Headline Generation via Adversarial Training

Daniel Stancl
ucabds7@ucl.ac.uk

Guoliang He
ucabggh@ucl.ac.uk

Dorota Jagnesakova
ucabdj1@ucl.ac.uk

Zakhar Borok
zcabzbo@ucl.ac.uk

Abstract

Originally developed for computer vision, Generative Advarsarial Nets (GANs) have achieved great success in producing real-valued data. While these nets bypass a lot of the problems associated with more widely used models based on maximum-likelihood approaches, it remains a challenge to successfully train them for tasks involving discrete outputs, such as text summarization. In this paper, we propose an adversarial training approach for an abstractive text summarization task: generating headlines for WikiHow articles. We make use of two models participating in a contest to outperform each other — an encoder-decoder generator and a discriminator. We train these with respect to two different loss functions and evaluate the resulting model's performance using ROUGE metrics. In particular, we see an improvement by 25.9 % in ROUGE-1 and 42.6 % in ROUGE-2 of our proposed adversarially-trained model as compared to baseline seq2seq models.

1 Introduction

In today's era of ever-growing data, automatic text summarization is becoming an essential tool in condensing large amounts of information. Its goal is to understand main topics of documents and return concise summaries containing their most important points under a length limit. Both extractive and abstractive machine learning methods allow for fast utilization of relevant information from such documents, reducing the expensive and time-consuming nature of this problem.

In this work, we focus on summarizing shorter texts using headline generation. While a lot of text summarization uses extraction, i.e. phrases directly pulled from the text, this method has been shown to be non-ideal for generating headline styled summaries consisting of a single sentence or less than 10 words (Banko et al., 2000). Abstractive methods present an alternative; they are verbally innovative and build a semantic representation of the text, often resulting in novel words paraphrasing the text. Although sequence-to-sequence models have been repeatedly shown to be successful in abstractive summarization (Nallapati et al., 2016), they still face some issues, such as *exposure bias*. This refers to a train-test discrepancy occurring due to the fact that the model uses ground-truth context only during training, but when testing it is generating words conditioned on generated context that was potentially not seen before. There have been efforts to alleviate this problem, but none have been able to eradicate it (Schmidt, 2019).

For this reason, we take a different approach and use a generative adversarial network (GAN), which bypasses the problem of exposure bias by jointly training a discriminative and a generative network. The discriminative network is trained to distinguish generated headlines from real headlines using an adversarial loss function. The generative network uses the discriminator's evaluation to improve its performance, and so it is not directly exposed to the ground-truth data.

One sentence summaries have multiple applications in the news industry, search results of search engines and compressing web pages for portable devices. A lot of work so far has focused on news articles, using various linguistic datasets from major journalism companies. However, because of the specific writing style used in journalism, it is unclear whether or not models trained on these datasets can be generalized to other tasks. To avoid this pitfall of the news data, we use a large dataset from the WikiHow knowledge base instead, introduced by (Koupaee and Wang, 2018). WikiHow consists of how-to articles edited by readers of the website, which allows for a wider

range of writing styles.

In this work, we propose a form of adversarial training for generating headlines. We utilize two different loss functions that stipulate generator-produced headlines that match the target headlines. We use ROUGE scores to evaluate the performance of the chosen models (Lin and Och, 2004). Headlines produced by our baseline seq2seq models (pre-trained by minimizing cross-entropy loss) reach overlap with the references on the level of unigrams and bigrams at 15.40 and 4.35 respectively. On the other hand, the model trained using our proposed framework attained ROUGE-1 score of 19.38 and ROUGE-2 score of 6.21, which is an improvement by 25.9 % and 42.6 %. On average, the enhanced quality of produced summaries can also be observed by manual inspection.

2 Related Work

GANs were first introduced by (Goodfellow et al., 2014) in 2014. Since then, they have been studied and used for various tasks, achieving most success in computer vision, e.g image generation (Radford et al., 2015) or image synthesis (Zhang et al., 2017a). GANs have also been successfully used for dialogue generation (Li et al., 2017), machine translation (Yang et al., 2017) or emotion conversion as well as emotion understanding (Han et al., 2018).

