An Editorial Network for Enhanced Document Summarization

Edward Moroshko [*]
Electrical Engineering Dept.
Technion – Israel Institute of Technology
Haifa, Israel
edward.moroshko@gmail.com

Guy Feigenblat, Haggai Roitman, David Konopnicki
IBM Research AI
Haifa University Campus
Haifa, Israel
{guyf,haggai,davidko}@il.ibm.com

Abstract

We suggest a new idea of *Editorial Network* – a mixed extractive-abstractive summarization approach, which is applied as a post-processing step over a given sequence of extracted sentences. We further suggest an effective way for training the "editor" based on a novel soft-labeling approach. Using the CNN/DailyMail dataset we demonstrate the effectiveness of our approach compared to state-of-the-art extractive-only or abstractive-only baselines.

1 Introduction

Automatic text summarizers condense a given piece of text into a shorter version (the summary). This is done while trying to preserve the main essence of the original text and keeping the generated summary as readable as possible.

Existing summarization methods can be classified into two main types, either *extractive* or *abstractive*. Extractive methods select and order text fragments (e.g., sentences) from the original text source. Such methods are relatively simpler to develop and keep the extracted fragments untouched, allowing to preserve important parts, e.g., keyphrases, facts, opinions, etc. Yet, extractive summaries tend to be less fluent, coherent and readable and may include superfluous text.

Abstractive methods apply natural language paraphrasing and/or compression on a given text. A common approach is based on the encoder-decoder (seq-to-seq) paradigm (Sutskever et al., 2014), with the original text sequence being encoded while the summary is the decoded sequence. While such methods usually generate summaries with better readability, their quality declines over longer textual inputs, which may lead to a higher redundancy (Paulus et al., 2017). Moreover, such methods are sensitive to vocabulary size, making them more difficult to train and generalize (See et al., 2017).

A common approach for handling long text sequences in abstractive settings is through *attention* mechanisms, which aim to imitate the attentive reading behaviour of humans (Chopra et al., 2016). Two main types of attention methods may be utilized, either *soft* or *hard*. Soft attention methods first locate salient text regions within the input text and then bias the abstraction process to prefer such regions during decoding (Cohan et al., 2018; Gehrmann et al., 2018; Hsu et al., 2018; Nallapati et al., 2016; Li et al., 2018; Pasunuru and Bansal, 2018; Tan et al., 2017). On the other hand, hard attention methods perform abstraction only on text regions that were initially selected by some extraction process (Chen and Bansal, 2018; Nallapati et al., 2017; Liu et al., 2018).

Compared to previous works, whose final summary is either entirely extracted or generated using an abstractive process, in this work, we suggest a new idea of "*Editorial Network*" (EditNet) – a *mixed extractive-abstractive* summarization approach. A summary generated by *EditNet* may include sentences that were either extracted, abstracted or of both types. Moreover, per considered sentence, *EditNet* may decide not to take either of these decisions and completely reject the sentence.

Using the CNN/DailyMail dataset we demonstrate that, *EditNet*'s summarization quality is highly competitive to that obtained

[*] Work was done during a summer internship in IBM Research AI

Proceedings of the 2nd Workshop on New Frontiers in Summarization, pages 57–63
Hong Kong, China, November 4, 2019.

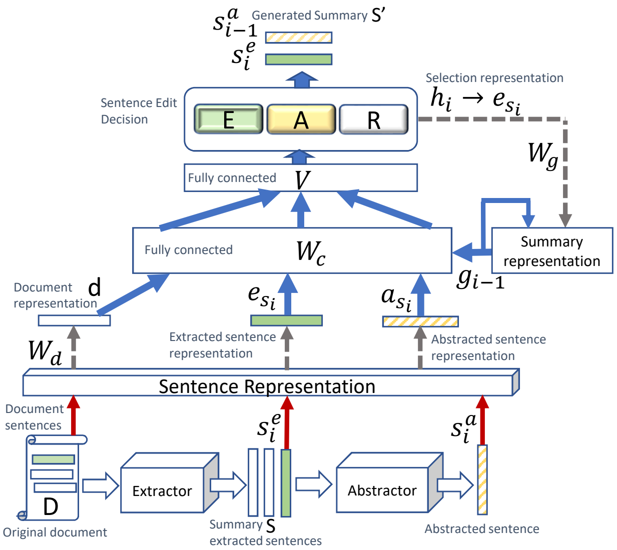

Figure 1: Editorial Network (*EditNet*)

by both state-of-the-art abstractive-only and extractive-only baselines.

