A Deep Reinforced Model for Abstractive Summarization

Romain Paulus, Caiming Xiong and **Richard Socher**
{rpaulus,cxiong,rsocher}@salesforce.com

Abstract

Attentional, RNN-based encoder-decoder models for abstractive summarization have achieved good performance on short input and output sequences. However, for longer documents and summaries, these models often include repetitive and incoherent phrases. We introduce a neural network model with intra-attention and a new training method. This method combines standard supervised word prediction and reinforcement learning (RL). Models trained only with the former often exhibit "exposure bias" – they assume ground truth is provided at each step during training. However, when standard word prediction is combined with the global sequence prediction training of RL the resulting summaries become more readable. We evaluate this model on the CNN/Daily Mail and New York Times datasets. Our model obtains a 41.16 ROUGE-1 score on the CNN/Daily Mail dataset, a 5.7 absolute points improvement over previous state-of-the-art models. It also performs well as the first abstractive model on the New York Times corpus. Human evaluation also shows that our model produces higher quality summaries.

1 Introduction

Text summarization is the process of automatically generating natural language summaries from an input document while retaining the important points.

By condensing large quantities of information into short, informative summaries, summarization can aid many downstream applications such as creating news digests, search, and report generation.

There are two prominent types of summarization algorithms. First, extractive summarization systems form summaries by copying parts of the input (Neto et al., 2002; Dorr et al., 2003; Nallapati et al., 2017). Second, abstractive summarization systems generate new phrases, possibly rephrasing or using words that were not in the original text (Chopra et al., 2016; Nallapati et al., 2016; Zeng et al., 2016).

Recently, neural network models (Nallapati et al., 2016; Zeng et al., 2016), based on the attentional encoder-decoder model for machine translation (Bahdanau et al., 2014), were able to generate abstractive summaries with high ROUGE scores. However, these systems have typically focused on summarizing short input sequences (one or two sentences) to generate even shorter summaries. For example, the summaries on the DUC-2004 dataset generated by the state-of-the-art system by Zeng et al. (2016) are limited to 75 characters.

Nallapati et al. (2016) also applied their abstractive summarization model on the CNN/Daily Mail dataset (Hermann et al., 2015), which contains input sequences of up to 800 tokens and multi-sentence summaries of up to 100 tokens. The analysis by Nallapati et al. (2016) illustrate a key problem with attentional encoder-decoder models: they often generate unnatural summaries consisting of repeated phrases.

We present a new abstractive summarization model that achieves state-of-the-art results on the CNN/Daily Mail and similarly good results on the New York Times dataset (NYT) (Sandhaus, 2008). To our knowledge, this is the first model for abstractive summarization on the NYT dataset. We introduce a key attention mechanism and a new learning objective to address the repeating phrase problem: (i) we use an intra-temporal attention in the encoder that records previous attention weights for each of the input tokens while a sequential intra-attention model in the decoder takes into account which words have already been generated by the decoder. (ii) we propose a new objective function by combining the maximum-likelihood cross-entropy loss used in prior work with rewards from policy gradient reinforcement learning to reduce exposure bias. We show that our model achieves 41.16 ROUGE-1 on the CNN/Daily Mail dataset, an absolute improvement of 5.70 to the previous state-of-the-art result. Moreover, we show, through human evaluation of generated outputs, that our model generates more readable summaries compared to other techniques.

2 Neural Intra-attention Model

In this section, we present our intra-attention model based on the encoder-decoder network (Sutskever et al., 2014). In all our equations, $x = \{x_1, x_2, \dots, x_n\}$ represents the sequence of input (article) tokens, $y = \{y_1, y_2, \dots, y_{n'}\}$ the sequence of output (summary) tokens, and $\|$ denotes the vector concatenation operator.

Our model reads the input sequence with a bi-directional LSTM encoder $\{\text{RNN}^{e\text{-}fwd}, \text{RNN}^{e\text{-}bwd}\}$ computing hidden states $h_i^e = [h_i^{e\text{-}fwd} \| h_i^{e\text{-}bwd}]$ from the embedding vectors of x_i. We use a single LSTM decoder RNN^d, computing hidden states h_t^d from the embedding vectors of y_t. Both input and output embeddings are taken from the same matrix W_{emb}. We initialize the decoder hidden state with $h_0^d = h_n^e$.

2.1 Intra-temporal attention on input sequence

At each decoding step t, we use an intra-temporal attention function to attend over specific parts of the encoded input sequence in addition to the decoder's own hidden state and the previously-generated word (Sankaran et al., 2016). This kind of attention prevents the model from attending over the sames parts of the input on different decoding steps. Nallapati et al. (2016) have shown that such an intra-temporal attention can reduce the amount of repetitions when attending over long documents.