Unlike computer vision tasks, the generator output for text generation is discrete (rather than a probability distribution), and therefore not differentiable, which introduces a problem in error back-propagation from the discriminator to the generator. To address this problem in text generation, (Yu et al., 2016) proposed a network called SeqGAN, which uses policy gradient reinforcement learning and treats sentence generation as a sequential decision-making process. Their work shows notable results in poem, music and speech language generation. Later, LeakGAN introduced by (Guo et al., 2017) used a similar framework for longer text generation, leveraging intermediate information about text structure during the generative process, thus eliminating the problem of sparse reward for longer text.

Both SeqGAN and LeakGAN face two problems. First, *mode collapse*, caused by the discriminator giving a high evaluation to certain tokens that then receive a higher estimated probability of being sampled. This results in the generator producing observations from a narrow part of the target distribution, causing lower diversity of the outputs (Lu et al., 2018). Second, *gradient vanishing*, caused by the discriminator becoming too strong and thus halting the generator training (Arjovsky and Bottou, 2017).

TextGAN introduced by (Zhang et al., 2017b) aims to address the above problems by using a long short-term memory network (LSTM) as generator, a convolutional network as discriminator, and matching the high-dimensional latent feature distributions of real and synthetic sentences using a kernelized discrepancy metric. The authors of this work used four different loss functions for their generator and discriminator, some of which (e.g. Maximum Mean Discrepancy) helped alleviate mode collapsing.

In our work, we follow the methods used by the TextGAN authors and utilize two distinct loss functions. Furthermore, we try to avoid the gradient vanishing problem by using gradient clipping during the optimization process of the generator encoder and a low learning rate for the discriminator during adversarial training.

3 Methods

In this section, first we introduce Generative Adversarial Networks (GANs). Subsequently, we explain the concept of Summarization GANs and their optimization as it relates to our problem. Lastly, we elaborate on the use of a generator and a discriminator in our work.

3.1 Generative Adversarial Networks

(Goodfellow et al., 2014) came up with a new approach in fitting generative models via simultaneously optimizing the problem

$$\mathcal{L}_{GAN} = \mathbb{E}_{\mathbf{x}\sim p_{\mathbf{x}}} \log D(\mathbf{x}) + \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}} \log\left[1 - D(G(\mathbf{z}))\right], \quad (1)$$

with respect to D and G. The discriminator, D, is trained to maximize the probability of correctly assigning a label to the data points. Hence $\mathcal{L}$ is iteratively maximized with respect to D. On the other hand, G is trained to minimize $\mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}} \log\left[1 - D(G(\mathbf{z}))\right]$, that is to fool the discriminator. These settings represent a well-known scenario of a mini-max game of two players.

Later, it was shown by (Arjovsky and Bottou, 2017) that a direct application of (1) during the op-

timization process suffers from very poor learning due to gradient vanishing. Since then, the update step for the generator has been adjusted to

$$\nabla_{\mathbf{z}} \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}} \left[-\log D(G(\mathbf{z})) \right]. \quad (2)$$

3.2 Summarization GAN

While the approach above is well applicable in computer vision, where an image is generated based on input noise $\mathbf{z}$, this cannot be easily utilized in the area of text summarization. This is because the output of the model must not only look real, but also be closely related to carry information from the underlying paragraph or article.

Therefore, we adopt a similar technique and utilize two distinct loss functions for the discriminator and generator as in (Zhang et al., 2017b). Specifically, we estimate the models' parameters w.r.t. to the objecive

$$\mathcal{L}_D = \mathcal{L}_{GAN}, \quad (3)$$

$$\mathcal{L}_G = -\mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}} \left[\log D(G(\mathbf{z})) \right] \cdot \mathcal{L}_{CE} G(\mathbf{z}), \quad (4)$$

where $\mathcal{L}_{CE}$ denotes cross-entropy loss, computed over the whole output sequence, which is commonly used for optimizing various language models. Our objective is again simultaneously maximized w.r.t. D and minimized w.r.t. G. Furthermore, our formulation of the generator's loss function $\mathcal{L}_G$ should stipulate the model to output summaries or headlines which resemble the real ones and at the same time contain the relevant information.