2 Editorial Network

Figure 1 depicts the architecture of *EditNet*. *EditNet* is applied as a post-processing step over a given input summary whose sentences were initially selected by some extractor. The key idea behind *EditNet* is to create an automatic editing process to enhance summary quality.

Let S denote a summary which was extracted from a given text (document) D. The editorial process is implemented by iterating over sentences in S according to the selection order of the extractor. For each sentence in S, the “editor" may make three possible decisions. The first decision is to keep the extracted sentence untouched (represented by label E in Figure 1). The second alternative is to rephrase the sentence (represented by label A in Figure 1). Such a decision, for example, may represent the editor’s wish to simplify or compress the original source sentence. The last possible decision is to completely reject the sentence (represented by label R in Figure 1). For example, the editor may wish to ignore a superfluous or duplicate information expressed in the current sentence. An example mixed summary generated by our approach is depicted in Figure 2 in the appendix, further emphasizing the various editor’s decisions.

Editor’s automatic summary:
E: what was supposed to be a fantasy sports car ride at walt disney world speedway turned deadly when a lamborghini crashed into a guardrail. A: *the crash took place sunday at the exotic driving experience*[a]. A: *the lamborghini ’s passenger , gary terry , died at the scene*[b]. R: ~~petty holdings , which operates the exotic driving experience at walt disney world speedway , released a statement sunday night about the crash.~~

[a] Original extracted sentence: “the crash took place sunday at the exotic driving experience , which bills itself as a chance to drive your dream car on a racetrack".

[b] Original extracted sentence: “the lamborghini ’s passenger , 36-year-old gary terry of davenport , florida , died at the scene , florida highway patrol said"

Ground truth summary:
the crash occurred at the exotic driving experience at walt disney world speedway. officials say the driver , 24-year-old tavon watson , lost control of a lamborghini. passenger gary terry , 36 , died at the scene.

Figure 2: An example mixed summary (annotated with the editor’s decisions) taken from the CNN/DM dataset

2.1 Implementing the editor’s decisions

For a given sentence $s \in D$, we now denote by s^e and s^a its original (extracted) and paraphrased (abstracted) versions. To obtain s^a we use an abstractor, whose details will be shortly explained (see Section 2.2). Let $e_s \in \mathbb{R}^n$ and $a_s \in \mathbb{R}^n$ further denote the corresponding sentence representations of s^e and s^a, respectively. Such representations allow to compare both sentence versions on the same grounds.

Recall that, for each sentence $s_i \in S$ (in order) the editor makes one of the three possible decisions: extract, abstract or reject s_i. Therefore, the editor may modify summary S by paraphrasing or rejecting some of its sentences, resulting in a mixed extractive-abstractive summary S'.

Let l be the number of sentences in S. In each step $i \in \{1, 2, \ldots, l\}$, in order to make an educated decision, the editor considers both sentence representations e_{s_i} and a_{s_i} as its input, together with two additional auxiliary representations. The first auxiliary representation is that of the whole document D itself, hereinafter denoted $d \in \mathbb{R}^n$. Such a representation provides a *global context* for decision making. Assuming document D has N sentences, let $\bar{e} = \frac{1}{N} \sum\limits_{s \in D}^{N} e_s$. Following (Chen and Bansal, 2018; Wu and Hu, 2018a), d is then calculated as follows: $d = tanh\left(W_d \bar{e} + b_d\right)$, where $W_d \in \mathbb{R}^{n \times n}$ and $b_d \in \mathbb{R}^n$ are learnable parameters.