We define e_{ti} as the attention score of the hidden input state h_i^e at decoding time step t:

$$e_{ti} = f(h_t^d, h_i^e), \tag{1}$$

where f can be any function returning a scalar e_{ti} from the h_t^d and h_i^e vectors. While some attention models use functions as simple as the dot-product between the two vectors, we choose to use a bilinear function:

$$f(h_t^d, h_i^e) = {h_t^d}^T W_{attn}^e h_i^e. \tag{2}$$

We normalize the attention weights with the following temporal attention function, penalizing input tokens that have obtained high attention scores in past decoding steps. We define new temporal scores e'_{ti}:

$$e'_{ti} = \begin{cases} exp(e_{ti}) & \text{if } t = 1 \\ \frac{exp(e_{ti})}{\sum_{j=1}^{t-1} \exp(e_{ji})} & \text{otherwise.} \end{cases} \tag{3}$$

Finally, we compute the normalized attention scores α_{ti}^e across the inputs and use these weights to obtain the input context vector c_t^e:

$$\alpha_{ti}^e = \frac{e'_{ti}}{\sum_{j=1}^{n} e'_{tj}} \tag{4}$$

$$c_t^e = \sum_{i=1}^{n} \alpha_{ti}^e h_i^e. \tag{5}$$

2.2 Intra-decoder attention

While this intra-temporal attention function ensures that different parts of the encoded input sequence are used, our decoder can still generate repeated phrases based on its own hidden states, especially when generating long sequences. To prevent that, we want to incorporate more information about the previously decoded sequence into the decoder. Looking back at previous decoding steps will allow our model to make more structured predictions and avoid repeating the same information, even if that information was generated many steps away. To achieve this, we introduce an intra-decoder attention mechanism. This mechanism is not present in current encoder-decoder models.

For each decoding step t, our model computes a new decoder context vector c_t^d. We set c_1^d to a vector of zeros since the generated sequence is empty on the first decoding step. For $t > 1$, we use the following equations:

$$e_{tt'}^d = {h_t^d}^T W_{attn}^d h_{t'}^d \tag{6}$$

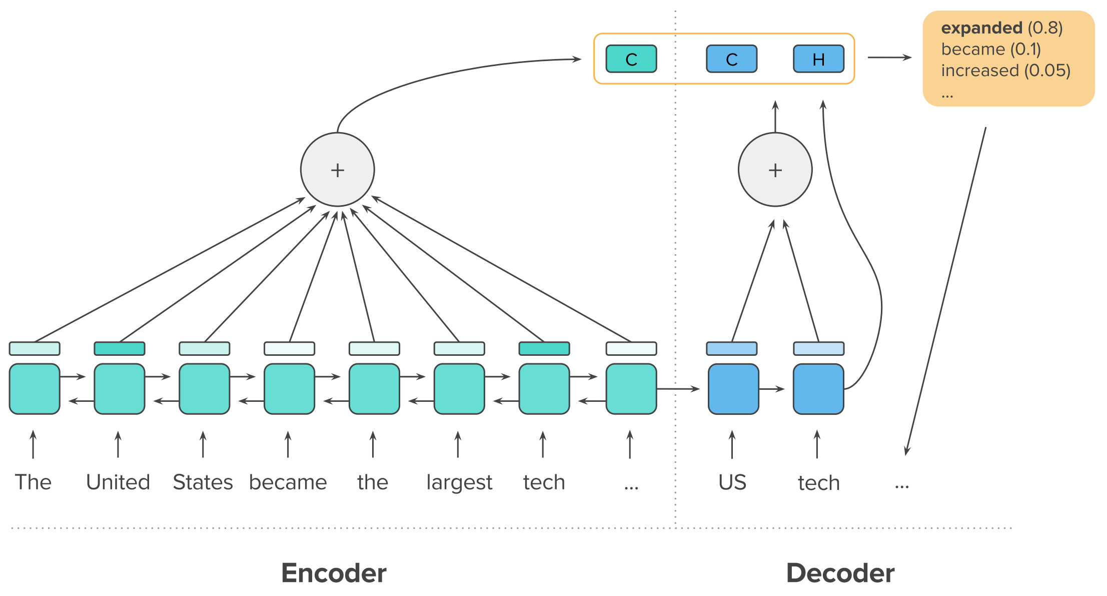

Figure 1: Illustration of the encoder and decoder attention functions combined. The two context vectors (marked "C") are computed from attending over the encoder hidden states and decoder hidden states. Using these two contexts and the current decoder hidden state ("H"), a new word is generated and added to the output sequence.

$$\alpha^d_{tt'} = \frac{exp(e^d_{tt'})}{\sum_{j=1}^{t-1} exp(e^d_{tj})} \tag{7}$$

$$c^d_t = \sum_{j=1}^{t-1} \alpha^d_{tj} h^d_j \tag{8}$$

Figure 1 illustrates the intra-attention context vector computation c^d_t, in addition to the encoder temporal attention, and their use in the decoder.

