A Reinforced Improved Attention Model for Abstractive Text Summarization

Yu Chang Hang Lei Xiaoyu Li Yiming Huang
School of Information and Software Engineering
University of Electronic Science and Technology of China
Chengdu, China
realchangyu@gmail.com {hlei, xiaoyuuestc}@uestc.edu.cn yiminghwang@gmail.com

Abstract

In recent times, RNN-based sequence-to-sequence attentional models have achieved good performance on abstractive summarization. However, numerous problems regarding repetition, incoherence, and exposure bias are encountered when applying these models. In this work, we propose a novel architecture that augments the standard sequence-to-sequence attentional model and a new training method combining reinforcement learning. We evaluate our proposed method on the CNN/Daily Mail dataset. The empirical results demonstrate the superiority of our proposed method in the abstractive summarization.

1 Introduction

Abstractive text summarization is an important aspect of natural language processing (NLP), which requires the machine to automatically generate a paragraph of general content (Wang et al. 2018), such as news title summarization (Kraaij, Spitters, and Hulth 2002) and abstract summarization (Barzilay and McKeown 2005), after reading an article. Nevertheless, compared with other NLP tasks, automatic summarization exists numerous problems. For example, unlike machine translation tasks where input and output sequences often share similar lengths, summarization tasks are more likely to have input and output sequences greatly imbalanced. There are two methods to summarization: extractive and abstractive. Whereas the extraction method collects abstracts only from paragraphs (usually entire sentences) that are extracted directly from the source text (Neto, Freitas, and Kaestner 2002; Dorr, Zajic, and Schwartz 2003; Martins and Smith 2009; Berg-Kirkpatrick, Gillick, and Klein 2011; Nallapati, Zhai, and Zhou 2017), the abstract method may generate new words and phrases that are not present in the source text (Ranzato et al. 2015; Nallapati et al. 2016; See, Liu, and Manning 2017; Zhou et al. 2018; Gehrmann, Deng, and Rush 2018).

With the success of the sequence-to-sequence (seq2seq) mode (Bahdanau, Cho, and Bengio 2014; Sutskever, Vinyals, and Le 2014), it is possible to use recurrent neural networks (RNN) to read articles and generate topics. However, there are some problems with the conventional seq2seq model. First, before the start of the summary generation task, a fixed size vocabulary needs to be established, and each word of the text is replaced by its index in the vocabulary when the text is processed. However, most source articles will have out-of-vocabulary (OOV) words that are not in the vocabulary list, such as names of people, place names, scores, etc. When these words are encountered in the conventional seq2seq model, they can only be regarded as unrecognized words (UNK) (Gulcehre et al. 2016), so the output will often appear as well. Second, when generating a summary of multiple sentences, it is common to generate repeated words or sentences (See, Liu, and Manning 2017). In addition, exposure bias is problem in sequence gen-

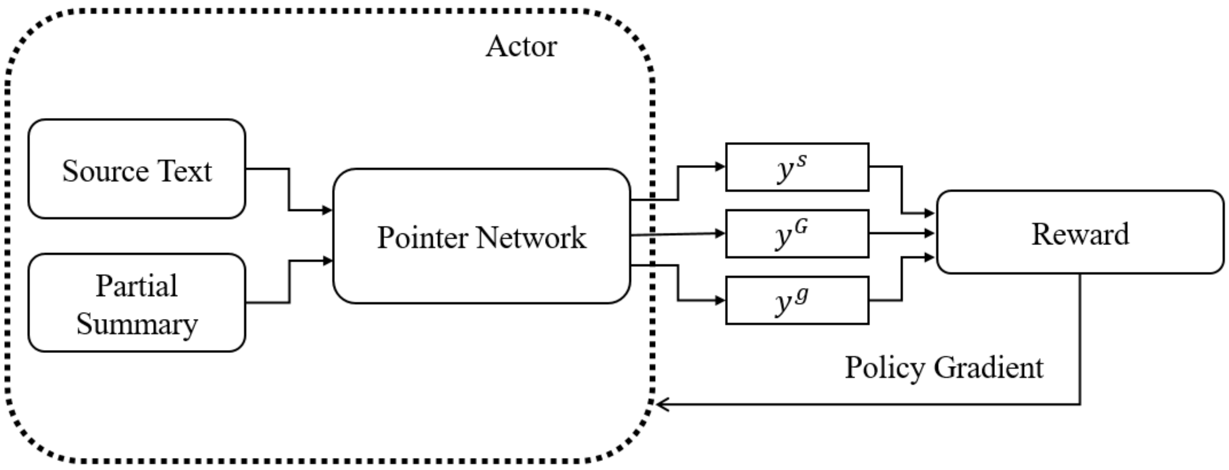

Figure 1：Model Overview. For each pointer network output distribution, a specific action y^s is sampled and the greedy action y^g is extracted and action y^G is the ground truth.

eration task (Ranzato et al. 2015).