3.3 Generator

The base model for out generator was inspired by (Nallapati et al., 2016). The model generally consists of two parts representing by an Encoder and a Decoder. Full architecture of our implementation is illustrated in Figure 1.

Encoder - The encoder reads an input sequentially and feeds it into a pre-trained embedding layer projecting our padded discrete input sequence of indexed words, $x_1, \ldots, x_T$, into an embedded representation $\mathbf{x}_1, \ldots, \mathbf{x}_T$.

This layer is then followed by a bidirectional recurrent neural network (later referred to as RNN) (Schuster and Paliwal, 1997) where a single RNN maps embedded input sequence into a fixed-size sequence of hidden state, $\mathbf{h}_1, \ldots, \mathbf{h}_T$ (we can call them annotations), which are iteratively updated according to equation (5)

$$\mathbf{h}_t = f\left(\mathbf{h}_{t-1}, \mathbf{x}_t\right), \quad (5)$$

where f is a non-linear activation function.

We decide to implement this RNN with Gated recurrent units (GRUs thereafter) used as the non-linear activation function f, introduced in (Cho et al., 2014). Also, we use clipping gradient during the optimization process, which should avert problems with the vanishing gradient during back-propagation through time.

Decoder - The decoder of our seq2seq model is an unidirectional RNN (again with a GRU mechanism) that is trained to generate an output sequence - headline or summary. The hidden states of the decoder, $\mathbf{s}_1, \ldots, \mathbf{s}_K$ (K denotes the length of an output sequence), are iteratively mapped to the output token distribution via the softmax activation function conditioned on the previous output elements and the vector $\mathbf{x}$ consisting of the input sequence $x_1, \ldots, x_T$

$$p(\mathbf{y}_t | \mathbf{y}_{t-1}, \ldots, \mathbf{y}_1, \mathbf{x}) = \text{softmax}\left(\mathbf{W}\mathbf{s}_t + \mathbf{b}'\right), \quad (6)$$

where $\mathbf{W}$ and $\mathbf{b}'$ are again learnable parameters and $\mathbf{s}_t$ denotes the very last decoder hidden state preceding the softmax activation function projecting output into the space of probabilites of particular words in a dictionary. The decoder hidden states $\mathbf{s}_t,\ t = 1, ..., K$ are updated conditionally on the previous hidden state $\mathbf{s}_{t-1}$, the last output token y_{t-1} and the convex vector, $\mathbf{c}_i$

$$\mathbf{s}_t = g\left(\mathbf{s}_{t-1}, y_{t-1}, \mathbf{c}_i\right), \quad (7)$$

where g is a non-linear activation function represented by GRUs.

Here, the attention mechanism (Bahdanau et al., 2014) is leveraged to improve the decoder performance. The attention makes a use of context vectors $\mathbf{c}_i,\ i = 1, ..., K$ defined as

$$\mathbf{c}_i = \sum_{j=1}^{T} \alpha_{ij} \mathbf{h}_j. \quad (8)$$

Weights α_{ij} are computed as

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T} \exp(e_{ik})}, \quad (9)$$

where $e_{ij},\ j = 1, \ldots, T$ is an output of the alignment model (which is a trainable feedforward neural network)

$$e_{ij} = a\left(\mathbf{s}_{t-1}, \mathbf{h}_j\right) \quad (10)$$

that looks for how well the decoder hidden state $\mathbf{s}_{t-1}$ and the encoder alignment around the position j matches.