A closely-related intra-RNN attention function has been introduced by Cheng et al. (2016) but their implementation works by modifying the underlying LSTM function, and they do not apply it to long sequence generation problems. This is a major difference with our method, which makes no assumptions about the type of decoder RNN, thus is more simple and widely applicable to other types of recurrent networks.

2.3 Token generation and pointer

To generate a token, our decoder uses either a token-generation softmax layer or a pointer mechanism to copy rare or unseen from the input sequence. We use a switch function that decides at each decoding step whether to use the token generation or the pointer (Gulcehre et al., 2016; Nallapati et al., 2016). We define u_t as a binary value, equal to 1 if the pointer mechanism is used to output y_t, and 0 otherwise. In the following equations, all probabilities are conditioned on $y_t, \dots, y_{t-1}, x$, even when not explicitly stated.

Our token-generation layer generates the following probability distribution:

$$\begin{aligned} p(y_t|u_t = 0) = \\ \text{softmax}(W_{out}[h^d_t \| c^e_t \| c^d_t] + b_{out}) \end{aligned} \tag{9}$$

On the other hand, the pointer mechanism uses the temporal attention weights α^e_{ti} as the probability distribution to copy the input token x_i.

$$p(y_t = x_i|u_t = 1) = \alpha^e_{ti} \tag{10}$$

We also compute the probability of using the copy mechanism for the decoding step t:

$$p(u_t = 1) = \sigma(W_u[h^d_t \| c^e_t \| c^d_t] + b_u), \tag{11}$$

where σ is the sigmoid activation function.

Putting Equations 9 , 10 and 11 together, we obtain our final probability distribution for the output token y_t:

$$\begin{aligned} p(y_t) = p(u_t = 1)p(y_t|u_t = 1) \\ +p(u_t = 0)p(y_t|u_t = 0). \end{aligned} \tag{12}$$

The ground-truth value for u_t and the corresponding i index of the target input token when $u_t = 1$ are provided at every decoding step during training. We set $u_t = 1$ either when y_t is an out-of-vocabulary token or when it is a pre-defined named entity (see Section 5).

2.4 Sharing decoder weights

In addition to using the same embedding matrix W_{emb} for the encoder and the decoder sequences, we introduce some weight-sharing between this embedding matrix and the W_{out} matrix of the token-generation layer:

$$W_{out} = \tanh(W_{emb} W_{proj}) \tag{13}$$

The goal of this weight-sharing is to use the syntactic and semantic information contained in the embedding matrix to improve the token-generation function. Similar weight-sharing methods have been applied to language modeling (Inan et al., 2016; Press and Wolf, 2016). We believe this method is even more applicable to sequence-to-sequence tasks like summarization where the input and output sequences are tightly related, sharing the same vocabulary and a similar syntax. In practice, we found that a summarization model using such shared weights converges much faster than when using separate W_{out} and W_{emb} matrices.

2.5 Repetition avoidance at test time

Another way to avoid repetitions comes from our observation that in both the CNN/Daily Mail and NYT datasets, ground-truth summaries almost never contain the same trigram twice. Based on this observation, we force our decoder to never output the same trigram more than once during testing. We do this by setting $p(y_t) = 0$ during beam search, when outputting y_t would create a trigram that already exists in the previously decoded sequence of the current beam. Even though this method makes assumptions about the output format and the dataset at hand, we believe that the majority of abstractive summarization tasks would benefit from this hard constraint. We apply this method to all our models in the experiments section.

3 Hybrid Learning Objective

In this section, we explore different ways of training our encoder-decoder model. In particular, we propose reinforcement learning-based algorithms and their application to our summarization task.

3.1 Supervised learning with teacher forcing

The most widely used method to train a decoder RNN for sequence generation, called the
teacher forcing" algorithm (Williams and Zipser, 1989), minimizes a maximum-likelihood loss at each decoding step. We define $y^* = \{y_1^*, y_2^*, \ldots, y_{n'}^*\}$ as the ground-truth output sequence for a given input sequence x. The maximum-likelihood training objective is the minimization of the following loss:

$$L_{ml} = -\sum_{t=1}^{n'} \log p(y_t^* | y_1^*, \ldots, y_{t-1}^*, x) \tag{14}$$

However, minimizing L_{ml} does not always produce the best results on discrete evaluation metrics such as ROUGE (Lin, 2004). This phenomenon has been observed with similar sequence generation tasks like image captioning with CIDEr (Rennie et al., 2016) and machine translation with BLEU (Wu et al., 2016; Norouzi et al., 2016).