In the model training process, each input word of the decoder uses the last word correctly output in the training sample, while in the testing stage, each input word of the decoder is its last output word, which results in the deviation between the test and training results.

The main contributions of this paper include: In this paper a new approach is proposed based on the pointer network (Vinyals, Fortunato, and Jaitly 2015) jointly with an improved attention mechanism, to solve the OOV word problem by using high attention words as candidate outputs based on different attention to the input text, that is, words in the input text can be copied into the output. Besides, the proposed model has been optimized by employing the reinforcement training. Next, We improve the temporal attention (Sankaran et al. 2016) and decoder self-attention (Paulus, Xiong, and Socher 2017). Benefiting from our approaches, repetition is reduced by storing the attention of the history input word and decoding the attention between words in different time steps. We abstract the text abstract model into a reinforcement learning model (Mnih et al. 2015). In the training process, the decoding input of each time step is the output of the previous time step, and the ROUGE score (Lin 2004) of the generated abstract and reference abstract is taken as the reward, which is solved by the policy gradient (Thomas and Brunskill 2017) to solve the exposure deviation problem.

2 Models

The symbols we will use are defined as follows: n_e represents the length of the encoder, n_d represents the length of the decoder, $x=\{x_1,x_2,\cdots x_{n_e}\}$ represents the encoder input word vector sequence, $h=\{h_1,h_2,\cdots,h_{n_e}\}$ represents the output sequence of the encoder, $s=\{s_1,s_2,\cdots,s_{n_d}\}$ represents the output sequence of the decoder, $y=\{y_1,y_2,\cdots,y_{n_d}\}$ represents the final output of the word vector sequence, $y^*=\{y_1^*,y_2^*,\cdots,y_{n_d}^*\}$ represents the ground-truth of training samples, $[a, b]$ represents the combination of a and b into one vector.

The overall structure of the model is shown in Figure 1.

The overall process of training is as follows:

(1) The input word sequence passes through the embedded layer to obtain the same length vector, which is then fed into the encoder.

(2) After encoding all the input text, the encoded information is fed to the decoder.

(3) Feed the real sample or its own output from the previous moment into the decoder (detailed in 2.4) to get the output of the current moment.

(4) Calculate temporal attention and decode self-attention to get the context vector of the encoder and decoder.

(5) Feed the context vector and decoder output into the generated and pointer network to get the output of the word.

(6) Calculate rewards based on output words and real samples, and train the entire network using the policy gradient method.

Only the first 5 steps are required for the test, and the decoder input of step 3 is the output of the previous moment.

The following is a detailed introduction to the basic structure, temporal attention, decoding self-attention, generation and pointer network, loss function and reinforcement learning.

2.1 Basic Structure and Temporal Attention

Our basic structure references (Nallapati and Xiang 2016), the encoder uses a single-layer bidirectional LSTM, consisting of a forward LSTM($LSTM^f$) and a backward LSTM ($LSTM^b$), the encoder's the *i* time step output $h_i = [h_i^f, h_i^b]$.

In order to prevent the generation of repetitive words, the temporal attention is introduced, that is, in each decoding time step to save attention, in the new time step to get attention divided by the sum of historical attention, weaken the previously high focus of the part, enhance the previously less attention to the part. The output of the t time step of the decoder for the attention a_t^e of each time step of the encoder is calculated as follows:

$$e_{ti} = \left(v^e\right)^T \tanh(W_h^e h_i + W_s^e s_t + b_{ti}^e)$$

$$\alpha_t^e = \begin{cases} \exp(e_t) & t = 1 \\ \dfrac{\exp(e_t)}{\exp(\sum_{j=1}^{t-1} e_j)} & \text{other} \end{cases}$$

$$a_t^e = \text{softmax}\left(\alpha_t^e\right)$$

where v^e , W_h^e , W_s^e and b_{ti}^e are learnable parameters.