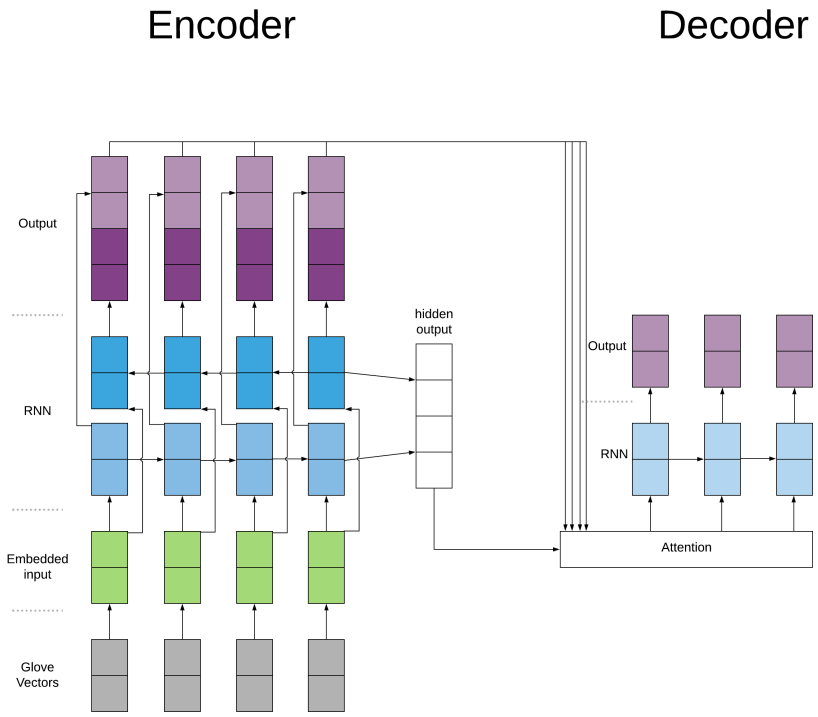

Figure 1: Scheme of Generator

3.4 Discriminator

The discriminator is a variation of the convolutional neural network from (Kim, 2014) and (Collobert et al., 2011). Full architecture of our implementation is illustrated in Figure 2.

The discriminator's input is represented by a vector of indexed words $(x_1, x_2, \ldots, x_n)$ which is directly fed into the pre-trained embedding layer that provides a $n \times k$ representation of target summary

$$\mathbf{x}_{1:n} = \mathbf{x}_1 \oplus \mathbf{x}_2 \oplus \cdots \oplus \mathbf{x}_n, \tag{11}$$

where $\mathbf{x}_i \in \mathbb{R}^k, k \in \mathbb{N}, \ i = 1, \ldots, n$, denotes a k-dimensional embedded vector corresponding to the word in position i in a given sentence of length n. The symbol $\oplus$ represents a concatenation and all the sentences are padded to the same length.

Subsequently, three convolutional layers computed in parallel with concatenated output follow. A convolution operation can be thought of as an application of filter $\mathbf{w} \in \mathbb{R}^{s \times k}$ on a span of s words from the data to generate a new feature c_i. In our setting, with the rectifier linear unit is used as a non-linear activation function, c_i can be mathematically expressed as

$$c_i = \max\left(\mathbf{w} \cdot \mathbf{x}_{i+i+s-i} + b,\ 0\right) \tag{12}$$

where $i = 1, 2, \ldots, n - s + 1$, b denotes a bias term, and the symbol $\cdot$ represents the dot product. This filter $\mathbf{w}$ is then applied over all windows of s words yielding a new feature map

$$\mathbf{c} = (c_1, c_2, \ldots, c_{n-s+1}), \tag{13}$$

that is subsequently fed into a max pooling layer spanning over the whole feature map

$$\hat{c} = \max(\mathbf{c}). \tag{14}$$

This operation is done in order to identify the most important features, i.e. the ones exhibiting the highest values. Max pooling also bypasses the issue of sequences having different lengths.

Subsequently, since many kernel filters and different filter windows are used, all the derived features are connected together to create a single vector

$$\hat{\mathbf{c}} = (\hat{c}_1, \hat{c}_2, ..., \hat{c}_m), \tag{15}$$

where $m \in \mathbb{N}$ equals the product of the number of kernel filters and the number of different filter sizes used.