The second auxiliary representation is that of the summary that was generated by the editor so far, denoted at step i as $g_{i-1} \in \mathbb{R}^n$, with $g_0 = \vec{0}$. Such a representation provides a *local context* for decision making. Given the four representations as an input, the editor's decision for sentence $s_i \in S$ is implemented using two fully-connected layers, as follows:

$$softmax\left(Vtanh\left(W_c[e_{s_i}, a_{s_i}, g_{i-1}, d] + b_c\right) + b\right), \tag{1}$$

where $[\cdot]$ denotes the vectors concatenation, $V \in \mathbb{R}^{3\times m}$, $W_c \in \mathbb{R}^{m\times 4n}$, $b_c \in \mathbb{R}^m$ and $b \in \mathbb{R}^3$ are learnable parameters.

In each step i, therefore, the editor chooses the action $\pi_i \in \{\mathsf{E}, \mathsf{A}, \mathsf{R}\}$ with the highest likelihood (according to Eq. 1), further denoted $p(\pi_i)$. Upon decision, in case it is either E or A, the editor appends the corresponding sentence version (i.e., either s_i^e or s_i^a) to S'; otherwise, the decision is R and sentence s_i is discarded. Depending on its decision, the current summary representation is further updated as follows:

$$g_i = g_{i-1} + tanh\left(W_g h_i\right), \tag{2}$$

where $W_g \in \mathbb{R}^{n\times n}$ are learnable parameters, g_{i-1} is the summary representation from the previous decision step; and $h_i \in \{e_{s_i}, a_{s_i}, \vec{0}\}$, depending on which decision is made.

Such a network architecture allows to capture various complex interactions between the different inputs. For example, the network may learn that given the global context, one of the sentence versions may allow to produce a summary with a better coverage. As another example, based on the interaction between both sentence versions with either of the local or global contexts (and possibly among the last two), the network may learn that both sentence versions may only add superfluous or redundant information to the summary, and therefore, decide to reject both.

2.2 Extractor and Abstractor

As a proof of concept, in this work, we utilize the extractor and abstractor that were previously used in (Chen and Bansal, 2018), with a slight modification to the latter, motivated by its specific usage within our approach. We now only highlight important aspects of these two sub-components and kindly refer the reader to (Chen and Bansal, 2018) for the full implementation details.

The extractor of (Chen and Bansal, 2018) consists of two main sub-components. The first is an *encoder* which encodes each sentence $s \in D$ into e_s using an hierarchical representation[1]. The second is a *sentence selector* using a *Pointer-Network* (Vinyals et al., 2015). For the latter, let $P(s)$ be the selection likelihood of sentence s.

The abstractor of (Chen and Bansal, 2018) is basically a standard encoder-aligner-decoder with a copy mechanism (See et al., 2017). Yet, instead of applying it directly only on a single given extracted sentence $s_i^e \in S$, we apply it on a "chunk" of three consecutive sentences[2] (s_-^e, s_i^e, s_+^e), where s_-^e and s_+^e denote the sentence that precedes and succeeds s_i^e in D, respectively. This in turn, allows to generate an abstractive version of s_i^e (i.e., s_i^a) that benefits from a wider local context. Inspired by previous soft-attention methods, we further utilize the extractor's sentence selection likelihoods $P(\cdot)$ for enhancing the abstractor's attention mechanism, as follows. Let $C(w_j)$ denote the abstractor's original attention value of a given word w_j occurring in (s_-^e, s_i^e, s_+^e); we then recalculate this value to be $C'(w_j) = \frac{C(w_j)\cdot P(s)}{Z}$, with $w_j \in s$ and $s \in \{s_-^e, s_i^e, s_+^e\}$; $Z = \sum_{s' \in \{s_-^e, s_i^e, s_+^e\}} \sum_{w_j \in s'} C(w_j)\cdot P(s')$ denotes the normalization term.

2.3 Sentence representation

Recall that, in order to compare s_i^e with s_i^a, we need to represent both sentence versions on as similar grounds as possible. To achieve that, we first replace s_i^e with s_i^a within the original document D. By doing so, we basically treat sentence s_i^a as if it was an ordinary sentence within D, where the rest of the document remains untouched. We then obtain s_i^a's representation by encoding it using the extractor's encoder in a similar way in which sentence s_i^e was originally supposed to be encoded. This results in a representation a_{s_i} that provides a comparable alternative to e_{s_i}, whose encoding is expected to be effected by similar contextual grounds.