In the traditional attention mechanism, historical attention is not preserved, so the calculation formula of the traditional attention mechanism a_t^e is $a_t^e = \text{softmax}\left(e_t^e\right)$.

We can get the context vector c_t^e of the encoder based on the attention of each output of the encoder at the t time step:

$$c_t^e = \sum_{i=1}^{n_e} a_{ti}^e h_i$$

2.2 Decoding Self-attention

In addition to the temporary temporal attention mechanism, we also introduce decoding self-attention in order to be able to focus on previously generated words and prevent duplication when generating new words. At the $t > 1$ time step, the decoder outputs attention to the output of the $0 < j < t$ time Step a_t^d :

$$e_{tj}^d = \left(v^d\right)^T \tanh(W_h^d s_j + W_s^d s_t + b_{tj}^d)$$

$$a_t^d = \text{softmax}\left(e_t^d\right)$$

where v^d , W_h^d , W_s^d and b_{tj}^d are learnable parameters.

At the $t = 1$ time step, the decoder context vector c_t^d is a 0 vector. When $t > 1, c_t^d$:

$$c_t^d = \sum_{k=1}^{j} a_{tk}^d s_i$$

2.3 Generate and Pointer Network

The final output word in the t time step is distributed as P_v , indicating the probability of each word being output in the word list, and is related to the context vector c_t^e of the encoder, the context vector c_t^d of the decoder, and the current output s_t of the decoder, using linear function and softmax to calculate:

$$P_v^t = \text{softmax}(W_{out}[c_t^e, c_t^d, s_t] + b_{out})$$

where W_{out} and b_{out} are learnable parameters.

However, P_v^t only decides that a word in the word list should be output. If a word in the original text is needed but not in the word list, it cannot be solved. Therefore, we use pointer network to determine whether a word should be copied based on the attention to the input word.

We define the variable P_{gen}^t to determine the probability of outputting a word based on P_v^t , then $1 - P_{gen}$ represents the probability of copying a word:

$$P_{gen}^t = \sigma\left({w_{ce}^t}^T c_t^e + {w_{cd}^t}^T c_t^d + {w_s^t}^T s_t + b_{gen}^t\right)$$

where w_{ce}^t , w_{cd}^t , w_s^t and b_{gen}^t are learnable parameters, σ is the sigmoid activation function.

Combining P_v^t and pointer network, we get the probability of the final output word y :

$$P^t(y) = P_{gen}^t P_v^t(y) + (1 - P_{gen}^t)\sum_{i=1}^{n_e} a_{ti}^e (x_i = y)$$

Of course, if the word y does not exist in the word list, then $P_v^t(y) = 0$.

2.4 Loss Function and Reinforcement Learning

When training RNN to do sequence generation tasks, the most common method is teacher forcing(Williams and Zipser 1989), which trains the network at each time step of decoding with maximum likelihood estimation as the target. Maximizing likelihood estimation is equivalent to minimizing the loss function below:

$$L_{ML} = -\sum_{t=1}^{n_d} \log P(y_t^* \mid y_1^*, \cdots, y_{t-1}^*, x)$$

Firstly, using such loss function, the decoder input is real output when training, and the decoder output is its own output when testing, it will cause exposure bias. Secondly, there is a certain deviation between the target of likelihood estimation and the evaluation index (such as ROUGE), the value of loss function will decrease, but the ROUGE will increase, or vice versa.

We use reinforcement learning to solve the above two problems. For exposure bias, use the output of the decoder itself as input to the next decoder during training. For the deviation between the optimization target and the evaluation index, using the principle of reinforcement learning, the evaluation index is directly taken as the target, and the network is trained by the strategy gradient.

We use the entire network as the actor, the ROUGE-L score of the actor's output y as a reward, denoted as $R(y)$, the maximum value is 1 and the minimum value is 0. So the task target is to maximize the reward, that is, the loss function $L_{RL}(\theta)$ is the negative expectation reward:

$$L_{RL}(\theta) = -E_{y \sim P_\theta(y)}[R(y)]$$

where θ represents all trained parameters, $P_\theta(y) = P(x)\prod_{t=1}^{n_d} P_\theta(y_t \mid y_1, \cdots, y_{t-1}, x)$ represents the probability of actor output sentence y .