These features are then passed through the fully-connected layer followed by a sigmoid activation function to compute probability of the

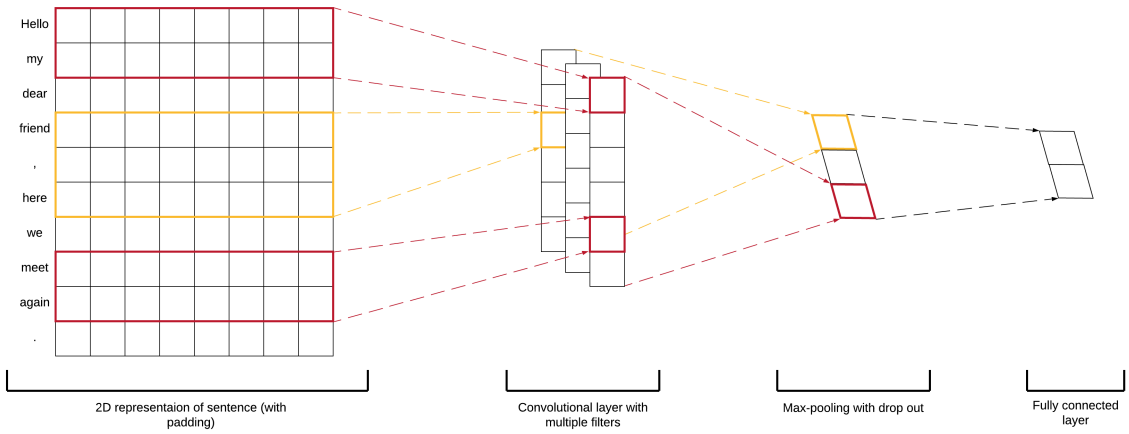

Figure 2: Scheme of Discriminator, inspired by (Kim, 2014)

headline being real or fake, i.e.

$$p\left(y = \text{real}|\mathbf{x}_{1:n}\right) = \sigma\left(\mathbf{u}^T\hat{\mathbf{c}} + b\right), \quad (16)$$

$$p\left(y = \text{fake}|\mathbf{x}_{1:n}\right) = 1 - p\left(y = \text{real}|\mathbf{x}_{1:n}\right), \quad (17)$$

the vector $\mathbf{u}$ and bias term b are learnable parameters.

4 Experiments

Dataset - WikiHow Dataset [1] is a publicly available dataset consisting of more than 230,000 articles from the WikiHow web-page, covering a wide range of topics and therefore providing a lot of diversity in terms of writing styles. Each WikiHow article is split into multiple paragraphs and each paragraph has its own headline summarizing it.

We use a separated dataset with the aim of training a model to generate the paragraph headlines, i.e. we treat each paragraph as a separate example with its own headline. We only use examples with paragraphs of length less than 200 words (due to memory constraints) and more than 10 words (shorter paragraphs are likely flawed) and headlines between 2 and 75 words. At the same time, we only use examples that have a headline length to paragraph length ratio of less than 0.75. From this filtered dataset, we randomly sample 150,000 examples which are used for training our models. After all the filtering described above, the average length of a paragraph in our data is 87.47 words and the average summary length is 9.45 words.

Padding - We start each paragraph and its headline using the token `sos`, and end using the token `eos`. In any batch, all sentences are padded to have the same length.

Word embedding - For text representation, we use GloVe [2], pre-trained word embeddings introduced by (Pennington et al., 2014). The embedding dimension chosen is 200.

Pre-training - First, our generator is pretrained by minimizing cross-entropy until convergence on the training data, which usually requires between 20 and 25 epochs. Three different variants of encoder-decoder with a hidden layer of size 128, 256 or 512, respectively, in both components are trained. Batch size of 32 examples in total with ADAM optimizer and its step size $3 \cdot 10^{-4}$ are used. All variations of the models are penalized with L_2 penalty $\lambda = 1 \cdot 10^{-4}$ and dropout probability of $p = 0.1$. Dropout layer is implemented within the RNN units.

After the generator is pre-trained, we use training examples from the other part of our dataset, dedicated to the pre-training of the discriminator, to produce "fake" headlines. The discriminator is trained by minimizing binary cross-entropy loss. The CNN discriminator uses kernel filters of size $(3, 4, 5)$ and 20 different models are fitted with a different number of kernels and varying values of dropout probability. Specifically, kernel number is drawn from $[10, 20, 30, 50, 100]$ and dropout probabilities are drawn from $[0.0, 0.2, 0.3, 0.5]$. Dropout layer is implemented as a penultimate layer preceding the very last fully-connected one. Again, the model is trained with batches of size 32 and ADAM optimizer with learning size $5 \cdot 10^{-4}$ is used. In this case, L_2 penalty is not applied.