There are two main reasons for this discrepancy. The first one, called exposure bias (Ranzato et al., 2015), comes from the fact that the network is fully supervised at each output token during training, always knowing the ground truth sequence up to the next token to predict, but does not have such supervision when testing, hence accumulating errors as it predicts the sequence. The second reason is more specific to our summarization task: while we only have one ground truth sequence per example during training, a summary can still be considered valid by a human even if it is not equal to the reference summary word for word. The number of potentially valid summaries increases as sequences get longer, since there are more ways to arrange tokens to produce paraphrases or different sentence orders. The ROUGE metrics take some of this flexibility into account, but the maximum-likelihood objective does not.

3.2 Policy learning

One way to remedy this is to learn a policy that maximizes a specific discrete metric instead of minimizing the maximum-likelihood loss, which is made possible with reinforcement learning. In our model, we use the self-critical policy gradient training algorithm (Rennie et al., 2016).

For this training algorithm, we produce two separate output sequences at each training iteration: y^s, which is obtained by sampling from the $p(y_t^s | y_1^s, \ldots, y_{t-1}^s, x)$ probability distribution at each decoding time step, and $\hat{y}$, the baseline output, obtained by maximizing the output probability distribution at each time step, essentially performing a greedy search. We define $r(y)$ as the reward function for an output sequence y, comparing it with the ground truth sequence y^* with the evaluation metric of our choice.

$$L_{rl} = (r(\hat{y}) - r(y^s)) \sum_{t=1}^{n'} \log p(y_t^s | y_1^s, \ldots, y_{t-1}^s, x) \tag{15}$$

We can see that minimizing L_{rl} is equivalent to maximizing the conditional likelihood of the sampled sequence y^s if it obtains a higher reward than the baseline $\hat{y}$, thus increasing the reward expectation of our model.

3.3 Mixed training objective function

One potential issue of this reinforcement training objective is that optimizing for a specific discrete metric like ROUGE does not guarantee an increase in quality and readability of the output. It is possible to game such discrete metrics and increase their score without an actual increase in readability or relevance (Liu et al., 2016). While ROUGE measures the n-gram overlap between our generated summary and a reference sequence, human-readability is better captured by a language model, which is usually measured by perplexity.

Since our maximum-likelihood training objective (Equation 14) is essentially a conditional language model, calculating the probability of a token y_t based on the previously predicted sequence $\{y_1, \ldots, y_{t-1}\}$ and the input sequence x, we hypothesize that it can assist our policy learning algorithm to generate more natural summaries. This motivates us to define a mixed learning objective function that combines equations 14 and 15:

$$L_{mixed} = \gamma L_{rl} + (1 - \gamma) L_{ml}, \tag{16}$$

where γ is a scaling factor accounting for the difference in magnitude between L_{rl} and L_{ml}. A similar mixed-objective learning function has been used by Wu et al. (2016) for machine translation on short sequences, but this is its first use in combination with self-critical policy learning for long summarization to explicitly improve readability in addition to evaluation metrics.

4 Related Work

4.1 Neural encoder-decoder sequence models

Neural encoder-decoder models are widely used in NLP applications such as machine translation (Sutskever et al., 2014), summarization (Chopra et al., 2016; Nallapati et al., 2016), and question answering (Hermann et al., 2015). These models use recurrent neural networks (RNN), such as long-short term memory network (LSTM) (Hochreiter and Schmidhuber, 1997) to encode an input sentence into a fixed vector, and create a new output sequence from that vector using another RNN. To apply this sequence-to-sequence approach to natural language, word embeddings (Mikolov et al., 2013; Pennington et al., 2014) are used to convert language tokens to vectors that can be used as inputs for these networks. Attention mechanisms (Bahdanau et al., 2014) make these models more performant and scalable, allowing them to look back at parts of the encoded input sequence while the output is generated. These models often use a fixed input and output vocabulary, which prevents them from learning representations for new words. One way to fix this is to allow the decoder network to point back to some specific words or sub-sequences of the input and copy them onto the output sequence (Vinyals et al., 2015; Ling et al., 2016). Gulcehre et al. (2016) and Merity et al. (2016) combine this pointer mechanism with the original word generation layer in the decoder to allow the model to use either method at each decoding step.

4.2 Reinforcement learning for sequence generation

Reinforcement learning (RL) is a way of training an agent to interact with a given environment in order to maximize a reward. RL has been used to solve a wide variety of problems, usually when an agent has to perform discrete actions before obtaining a reward, or when the metric to optimize is not differentiable and traditional supervised learning methods cannot be used. This is applicable to sequence generation tasks, because many of the metrics used to evaluate these tasks (like BLEU, ROUGE or METEOR) are not differentiable.