According to the policy gradient algorithm, we get the gradient of the loss function about θ:

$$\nabla_\theta L_{RL}(\theta) = -E_{y \sim P_\theta(y)}[R(y)\nabla_\theta \log P_\theta(y)]$$

In order to reduce the variance of the gradient, we use a policy gradient algorithm with baseline, and its loss function is as follows:

$$L_{RL} = -(R(y^s) - R(y^g))\sum_{t=1}^{n_d} \log P(y_t^s \mid y_1^s, \cdots, y_{t-1}^s, x)$$

where y^s represents output sampled according to distributed $P(y_t^s \mid y_1^s, \cdots, y_{t-1}^s, x)$, y^g represents the output obtained according to distributed $P(y_t^g \mid y_1^g, \cdots, y_{t-1}^g, x)$ greed.

In the above formula, $R(y^g)$ is the baseline and $R(y^s)$ is the target. When both L_{ML} and L_{RL} are considered in training, the network needs to be updated separately based on two loss functions. Therefore, the storage space occupied during training (the memory used when using the GPU) is twice times the use of a single loss function.

Considering the diversity of the training samples, the output y^g is inherently quite random, so we use $R(y^g)$ as the optimization target and $R(y^s)$ as the baseline. The new modified loss function is:

$$L_{RL} = -(R(y^g) - R(y^s))\sum_{t=1}^{n_d} \log P(y_t^g \mid y_1^g, \cdots, y_{t-1}^g, x)$$

This way, when using both loss functions, you do not need to save the intermediate parameters of output, just save the intermediate parameters that generate, and when you update, the two loss functions can be updated at the same time. Therefore, the storage space occupied during the training process is half of the previous formula and can achieve the same effect.

3 Related Work

Automatic text summarization models are usually divided into abstract models and extraction models. Early work focused on methods based on extraction and compression. From Rush (et al. 2015) for the first time to apply modern neural network to abstra-

Model	ROUGE-1	ROUGE-2	ROUGE-L
Lead-3 [See et al., 2017]	39.2	15.7	35.5
SummaRuNNer [Nallapati et al., 2017]	39.6	16.2	35.3
PointerGenerator+Coverage [See et al., 2017]	39.53	17.28	36.38
Inconsistency Loss [Hsu et al., 2018]	40.68	17.97	37.13
ML+RL [Paulus et al., 2017]	39.87	15.82	36.90
Ours			
Storing attention	37.14	15.35	34.59
Improved attention	39.57	17.15	36.83
Improved attention + RL	**40.75**	**18.03**	**38.11**

Table 1: ROUGE F1 results for various models and ablations on the CNN/Daily Mail test set.

ctive text summarization, abstract models show excellent performance. These models include the use of recurrent neural networks (RNN), where encoder and decoder are constructed using either Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber 1997) or Gated Recurrent Unit (GRU) (Cho et al. 2014), attention (Nallapati and Xiang 2016), coverage (Chen et al. 2016; See, Liu, and Manning 2017), the copy mechanism (Gu et al. 2016; See, Liu, and Manning 2017), and convolutional neural networks (CNN) (Dauphin et al. 2017; Gehring et al. 2017).

Reinforcement learning is used to optimize non-differential metrics for language generation and mitigate exposure bias. Ranzato (et al. 2015) have applied reinforcement learning to train various RNN-based sequence generation task models, which resulted in significant improvements over previous supervised learning methods. Paulus, Xiong, and Socher (2017) use reinforcement learning algorithm policy gradient methods for abstractive summarization, Rennie (et al. 2017) designed a self-critical sequence training method for image captioning tasks.

4 Experiment

For all experiments, the dimension of the word vector is 128, the pre-trained word vector is not used, such as word2vec (Mikolov et al. 2013), word vector is learned from scratch during training, the internal state of LSTM is 256 dimensions, and the word list uses 50,000 words. The optimization method uses Adagrad (Duchi, Hazan, and Singer 2011), which was found to work best of Stochastic Gradient Descent, Adadelta, Momentum, Adam and RMSProp, with a learning rate of 0.15 and an initial accumulator value of 0.1.