Based on the pre-training results, we chose a generator with a hidden RNN layer of size 512 and a CNN discriminator with 50 kernel filters and

[1] https://github.com/mahnazkoupaee/WikiHow-Dataset

[2] https://nlp.stanford.edu/projects/glove

dropout probability 0.5 for adversarial training.

Adversarial training - After the initial pre-training, our encoder-decoder model and discriminator are trained iteratively w.r.t. objective (3) and (4). Adversarial training is an esoteric art that requires using a number of unconventional methods to make training work. Since the generator has significantly more parameters than the discriminator, optimizing it is more complicated. Therefore, we conduct two update steps for the generator for every one optimization step of the discriminator. As this penalization of the discriminator ends up being insufficient and leads to an imbalance, we perform an update step for the discriminator only with a probability of 0.15.

The Generator and discriminator are trained on batches of size 32 with SGD (training looks more stable with SGD instead of ADAM) and ADAM optimizer, and learning rates $1 \cdot 10^{-3}$ and $5 \cdot 10^{-4}$, respectively (learning rate for the discriminator is chosen to be low intentionally in order to avoid gradient vanishing, i.e. an imbalance caused by the discriminator becoming too strong). Moreover, we apply dropout with $p = 0.5$ $L2$ regularization on a CNN discriminator with a considerably high value of parameter $\lambda = 0.1$. Regularization used for the generator remains unchanged from pre-training.

Testing metrics - We use ROUGE introduced by (Lin and Och, 2004) as a standardized metric for evaluation of the model performance on the summarization task. More specifically, we rely on ROUGE-1 and ROUGE-2 considering the overlap of unigrams and bigrams, respectively, between target summary and the output summary produced by our model. Furthermore, we also use ROUGE-L based on Longest Common Subsequence statistics.

Lastly, the models are also judged based on the length of the produced output. For randomly picked examples, we give a short comment reflecting on the quality of the output.

5 Results and Discussion

The results we achieve are far from stellar and definitely not comparable to the current state-of-the-art research. However they are satisfactory considering the variety of constrains we face:

1. Computation resources and limited time frame meant we had to use smaller smaller models in comparison with the contemporary papers.
2. We only used a small subset of the data to hasten the generalization of our models.

The performance of individual models measured by ROUGE metrics is summarized in Table 1.

Model	ROUGE-1	ROUGE-2	ROUGE-L
Base-128	14.197	3.168	14.210
Base-256	15.283	3.960	15.312
Base-512	15.399	4.353	15.385
SumGAN	**19.382**	**6.209**	**19.343**

Table 1: Models' performance on the training data measured by ROUGE metrics

The performance of our baseline model is improving with the hidden size of RNNs in the encoder and the decoder. An increase of hidden size from 128 to 256 pushes ROUGE-1 and ROUGE-L by around 1.1 and ROUGE-2 is by about 0.8. When we raise the complexity of our model further, a rise in the values of ROUGE-1 and ROUGE-L is rather negligible, however ROUGE-2 ascends from 3.96 to 4.35, which is undoubtedly a considerable improvement.

During adversarial training we witness an unprecedented increment - ROUGE-1 and ROUGE-L rise by almost 4 from roughly 15.40 and 15.39 to 19.38 and 19.34, respectively. Substantial improvement is also measured by ROUGE-2 as this metric rises from 4.35 to 6.21 compared to the baseline model with a hidden size of 512.

A similar improving trend is also observable when we compare our models based on the average length of headlines they produce. While our baseline models output headlines of mean length 4.26, 4.31 and 4.41 words respectively, the SumGAN's output headlines of mean length 5.12 words. Though this is an improvement, it is important to emphasize these results are not completely satisfactory, as the mean length of headlines in the data is 9.45 words. The poor performance of our model is likely due to its low complexity as compared to the models used in contemporary research and the fact that a small data set is used for training.

