Just News It: Abstractive Text Summarization with a Pointer-Generator Transformer

Vrinda Vasavada, Alexandre Bucquet
Department of Computer Science
Stanford University
`vrindav@stanford.edu, bucqueta@stanford.edu`
`https://github.com/vrindav/just-news-it`
Mentor: Sahil Chopra

Abstract

In this paper, we tackle the task of abstractive text summarization for news articles. We first evaluate See et al.'s pointer-generator model [1] on complex pieces (such as op-eds) to understand potential areas of improvement. We then extend the model by experimenting with transformers [2] instead of traditional LSTM encoders and decoders, strictly avoiding input sentence preprocessing unlike most other recent work [3, 4]. Through our experiments, we conclude that LSTM summarizers lack generalization capacities to more unstructured articles such as op-eds. We also outline the difficulties of training a transformer model for abstractive text summarization. While we do not succeed in training a successful transformer architecture for this task, we hope that this paper provides both insight about the difficulty of this task and direction for future work on abstractive summarization.

1 Introduction

Text summarization is one of the core applications of Natural Language Processing, especially since newer and faster machines can help train larger neural architectures. In this task, given input text x, the model must write a summary y which is shorter than x but contains the main ideas and information of x. In order to successfully complete this task, a model must be able to capture a deep understanding of language to output text that (1) is understandable and related to the passage and (2) contains most of the relevant pieces of information from the input text. As a result, this makes text summarization a great benchmark for evaluating the current state of language modeling and language understanding.

There are two types of text summarization techniques, *extractive* and *abstractive*. In extractive summarization, the summary y is a subset of x, which means that all words in y come from the input x. In contrast, abstractive summaries contain both words from x and from the rest of the vocabulary. Much previous work has highlighted that it is difficult for an extractive model to produce detailed, human-like summaries. Thus, an abstractive model is required for the summarization of longer, more complex documents, and, for this reason, we focus on the latter [5].

In the past few years, LSTM encoder/decoder architectures have obtained state-of-the-art results in text summarization. Examples of successful architectures include See et al.'s pointer-generator network [1]. In this paper, we first test the pointer-generator's performance on complex op-ed pieces and highlight areas of improvement. As transformers promise more nuanced modeling than recurrent units, we then explore such architectures for the task of abstractive summarization. Unlike previous work, we decide to use transformers without input sentence trimming, as they should be able to capture the long-range dependencies that LSTMs fail to understand [3, 4], . Ultimately, we propose the model architecture for the *transpointer* (a combined transformer and pointer-generator) and outline the numerous training experiments we conducted with it. We detail the issues we faced and describe areas for future improvement for such a model's success.

2 Related Work

In *Get To The Point: Summarization with Pointer-Generator Networks*, See et al. introduce the use of pointer-generator networks for the task of text summarization [1]. This paper points out two shortcomings in neural sequence-to-sequence models for this task and introduces methods to address these. First, See et al. address the potential inaccuracy of generated summaries resulting from uncommon or out-of-vocabulary words in the original text. They do so by computing the *generation probability* for every timestep using the context vector, the decoder hidden state, and the input to the decoder. This generation probability is then used to compute a new adjusted probability distribution over the vocabulary. It allows the model to copy words from the source and thus enables the preservation of factual details. Second, See et al. address the repetition of words and phrases in generated summaries. They do so by adapting the concept of *coverage* from Neural Machine Translation into this context. Specifically, the coverage vector represents an unnormalized distribution over the source document words that represents how much each word has contributed to the output summary thus far. This is then used to disincentivize the model from attending to portions of the source that have already been summarized. In this paper, we will use this model as our baseline to evaluate the current state of text summarizers that use recurrent encoder/decoder structures.

Much recent work has extended See et al.'s work by incorporating more nuanced forms of attention into the model. In *Bottom-Up Abstractive Summarization*, Gehrmann et al. reuse the concept of generation probability and attempt to solve one of the main issues that abstractive models often face: poor content selection from the source text [4]. They do so by applying a selection mask over the source document, then using this masked version as input to the model.

Then, in *Generating Wikipedia by Summarizing Long Sequences*, Liu et al. incorporate a transformer [3]. These models are attractive because they can be trained in parallel and only use attention mechanisms to compute hidden representations of the input and output sequences [2]. Liu et al. use a heuristic to determine which words to feed into the transformer, experimenting with *tf-idf* and *SumBasic* among others (Liu et al.). These methods assign scores to words or sentences in the input, and the proposed architecture only keeps the first n candidates to feed into the transformer network.

To a certain extent, these attempts at selecting subsets of words from the source document resemble extractive summarization. While we believe this would work well for certain, more traditionally structured news types, we hypothesize that specifically with op-ed pieces masking words out of the input will make summarization even more difficult. In addition, we seek to test whether it is feasible to reduce model complexity by maintaining the raw input and entirely replacing both the encoder and decoder with a transformer.

