image captioning. *CVPR*.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. *EMNLP*.

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

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. *NIPS*.

Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. 2016. Modeling coverage for neural machine translation. *ACL*.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. *NIPS*.

Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. 2017. Seqgan: Sequence generative adversarial nets with policy gradient. *AAAI*.

Qingyu Zhou, Nan Yang, Furu Wei, and Ming Zhou. 2017. Selective encoding for abstractive sentence summarization. *ACL*.

Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. 2017. Unpaired image-to-image translation using cycle-consistent adversarial networks. *arXiv preprint arXiv:1703.10593*.