2.4 Network training

We conclude this section with the description of how we train the editor using a novel soft labeling approach. Given text S (with l extracted sentences), let $\pi = (\pi_1, \ldots, \pi_l)$ denote its editing decisions

[1] Such a representation is basically a combination of a temporal convolutional model followed by a biLSTM encoder.

[2] The first and last chunks would only have two consecutive sentences.

(sequence). We define the following "soft" cross-entropy loss:

$$\mathcal{L}(\pi|S) = -\frac{1}{l}\sum_{s_i\in S}\sum_{\pi_i\in\{\mathsf{E,A,R}\}} y(\pi_i)\log p(\pi_i), \tag{3}$$

where, for a given sentence $s_i \in S$, $y(\pi_i)$ denotes its soft-label for decision.

We next explain how each soft-label $y(\pi_i)$ is estimated. To this end, we utilize a given summary quality metric $r(S')$ which can be used to evaluate the quality of any given summary S' (e.g., ROUGE (Lin, 2004)). Overall, for a given text input S with l sentences, there are 3^l possible summaries S' to consider. Let $\pi^* = (\pi^*_1, \ldots, \pi^*_l)$ denote the best decision sequence which results in the summary which maximizes $r(\cdot)$. For $i \in \{1, 2, \ldots, l\}$, let $\bar{r}(\pi^*_1, \ldots, \pi^*_{i-1}, \pi_i)$ denote the average $r(\cdot)$ value obtained by decision sequences that start with the prefix $(\pi^*_1, \ldots, \pi^*_{i-1}, \pi_i)$. Based on π^*, the soft label $y(\pi_i)$ is then calculated[3] as follows:

$$y(\pi_i) = \frac{\bar{r}(\pi^*_1, \ldots, \pi^*_{i-1}, \pi_i)}{\sum_{\pi_j\in\{\mathsf{E,A,R}\}} \bar{r}(\pi^*_1, \ldots, \pi^*_{i-1}, \pi_j)} \tag{4}$$

3 Evaluation

3.1 Dataset and Setup

We trained, validated and tested our approach using the non-annonymized version of the CNN/DailyMail dataset (Hermann et al., 2015). Following (Nallapati et al., 2016), we used the story highlights associated with each article as its ground truth summary. We further used the F-measure versions of ROUGE-1, ROUGE-2 and ROUGE-L as our evaluation metrics (Lin, 2004).

The extractor and abstractor were trained similarly to (Chen and Bansal, 2018) (including the same hyperparameters). The Editorial Network (hereinafter denoted *EditNet*) was trained according to Section 2.4, using the ADAM optimizer with a learning rate of 10^{-4} and a batch size of 32. Following (Dong et al., 2018; Wu and Hu, 2018a), we set the reward metric to be $r(\cdot) = \alpha R\text{-}1(\cdot) + \beta R\text{-}2(\cdot) + \gamma R\text{-}L(\cdot)$; with $\alpha = 0.4$, $\beta = 1$ and $\gamma = 0.5$, which were further suggested by (Wu and Hu, 2018a).

We further applied the *Teacher-Forcing* approach (Lamb et al., 2016) during training, where we considered the true-label instead of the editor's decision (including when updating g_i at each step i according to Eq. 2). Following (Chen and Bansal, 2018), we set $m = 512$ and $n = 512$. We trained for 20 epochs, which has taken about 72 hours on a single GPU. We chose the best model over the validation set for testing. Finally, all components were implemented in Python 3.6 using the pytorch 0.4.1 package.