Human inspection of the generated headlines can also confirm the improvement in the quality by the SumGAN model in comparison with our baselines. Still, some headlines provided by either

model seem nonsensical. This might be partially caused by some flawed examples in the data.[3]

In conclusion, we saw a significant improvement brought by adversarial training, though it remains a question whether such performance boost would be satisfactory if our training method was put to use.

In spite of the large regularization of the discriminator, we failed to keep a balanced training of the generator and the discriminator as the discriminator strongly outperforms the generator after a few iterations, and its accuracy rises to almost 100 %. Hand in hand with this flaw, the generator's performance measured by ROUGE metrics seems to be fluctuating for the majority of training time, and we suspect this flattening learning curve may stem from the strongly outperforming discriminator.

6 Conclusion

We have introduced a valid adversarially-trained model which outperformed traditional seq2seq baseline models on all ROUGE metrics inspected. We have discussed the necessary techniques to train our model and the issues encountered when optimizing its performance. Manually inspecting the generated headlines, we observed that while it is not perfect, it is capable of producing realistic headlines.

In future work, we would use a more appropriate dataset with consistently logical headlines at every instance. Our limited computational resources restricted us in terms of the complexity of the models we used, and so it would be interesting to see the results of our proposed network when applied to a larger dataset, potentially using larger hidden sizes and more parameters.

7 Appendix

Training time - During training, we usually rely on Nvidia Tesla P100 16 GB sourced by *Google Colaboratory*. Given this hardware, completing one epoch of generator training takes roughly between 20 and 35 minutes depending on the model size while training of discriminator lasts just a fraction of a time of the generator.

Github repository - The GitHub repository containing all the scripts and results can be found via github.com/stancld/GeneratingHeadline_GANs.

[3] If one reads through the examples, sometimes the headlines does not seem perfectly related to the paragraph. However, cleaning our data set would be an unfeasible feat.

References

Martín Arjovsky and Léon Bottou. 2017. Towards principled methods for training generative adversarial networks. *ArXiv*, abs/1701.04862.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Michele Banko, Vibhu O. Mittal, and Michael J. Witbrock. 2000. Headline generation based on statistical translation. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*, pages 318–325, Hong Kong. Association for Computational Linguistics.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of machine learning research*, 12(Aug):2493–2537.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc.

Jiaxian Guo, Sidi Lu, Han Cai, Weinan Zhang, Yong Yu, and Jun Wang. 2017. Long text generation via adversarial training with leaked information.

Jing Han, Zixing Zhang, Nicholas Cummins, and Bjorn Schuller. 2018. Adversarial training in affective computing and sentiment analysis: Recent advances and perspectives.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv*, abs/408.5882v2.

Mahnaz Koupaee and William Wang. 2018. Wikihow: A large scale text summarization dataset.

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

Chin-Yew Lin and FJ Och. 2004. Looking for a few good metrics: Rouge and its evaluation. In *Ntcir Workshop*.

Sidi Lu, Yaoming Zhu, Weinan Zhang, Jun Wang, and Yong Yu. 2018. Neural text generation: Past, present and beyond. *ArXiv*, abs/1803.07133.

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Çağlar GuÌ‡lçehre, 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. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Alec Radford, Luke Metz, and Soumith Chintala. 2015. Unsupervised representation learning with deep convolutional generative adversarial networks.

Florian Schmidt. 2019. Generalization in generation: A closer look at exposure bias. In *Proceedings of the 3rd Workshop on Neural Generation and Translation*, pages 157–167, Hong Kong. Association for Computational Linguistics.

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE transactions on Signal Processing*, 45(11):2673–2681.

Zhen Yang, Wei Chen, Feng Wang, and Bo Xu. 2017. Improving neural machine translation with conditional sequence generative adversarial nets.

Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. 2016. Seqgan: Sequence generative adversarial nets with policy gradient. *ArXiv*, abs/1609.05473.

Han Zhang, Tao Xu, and Hongsheng Li. 2017a. Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. pages 5908–5916.

Yizhe Zhang, Zhe Gan, Kai Fan, Zhi Chen, Ricardo Henao, Dinghan Shen, and Lawrence Carin. 2017b. Adversarial feature matching for text generation. In *ICML*.