3 Approach

As our baseline, we used the pointer-generator network as described by See et al. [1]. We found an implementation of this network in PyTorch [6]. Since this implementation used an outdated version of PyTorch (0.4.1), we updated the codebase before training it. We also found that the data cleaning and loading steps in the implementation did not exactly match the format of our data, and we spent some time writing our own scripts for this.

Our second step was to find a working implementation of a transformer. For this project, we used Y. Huang's PyTorch implementation of a transformer network [7]. Before adding the module to our model, we decided to attempt to replicate the results claimed by the author (for the task of Neural Machine Translation from English to German) as a test for the code. We were able to reproduce those results fairly easily after checking that the code provided correctly implemented a transformer model.

In this paper, we combine these two implementations into new models. We first adapt the transformer for the task of text summarization, without adding any other modules to our model. We thus use a transformer encoder and decoder, feeding into the encoder the input sequence as well as positional encodings (as in [2]). At every timestep, the decoder predicts the next word using self-attention and attention with respect to the encoder state.

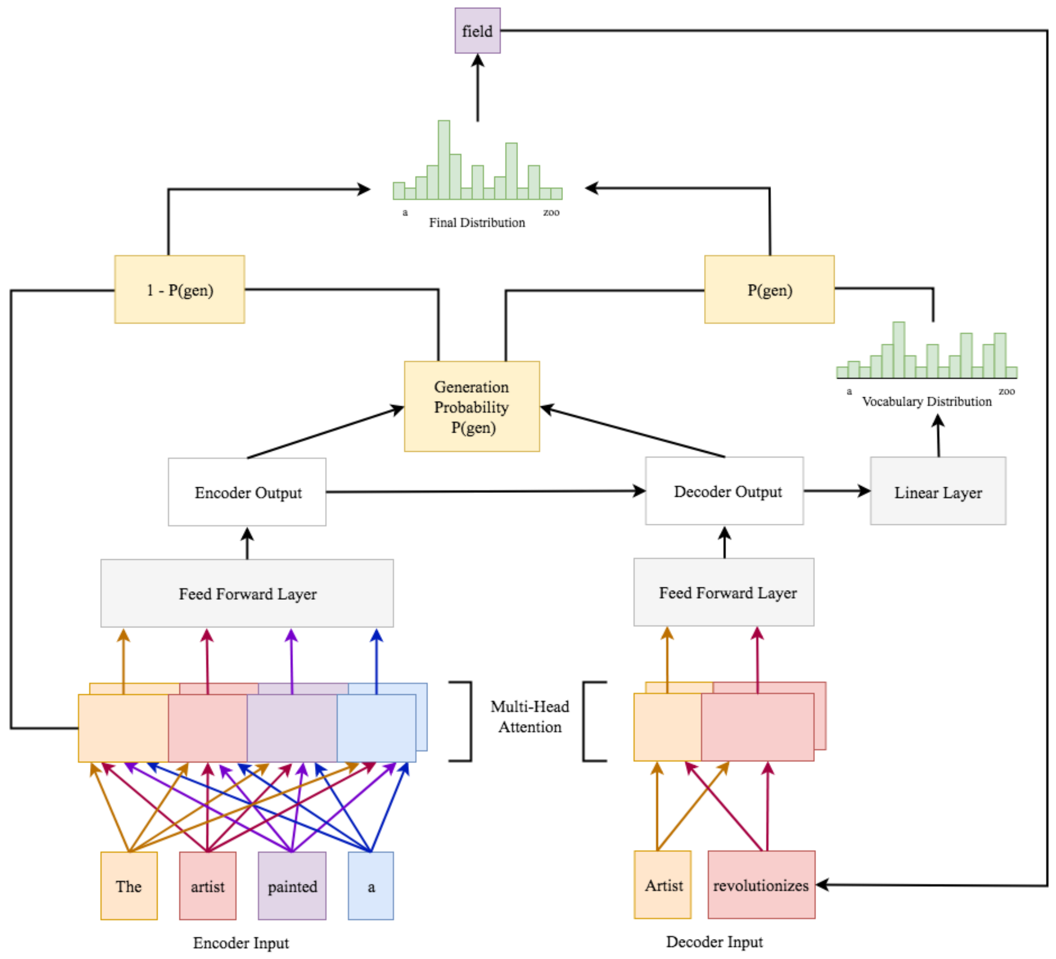

Figure 1: Transpointer Model Architecture

The main model of this paper, however, is the *transpointer* (Figure 1). This model combines features from both the pointer-generator and the transformer as follows. We first feed the input sequences $(x_0, x_1, x_2, ..., x_n)$ into a transformer encoder with positional encodings to compute a self-attended embedding x of the input text.