In order to optimize that metric directly, Ranzato et al. (2015) have applied the REINFORCE algorithm (Williams, 1992) to train various RNN-based models for sequence generation tasks, leading to significant improvements compared to previous supervised learning methods. While their method requires an additional neural network, called a critic model, to predict the expected reward and stabilize the objective function gradients, Rennie et al. (2016) designed a self-critical sequence training method that does not require this critic model and lead to further improvements on image captioning tasks.

4.3 Text summarization

Most summarization models studied in the past are extractive in nature (Neto et al., 2002; Dorr et al., 2003; Filippova and Altun, 2013; Colmenares et al., 2015; Nallapati et al., 2017), which usually work by identifying the most important phrases of an input document and re-arranging them into a new summary sequence. The more recent abstractive summarization models have more degrees of freedom and can create more novel sequences. Many abstractive models such as Rush et al. (2015), Chopra et al. (2016), Zeng et al. (2016) and Nallapati et al. (2016) are all based on the neural encoder-decoder architecture (Section 4.1).

A well-studied set of summarization tasks is the Document Understanding Conference (DUC) [1]. These summarization tasks are varied, including short summaries of a single document and long summaries of multiple documents categorized by subject. Most abstractive summarization models have been evaluated on the DUC-2004 dataset, and outperform extractive models on that task (Dorr et al., 2003). However, models trained on the DUC-2004 task can only generate very short summaries up to 75 characters, and are usually used with one or two input sentences. Chen et al. (2016) applied different kinds of attention mechanisms for summarization on the CNN dataset, and Nallapati et al. (2016) used different attention and pointer functions on the CNN and Daily Mail datasets combined. In parallel of our work, See et al. (2017) also developed an abstractive summarization model on this dataset with an extra loss term to increase temporal coverage of the encoder attention function.

5 Datasets

5.1 CNN/Daily Mail

We evaluate our model on a modified version of the CNN/Daily Mail dataset (Hermann et al., 2015), following the same pre-processing steps described in Nallapati et al. (2016). We refer the reader to that paper for a detailed description. The final dataset contains 286,817 training examples, 13,368 validation examples and 11,487 testing examples. After limiting the input length to 800 tokens and output length to 100 tokens, the average input and output lengths are respectively 632 and 53 tokens.

5.2 New York Times

The New York Times (NYT) dataset (Sandhaus, 2008) is a large collection of articles published between 1996 and 2007. Even though this dataset has been used to train extractive summarization systems (Hong and Nenkova, 2014; Li et al., 2016) or closely-related models for predicting the importance of a phrase in an article (Yang and Nenkova, 2014; Nye and Nenkova, 2015; Hong et al., 2015), we are the first group to run an end-to-end abstractive summarization model on the article-abstract pairs of this dataset. While CNN/Daily Mail summaries have a similar wording to their corresponding articles, NYT abstracts are more varied, are shorter and can use a higher level of abstraction and paraphrase. We believe that these two formats are a good complement to each other for abstractive summarization models.

Preprocessing: We remove all documents that do not have a full article text, abstract or headline. We concatenate the headline, byline and full article text, separated by special tokens, to produce a single input sequence for each example. We tokenize the input and abstract pairs with the Stanford tokenizer (Manning et al., 2014). We convert all tokens to lower-case and replace all numbers with “0”, remove “(s)” and “(m)” marks in the abstracts and all occurrences of the following words, singular or plural, if they are surrounded by semicolons or at the end of the abstract: “photo”, “graph”, “chart”, “map”, “table” and “drawing”. Since the NYT abstracts almost never contain periods, we consider them multi-sentence summaries if we split sentences based on semicolons. This allows us to make the summary format and evaluation procedure similar to the CNN/Daily Mail dataset. These pre-processing steps give us an average of 549 input tokens and 40 output tokens per example, after limiting the input and output lengths to 800 and 100 tokens.

Pointer supervision: We run each input and abstract sequence through the Stanford named entity recognizer (NER) (Manning et al., 2014). For all named entity tokens in the abstract if the type “PERSON”, “LOCATION”, “ORGANIZATION”

[1] `http://duc.nist.gov/`

or "MISC", we find their first occurrence in the input sequence. We use this information to supervise $p(u_t)$ (Equation 11) and α^e_{ti} (Equation 4) during training. Note that the NER tagger is only used to create the dataset and is no longer needed during testing, thus we're not adding any dependencies to our model. We also add pointer supervision for out-of-vocabulary output tokens if they are present in the input.