Table 1: Quality evaluation using ROUGE F-measure (ROUGE-1, ROUGE-2, ROUGE-L) on CNN/DailyMail non-annonymized dataset

	R-1	R-2	R-L
Extractive			
Lead-3	40.00	17.50	36.20
SummaRuNNer (Nallapati et al., 2017)	39.60	16.20	35.30
EditNet$_\mathsf{E}$	38.43	18.07	35.37
Refresh (Narayan et al., 2018)	40.00	18.20	36.60
Rnes w/o coherence (Wu and Hu, 2018b)	41.25	18.87	37.75
BanditSum (Dong et al., 2018)	41.50	18.70	37.60
Latent (Zhang et al., 2018)	41.05	18.77	37.54
rnn-ext+RL (Chen and Bansal, 2018)	41.47	18.72	37.76
NeuSum (Zhou et al., 2018)	41.59	19.01	37.98
BERTSUM (Liu, 2019)	43.25	20.24	39.63
Abstractive			
Pointer-Generator (See et al., 2017)	39.53	17.28	36.38
KIGN+Prediction-guide (Li et al., 2018)	38.95	17.12	35.68
Multi-Task(EG+QG) (Guo et al., 2018)	39.81	17.64	36.54
EditNet$_\mathsf{A}$	40.00	17.73	37.53
rnn-ext+abs+RL (Chen and Bansal, 2018)	40.04	17.61	37.59
RL+pg+cbdec (Jiang and Bansal, 2018)	40.66	17.87	37.06
Saliency+Entail. (Pasunuru and Bansal, 2018)	40.43	18.00	37.10
Inconsistency loss (Hsu et al., 2018)	40.68	17.97	37.13
Bottom-up (Gehrmann et al., 2018)	41.22	18.68	38.34
DCA (Celikyilmaz et al., 2018)	41.69	19.47	37.92
Mixed Extractive-Abstractive			
EditNet	41.42	19.03	38.36

3.2 Results

Table 1 compares the quality of *EditNet* with that of several state-of-the-art extractive-only or abstractive-only baselines. This includes the extractor (*rnn-ext-RL*) and abstractor (*rnn-ext-abs-RL*) components of (Chen and Bansal, 2018) that we utilized for implementing *EditNet* [4].

We further report the quality of *EditNet* when it was being enforced to take an extract-only or abstract-only decision, denoted hereinafter as *EditNet*$_\mathsf{E}$ and *EditNet*$_\mathsf{A}$, respectively. The comparison of *EditNet* to both *EditNet*$_\mathsf{E}$ and *EditNet*$_\mathsf{A}$ variants provides a strong empirical proof that, by utilizing an hybrid decision approach, a

[3] For $i = 1$ we have: $\bar{r}(\pi^*_1, \ldots, \pi^*_0, \pi_1) = \bar{r}(\pi_1)$.

[4] The *rnn-ext-RL* extractor results reported in Table 1 are the ones that were reported by (Chen and Bansal, 2018). Training the public extractor released by these authors, we obtained the following significantly lower results: see *EditNet*$_\mathsf{E}$

better summarization quality is obtained.

Overall, *EditNet* provides a highly competitive summary quality, where it outperforms most baselines. Interestingly, *EditNet*'s summarization quality is quite similar to that of *NeuSum* (Zhou et al., 2018). Yet, while *NeuSum* applies an extraction-only approach, summaries generated by *EditNet* include a mixture of sentences that have been either extracted or abstracted.

Two models outperform *EditNet*, *BERTSUM* (Liu, 2019) and *DCA* (Celikyilmaz et al., 2018). The *BERTSUM* model gains an impressive accuracy, yet it is an extractive model that utilizes many attention layers running in parallel with millions of parameters (Devlin et al., 2019). *DCA* gains a comparable quality to *EditNet*, it outperforms on R-2 and slightly on R-1. The contextual encoder of *DCA* is comprised of several *LSTM* layers one on top of the other with varied number of agents (hyper-tuned) that transmit messages to each other. Considering the complexity of these models, and the slow down that can incur during training and inference, we think that *EditNet* still provides a useful, high quality and relatively simple extension on top of standard encoder aligned decoder architectures.

On average, 56% and 18% of *EditNet*'s decisions were to abstract (A) or reject (R), respectively. Moreover, on average, per summary, *EditNet* keeps only 33% of the original (extracted) sentences, while the rest (67%) are abstracted ones. This demonstrates that, *EditNet* has a high capability of utilizing abstraction, while being also able to maintain or reject the original extracted text whenever it is estimated to provide the best benefit for the summary's quality.