We use the CNN/Daily Mail dataset for training and validation, which online news articles and multiple-sentence summaries, averaging an article with 781 tokens, each article matching an average of 3.75 sentences, with an average of 56 tokens. We used scripts supplied by (Nallapati and Xiang 2016) to obtain the same version of the data, which has 287,226 training pairs, 13,368 validation pairs and 11,490 test pairs. Following (See, Liu, and Manning 2017) we choose the non-anonymized version of the dataset.

On CNN/Daily Mail dataset, we report the full-length F-1 score of the ROUGE-1, ROUGE-2 and ROUGE-L metrics (which respectively measure the word-overlap, bigram-overlap, and longest common sequence between the reference summary and the summary to be evaluated), calculated using PyRouge package. For ML+RL training, we use the ROUGE-L score as a reinforcement reward.

5 Results

Our results for the CNN/Daily Mail dataset are shown in Table 1. We compare the performance of many recent approaches with our model. Our full model scores are shown in the last line of the table. Compared with other models, we can find that there are some improvements in the scores of the three evaluation indicators. Compared with the best performing inconsistency loss (Hsu et al. 2018), our model has a slight improvement in ROUGE-1 and ROUGE-2 scores, and the ROUGE-L score is more obvious. This is due to the fact that we set ROUGE-L as reward for training.

As shown in the last four lines of Table 1, we study the ablation of our model variables to analyze the importance of each component. We use three ablation models for the experiments. The first model is just to store attention; The second model uses improved attention; And the third model is to use RL based on improved attention. By comparing the first two models, using improved attention can be 2.16 average ROUGE higher than storing attention, indicating that improved attention provides effective help to the model. Comparing the latter two models, we observe that full model outperforms by 1.11 on average ROUGE, indicating that RL has an effect on the model. Ablation studies have shown that each module is necessary for our complete model, and that improvements on all indicators are statistically significant.

6 Conclusion and Future Work

In this work, we propose an improved attention model with reinforcement learning for abstractive text summarization. We evaluate our model on CNN/Daily Mail dataset, the experimental results show that compared to previous systems our approach effectively improves performance.

Note that the model in this paper mainly uses the basic reinforcement learning algorithm. In the future, our goal is to use more advanced reinforcement learning algorithm to achieve better results.

Acknowledgments

The financial support for this work is provided by the National Science Foundation of China (No.61502082, No.61502080).

References

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

Barzilay, Regina, and Kathleen R McKeown. 2005. Sentence fusion for multidocument news summarization. Computational Linguistics 31(3), 297–328.

Berg-Kirkpatrick, Taylor, Dan Gillick, and Dan Klein. 2011. Jointly learning to extract and compress, Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1, pp. 481–90.

Chen, Qian, Xiaodan Zhu, Zhenhua Ling, Si Wei, and Hui Jiang. 2016. Distraction-based neural networks for modeling documents. IJCAI International Joint Conference on Artificial Intelligence 2016-January, 2754–60.

Chen, Qian, Xiao-Dan Zhu, Zhen-Hua Ling, Si Wei, and Hui Jiang. 2016. Distraction-Based Neural Networks for Modeling Document. IJCAI, pp. 2754–60.

Cho, Kyunghyun, 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.

Chopra, Sumit, Michael Auli, and Alexander M Rush. 2016. Abstractive sentence summarization with attentive recurrent neural networks, Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp. 93–98.

Dauphin, Yann N, Angela Fan, Michael Auli, and David Grangier. 2017. Language modeling with gated convolutional networks, Proceedings of the 34th International Conference on Machine Learning-Volume 70, pp. 933–41.

Dorr, Bonnie, David Zajic, and Richard Schwartz. 2003. Hedge trimmer: A parse-and-trim approach to headline generation, Proceedings of the HLT-NAACL 03 on Text summarization workshop-Volume 5, pp. 1–8.

Duchi, J, E Hazan, and Y Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. The Journal of Machine Learning Research 12, 2121–59.

Duchi, John, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. Journal of Machine Learning Research 12(Jul), 2121–59.

Gehring, Jonas, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017. Convolutional sequence to sequence learning, Proceedings of the 34th International Conference on Machine Learning-Volume 70, pp. 1243–52.

Gehrmann, Sebastian, Yuntian Deng, and Alexander M Rush. 2018. Bottom-up abstractive summarization. arXiv preprint arXiv:1808.10792.