Dataset splits: We created our own training,validation, and testing splits for this dataset. Instead of producing random splits, we sorted the documents by their publication date in chronological order and used the first 90% (589,284 examples) for training, the next 5% (32,736) for validation, and the remaining 5% (32,739) for testing. This makes our dataset splits easily reproducible and follows the intuition that if used in a production environment, such a summarization model would be used on recent articles rather than random ones.

6 Results

6.1 Experiments

Setup: We evaluate the intra-decoder attention mechanism and the mixed-objective learning by running the following experiments on both datasets. We first run maximum-likelihood (ML) training with and without intra-decoder attention (removing c^d_t from Equations 9 and 11 to disable intra-attention) and select the best performing architecture. Next, we initialize our model with the best ML parameters and we compare reinforcement learning (RL) with our mixed-objective learning (ML+RL), following our objective functions in Equation 15 and 16. For ML training, we use the teacher forcing algorithm with the only difference that at each decoding step, we choose with a 25% probability the previously generated token instead of the ground-truth token as the decoder input token y_{t-1}, which reduces exposure bias (Venkatraman et al., 2015). We use a $\gamma = 0.9984$ for the ML+RL loss function.

Implementation details: We use two 200-dimensional LSTMs for the bidirectional encoder and one 400-dimensional LSTM for the decoder. We limit the input vocabulary size to 150,000 tokens, and the output vocabulary to 50,000 tokens by selecting the most frequent tokens in the training set. Input word embeddings are 100-dimensional and are initialized with GloVe (Pennington et al., 2014). We train all our models with Adam (Kingma and Ba, 2014) with a batch size of 50 and a learning rate α of 0.001 for ML training and 0.0001 for RL and ML+RL training. At test time, we use beam search of width 5 on all our models to generate our final predictions.

ROUGE metrics and options: We report the full-length F-1 score of the ROUGE-1, ROUGE-2 and ROUGE-L metrics with the Porter stemmer option. For RL and ML+RL training, we use the ROUGE-L score as a reinforcement reward. We also tried ROUGE-2 but we found that it created summaries that almost always reached the maximum length, often ending sentences abruptly.

6.2 Quantitative analysis

Our results for the CNN/Daily Mail dataset are shown in Table 1, and for the NYT dataset in Table 2. We observe that the intra-decoder attention function helps our model achieve better ROUGE scores on the CNN/Daily Mail but not on the NYT dataset. We believe that the difference in summary lengths between the CNN/Daily Mail and NYT datasets is one of the main reason for this difference in outcome, given that our intra-decoder was designed to improve performance over long output sequences. Further differences in the nature of the summaries and the level of complexity and abstraction between these datasets could also explain these intra-attention results, as well as the absolute ROUGE score differences between CNN/Daily Mail and NYT results.

In addition, we can see that on all datasets, both the RL and ML+RL models obtain much higher scores than the ML model. In particular, these methods clearly surpass the state-of-the-art model from Nallapati et al. (2016) on the CNN/Daily Mail dataset.

6.3 Qualitative analysis

We perform human evaluation to ensure that our increase in ROUGE scores is also followed by an increase in human readability and quality. In particular, we want to know whether the ML+RL training objective did improve readability compared to RL.

Evaluation setup: To perform this evaluation, we randomly select 100 test examples from the CNN/Daily Mail dataset. For each example, we show the ground truth summary as well as summaries generated by different models side by side to a human evaluator. The human evaluator does

Model	**ROUGE-1**	**ROUGE-2**	**ROUGE-L**
words-lvt2k-temp-att (Nallapati et al., 2016)	35.46	13.30	32.65
ML, no intra-attention	37.86	14.69	34.99
ML, with intra-attention	38.30	14.81	35.49
RL, with intra-attention	**41.16**	15.75	**39.08**
ML+RL, with intra-attention	39.87	**15.82**	36.90

Table 1: Quantitative results for various models on the CNN/Daily Mail test dataset

Model	**ROUGE-1**	**ROUGE-2**	**ROUGE-L**
ML, no intra-attention	44.26	27.43	40.41
ML, with intra-attention	43.86	27.10	40.11
RL, no intra-attention	**47.22**	30.51	**43.27**
ML+RL, no intra-attention	47.03	**30.72**	43.10

Table 2: Quantitative results for various models on the New York Times test dataset