4 Conclusions and Future Work

We have proposed *EditNet* – a novel alternative summarization approach that instead of solely applying extraction or abstraction, mixes both together. Moreover, *EditNet* implements a novel sentence rejection decision, allowing to "correct" initial sentence selection decisions which are predicted to negatively effect summarization quality. As future work, we plan to evaluate other alternative extractor-abstractor configurations and try to train the network end-to-end. We further plan to explore reinforcement learning (RL) as an alternative decision making approach.

References

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

Yen-Chun Chen and Mohit Bansal. 2018. Fast abstractive summarization with reinforce-selected sentence rewriting. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 675–686. Association for Computational Linguistics.

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

Arman Cohan, Franck Dernoncourt, Doo Soon Kim, Trung Bui, Seokhwan Kim, Walter Chang, and Nazli Goharian. 2018. A discourse-aware attention model for abstractive summarization of long documents. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 615–621. Association for Computational Linguistics.

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

Yue Dong, Yikang Shen, Eric Crawford, Herke van Hoof, and Jackie Chi Kit Cheung. 2018. Banditsum: Extractive summarization as a contextual bandit. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 3739–3748.

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

Han Guo, Ramakanth Pasunuru, and Mohit Bansal. 2018. Soft layer-specific multi-task summarization with entailment and question generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume*

1: Long Papers), pages 687–697. Association for Computational Linguistics.

Karl Moritz Hermann, Tomáš Kočiský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, pages 1693–1701, Cambridge, MA, USA. MIT Press.

Wan-Ting Hsu, 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. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 132–141. Association for Computational Linguistics.

Yichen Jiang and Mohit Bansal. 2018. Closed-book training to improve summarization encoder memory. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4067–4077. Association for Computational Linguistics.

Alex M Lamb, Anirudh Goyal ALIAS PARTH GOYAL, Ying Zhang, Saizheng Zhang, Aaron C Courville, and Yoshua Bengio. 2016. Professor forcing: A new algorithm for training recurrent networks. In *Advances In Neural Information Processing Systems*, pages 4601–4609.

Chenliang Li, Weiran Xu, Si Li, and Sheng Gao. 2018. Guiding generation for abstractive text summarization based on key information guide network. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 55–60. Association for Computational Linguistics.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out: Proceedings of the ACL-04 workshop*, volume 8. Barcelona, Spain.

Peter J Liu, Mohammad Saleh, Etienne Pot, Ben Goodrich, Ryan Sepassi, Lukasz Kaiser, and Noam Shazeer. 2018. Generating wikipedia by summarizing long sequences. *arXiv preprint arXiv:1801.10198*.

Yang Liu. 2019. Fine-tune bert for extractive summarization. *arXiv preprint arXiv:1903.10318*.

Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. 2017. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA.*, pages 3075–3081.

Ramesh Nallapati, Bowen Zhou, Cícero Nogueira dos Santos, Çaglar Gülçehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning, CoNLL 2016, Berlin, Germany, August 11-12, 2016*, pages 280–290.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018. Ranking sentences for extractive summarization with reinforcement learning. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pages 1747–1759.

Ramakanth Pasunuru and Mohit Bansal. 2018. Multi-reward reinforced summarization with saliency and entailment. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 646–653. Association for Computational Linguistics.

Romain Paulus, Caiming Xiong, and Richard Socher. 2017. A deep reinforced model for abstractive summarization. *CoRR*, abs/1705.04304.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 1073–1083.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, pages 3104–3112, Cambridge, MA, USA. MIT Press.

Jiwei Tan, Xiaojun Wan, and Jianguo Xiao. 2017. Abstractive document summarization with a graph-based attentional neural model. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1171–1181. Association for Computational Linguistics.

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

Yuxiang Wu and Baotian Hu. 2018a. Learning to extract coherent summary via deep reinforcement learning. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 5602–5609.

Yuxiang Wu and Baotian Hu. 2018b. Learning to extract coherent summary via deep reinforcement learning. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 5602–5609.

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

Qingyu Zhou, Nan Yang, Furu Wei, Shaohan Huang, Ming Zhou, and Tiejun Zhao. 2018. Neural document summarization by jointly learning to score and select sentences. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 654–663. Association for Computational Linguistics.