Gu, Jiatao, Zhengdong Lu, Hang Li, and Victor OK Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. arXiv preprint arXiv:1603.06393.

Gulcehre, Caglar, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. 2016. Pointing the unknown words. arXiv preprint arXiv:1603.08148.

Hochreiter, Sepp, and Jurgen Schmidhuber. 1997. Long short term memory. Neural computation. Neural Computation 9(8), 1735–80.

Hochreiter, Sepp, and Jürgen Schmidhuber. 1997. Long short-term memory. Neural computation 9(8), 1735–80.

Hsu, Wan-Ting, Chieh-Kai Lin, Ming-Ying Lee, Kerui Min, Jing Tang, and Min Sun. 2018. A unified model for extractive and abstractive summarization using inconsistency loss. arXiv preprint arXiv:1805.06266.

Kraaij, W, M Spitters, and A Hulth. 2002. Headline extraction based on a combination of uni- and multidocument summarization techniques. Duc 2002.

Kraaij, Wessel, Martijn Spitters, and Anette Hulth. 2002. Headline extraction based on a combination of uni-and multidocument summarization techniques, Proceedings of the ACL workshop on Automatic Summarization/Document Understanding Conference (DUC 2002). ACL.

Lin, CY. 2004. Rouge: A package for automatic evaluation of summaries, Proceedings of the workshop on text summarization branches out (WAS 2004), pp. 25–26.

Lin, Chin-Yew. 2004. Rouge: A package for automatic evaluation of summaries, Text summarization branches out, pp. 74–81.

Martins, André FT, and Noah A Smith. 2009. Summarization with a joint model for sentence extraction and compression, Proceedings of the Workshop on Integer Linear Programming for Natural Langauge Processing, pp. 1–9.

Mikolov, Tomas, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781.

Mnih, Volodymyr, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. 2015. Human-level control through deep reinforcement learning. Nature 518, 529.

Nallapati, Ramesh, and Bing Xiang. 2016. Abstractive Text Summarization using Sequence-to-sequence RNNs and Beyond Cicero dos Santos, 280–90.

Nallapati, Ramesh, Feifei Zhai, and Bowen Zhou. 2017. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents, Thirty-First AAAI Conference on Artificial Intelligence.

Nallapati, Ramesh, Bowen Zhou, Caglar Gulcehre, Bing Xiang, and others. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. arXiv preprint arXiv:1602.06023.

Neto, Joel Larocca, Alex A Freitas, and Celso AA Kaestner. 2002. Automatic text summarization using a machine learning approach, Brazilian Symposium on Artificial Intelligence, pp. 205–15.

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

Ranzato, Marc'Aurelio, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2015. Sequence level training with recurrent neural networks. arXiv preprint arXiv:1511.06732.

Rennie, Steven J, Etienne Marcheret, Youssef Mroueh, Jerret Ross, and Vaibhava Goel. 2017. Self-critical sequence training for image captioning, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 7008–24.

Sankaran, Baskaran, Haitao Mi, Yaser Al-Onaizan, and Abe Ittycheriah. 2016. Temporal attention model for neural machine translation. arXiv preprint arXiv:1608.02927.

See, Abigail, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. arXiv preprint arXiv:1704.04368.

Sutskever, Ilya, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks, Advances in neural information processing systems, pp. 3104–12.

Thomas, Philip S, and Emma Brunskill. 2017. Policy gradient methods for reinforcement learning with function approximation and action-dependent baselines. arXiv preprint arXiv:1706.06643.

Vinyals, Oriol, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks, Advances in Neural Information Processing Systems, pp. 2692–2700.

Wang, Li, Junlin Yao, Yunzhe Tao, Li Zhong, Wei Liu, and Qiang Du. 2018. A reinforced topic-aware convolutional sequence-to-sequence model for abstractive text summarization. arXiv preprint arXiv:1805.03616.

Williams, Ronald J, and David Zipser. 1989. A learning algorithm for continually running fully recurrent neural networks. Neural computation 1(2), 270–80.

Zhou, Qingyu, Nan Yang, Furu Wei, Shaohan Huang, Ming Zhou, and Tiejun Zhao. 2018. Neural document summarization by jointly learning to score and select sentences. arXiv preprint arXiv:1807.02305.