Source document
Jenson Button was denied his 100th race for McLaren after an ERS prevented him from making it to the start-line. It capped a miserable weekend for the Briton; his time in Bahrain plagued by reliability issues. Button spent much of the race on Twitter delivering his verdict as the action unfolded. 'Kimi is the man to watch,' and 'loving the sparks', were among his pearls of wisdom, but the tweet which courted the most attention was a rather mischievous one: 'Ooh is Lewis backing his team mate into Vettel?' he quizzed after Rosberg accused Hamilton of pulling off such a manoeuvre in China. Jenson Button waves to the crowd ahead of the Bahrain Grand Prix which he failed to start Perhaps a career in the media beckons Lewis Hamilton has out-qualified and finished ahead of Nico Rosberg at every race this season. Indeed Rosberg has now beaten his Mercedes team-mate only once in the 11 races since the pair infamously collided in Belgium last year. Hamilton secured the 36th win of his career in Bahrain and his 21st from pole position. Only Michael Schumacher (40), Ayrton Senna (29) and Sebastian Vettel (27) have more. He also became only the sixth F1 driver to lead 2,000 laps. Nico Rosberg has been left in the shade by Lewis Hamilton who celebrates winning his third race of the year Kimi Raikkonen secured a record seventh podium finish in Bahrain following his superb late salvo, although the Ferrari driver has never won in the Gulf Kingdom. It was the Finn's first trip to the rostrum since the 2013 Korean Grand Prix, but his triumph brought a typically deadpan response: 'You're never happy when you finish second... I'm a bit pleased to get a result.' Sparks fly off the back of Kimi Raikkonen's Ferrari en route to finishing second in Bahrain Bernie Ecclestone was in the Bahrain paddock this weekend. He denied trying to engineer a deal for Hamilton, out of contract at the end of the season, to join Ferrari despite earlier insisting that such a move would be 'great' for the sport. The 84-year-old also confirmed that F1 would be in Azerbaijan for the first time next year, even with concerns surrounding the countrys human rights record. 'I think everybody seems to be happy,' Ecclestone said. 'There doesn't seem to be any big problem there. There's no question of it not being on the calendar. It's going to be another good race. Formula One supremo Bernie Ecclestone speaks to Nico Rosberg ahead of the Bahrain Grand Prix
Ground truth summary
Button denied 100th race start for McLaren after ERS failure. Button then spent much of the Bahrain Grand Prix on Twitter delivering his verdict on the action as it unfolded. Lewis Hamilton has out-qualified and finished ahead of Mercedes team-mate Nico Rosberg at every race this season. Bernie Ecclestone confirms F1 will make its bow in Azerbaijan next season.
ML, with intra-attention (ROUGE-1 41.58)
Button was denied his 100th race for McLaren. ERS prevented him from making it to the start-line. The Briton. He quizzed after Nico Rosberg accused Lewis Hamilton of pulling off such a manoeuvre in China. Button has been in Azerbaijan for the first time since 2013.
RL, with intra-attention (ROUGE-1 50.00)
Button was denied his 100th race for McLaren after an ERS prevented him from making it to the start-line. It capped a miserable weekend for the Briton. Button has out-qualified. Finished ahead of Nico Rosberg at Bahrain. Lewis Hamilton has. In 11 races. . The race. To lead 2,000 laps. . In. . . And. .
ML+RL, with intra-attention (ROUGE-1 44.00)
Button was denied his 100th race for McLaren. The ERS prevented him from making it to the start-line. Button was his team mate in the 11 races in Bahrain. He quizzed after Nico Rosberg accused Lewis Hamilton of pulling off such a manoeuvre in China.

Table 3: Example from the CNN/Daily Mail test dataset showing the outputs of our three best models after de-tokenization, re-capitalization, replacing anonymized entities, and replacing numbers. The ROUGE score corresponds to the specific example.

Model	Average readability
ML	7.88
RL	5.43
ML+RL	**8.15**
Ground truth	9.85

Table 4: Comparison of human readability scores on a random subset of the CNN/Daily Mail test dataset. All models are with intra-decoder attention.

not know which summaries come from which model or which one is the ground truth. A score from 1 to 10 is then assigned to each summary, 1 corresponding to the lower level of readability and 10 the highest.

Results: Our human evaluation results are shown in Table 4. We can see that even though RL has the highest ROUGE-1 and ROUGE-L scores, it produces the least readable summaries among our experiments. The most common readability issue observed in our RL results, as shown in the example of Table 3, is the presence of short and truncated sentences towards the end of sequences. This confirms that optimizing for single discrete evaluation metric such as ROUGE with RL can be detrimental to the model quality.

On the other hand, our RL+ML summaries obtain the highest readability scores among our models, hence solving the readability issues of the RL model while also having a higher ROUGE score than ML. This demonstrates the usefulness and value of our RL+ML training method for abstractive summarization.

7 Conclusion

We presented a new model and training procedure that obtains state-of-the-art results in text summarization for the CNN/Daily Mail, improves the readability of the generated summaries and is better suited to long output sequences. We also run our abstractive model on the NYT dataset for the first time. We saw that despite their common use for evaluation, ROUGE scores have their shortcomings and should not be the only metric to optimize on summarization model for long sequences. We believe that our intra-attention decoder and combined training objective could be applied to other sequence-to-sequence tasks with long inputs and outputs, which is an interesting direction for further research.

References

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

Qian Chen, Xiaodan Zhu, Zhenhua Ling, Si Wei, and Hui Jiang. 2016. Distraction-based neural networks for modeling documents. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI-16)*. pages 2754–2760.

Jianpeng Cheng, Li Dong, and Mirella Lapata. 2016. Long short-term memory-networks for machine reading. *arXiv preprint arXiv:1601.06733* .

Sumit Chopra, Michael Auli, Alexander M Rush, and SEAS Harvard. 2016. Abstractive sentence summarization with attentive recurrent neural networks. *Proceedings of NAACL-HLT16* pages 93–98.

Carlos A Colmenares, Marina Litvak, Amin Mantrach, and Fabrizio Silvestri. 2015. Heads: Headline generation as sequence prediction using an abstract feature-rich space. In *HLT-NAACL*. pages 133–142.

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

Katja Filippova and Yasemin Altun. 2013. Overcoming the lack of parallel data in sentence compression. In *EMNLP*. Citeseer, pages 1481–1491.

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

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*. pages 1693–1701.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9(8):1735–1780.

Kai Hong, Mitchell Marcus, and Ani Nenkova. 2015. System combination for multi-document summarization. In *EMNLP*. pages 107–117.

Kai Hong and Ani Nenkova. 2014. Improving the estimation of word importance for news multi-document summarization-extended technical report .

Hakan Inan, Khashayar Khosravi, and Richard Socher. 2016. Tying word vectors and word classifiers: A loss framework for language modeling. *arXiv preprint arXiv:1611.01462* .

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* .

Junyi Jessy Li, Kapil Thadani, and Amanda Stent. 2016. The role of discourse units in near-extractive summarization. In *17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*. page 137.

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

Wang Ling, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiskỳ, Andrew Senior, Fumin Wang, and Phil Blunsom. 2016. Latent predictor networks for code generation. *arXiv preprint arXiv:1603.06744* .

Chia-Wei Liu, Ryan Lowe, Iulian V Serban, Michael Noseworthy, Laurent Charlin, and Joelle Pineau. 2016. How not to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. *arXiv preprint arXiv:1603.08023* .

Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *ACL (System Demonstrations)*. pages 55–60.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843* .

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. pages 3111–3119.

Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. 2017. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. *hiP (yi= 1— hi, si, d)* 1:1.

Ramesh Nallapati, Bowen Zhou, Çağlar Gülçehre, Bing Xiang, et al. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. *arXiv preprint arXiv:1602.06023* .

Joel Larocca Neto, Alex A Freitas, and Celso AA Kaestner. 2002. Automatic text summarization using a machine learning approach. In *Brazilian Symposium on Artificial Intelligence*. Springer, pages 205–215.

Mohammad Norouzi, Samy Bengio, Navdeep Jaitly, Mike Schuster, Yonghui Wu, Dale Schuurmans, et al. 2016. Reward augmented maximum likelihood for neural structured prediction. In *Advances In Neural Information Processing Systems*. pages 1723–1731.

Benjamin Nye and Ani Nenkova. 2015. Identification and characterization of newsworthy verbs in world news. In *HLT-NAACL*. pages 1440–1445.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*. volume 14, pages 1532–1543.

Ofir Press and Lior Wolf. 2016. Using the output embedding to improve language models. *arXiv preprint arXiv:1608.05859* .

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

Steven J Rennie, Etienne Marcheret, Youssef Mroueh, Jarret Ross, and Vaibhava Goel. 2016. Self-critical sequence training for image captioning. *arXiv preprint arXiv:1612.00563* .

Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. *arXiv preprint arXiv:1509.00685* .

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

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

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

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*. pages 3104–3112.

Arun Venkatraman, Martial Hebert, and J Andrew Bagnell. 2015. Improving multi-step prediction of learned time series models. In *AAAI*. pages 3024–3030.

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

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning* 8(3-4):229–256.

Ronald J Williams and David Zipser. 1989. A learning algorithm for continually running fully recurrent neural networks. *Neural computation* 1(2):270–280.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144* .

Yinfei Yang and Ani Nenkova. 2014. Detecting information-dense texts in multiple news domains. In *AAAI*. pages 1650–1656.

Wenyuan Zeng, Wenjie Luo, Sanja Fidler, and Raquel Urtasun. 2016. Efficient summarization with read-again and copy mechanism. *arXiv preprint arXiv:1611.03382* .