Then, at every step of the decoding process, a transformer decoder uses the summary generated so far y_t with positional encodings to generate a self-attended representation of the decoder state s_t. We use s_t and x to compute a final representation of the decoder state d_t attending to the encoder state. By passing d_t through a linear layer, we compute a probability distribution for the next word P_{vocab} over the vocabulary.

Then, we compute the generation probability p_{gen} as in [1] as follows:

$$p_{gen} = \sigma(w_x x + w_y y_t + w_d d_t + b_{gen})$$

where w_x, w_y, w_d and b_{gen} are parameters learned during training.

Finally, we use the attention distribution a^t of the encoder with respect to the decoder (that we computed while deriving d_t) along with p_{gen} and P_{vocab} to compute our final probability distribution $P(w)$:

$$P(w) = p_{gen} P_{vocab}(w) + (1 - p_{gen}) \sum_{w_i = w} a_i^t$$

where $P(w)$ and $P_{vocab}(w)$ are the respective probabilities of w being the next word under the probability distributions P and P_{vocab}.

4 Experiments

4.1 Data

For this paper we use the *CNN/Daily Mail* dataset [8], which contains news articles (781 tokens on average) paired with multi-line summaries (56 tokens on average). Unlike some previous work, where entities in the source text were anonymized, we use the raw news article texts. For example, previous work might replace proper nouns such as "Stanford University" with a unique placeholder such as $\{\text{entity}_0\}$. Instead, we use the raw dataset without anonymization to test our model's ability to preserve entity names and factual accuracy.

To assess the generalization capability of our models, we also manually collected 20 op-eds from the *New York Times* [9]. Since our goal is to evaluate the model on only a few pieces, we focus on the first few pages of results, giving us a wide range of themes for the collected articles.

4.2 Evaluation Method

To evaluate the generated summaries, we use ROUGE score, commonly used in summarization tasks. Specifically, we report F1 scores (combined precision and accuracy) for ROUGE-1, ROUGE-2, and ROUGE-L. ROUGE-1 and ROUGE-2 are similar to the BLEU score, as they measure the unigram and bigram similarity between the reference translation and the model-generated one. ROUGE-L measures the longest common subsequence between the two translations.

We also use qualitative metrics by manually evaluating summaries generated by our model. We do this on two datasets: a held-out test set from the *CNN/DailyMail* corpus, and op-eds from the *New York Times* (as described above). We analyze the predicted summaries to classify the strengths and shortcomings of the pointer-generator network.

4.3 Results

4.3.1 Pointer-Generator

We trained the pointer-generator network for $100{,}000$ iterations using a learning rate of 0.15. We used a batch size of 8 and a beam size of 4. The hidden state had dimension 256, and the embeddings were of size 128. The vocabulary size was $50{,}000$. This took 22 hours to train.

The scores claimed by See et al. [1], the GitHub repository whose code we adapted [6], and our self-trained model are shown in Table 1.

Model	ROUGE-1	ROUGE-2	ROUGE-L	Train Time	# Batches
See et al. (TF) [1]	39.53	17.28	36.38	4-8 days	N/A
See et al. (Pytorch) [6]	39.07	16.97	35.87	N/A	700k
Updated version of [1]	24.96	7.78	22.90	22 hours	100k

Table 1: Comparison of ROUGE scores and training details between summarization models

Note that while we were not able to reproduce the results of either work, this is understandable due to a wide difference in computation time and resources. Our trained model architecture should still be able to show the main strengths and weaknesses of current recurrent architectures.

Sample outputs from the pointer-generator network we trained are shown in Tables 2 and 3.

4.3.2 Transformer and Transpointer

As detailed in the next sections, we were not able to achieve good results with either transformer architecture. Since the plain transformer faced similar issues, we will only analyze the results of the transpointer model.

Example 1
Reference: "furious 7" pays tribute to star paul walker, who died during filming. vin diesel: "this movie is more than a movie." "furious 7" opens friday .
Model Output: paul walker 's death in november 2013 at the age of 40 after a car crash. the actor was on a run at the age of 40 after a car crash in november 2013. the actor was on broken from filming "furious 7" on friday.

Example 2
Reference: hardy was convicted of domestic abuse against ex-girlfriend nicki holder and was suspended from the dallas cowboys for 10 days by the nfl. charges were eventually dropped after holder could not be located when hardy's lawyers appealed the decision and asked for a jury trial. this week he got stuck in his bentley in deep flash flood waters in dallas. hardy was forced to abandon his car and it was towed away hours later.
Model Output: greg hardy was suspended from 10 nfl games. he was fined $ [UNK] a week. he was dropped by his previous team, the sheriff says.

Example 3
Reference: museum: anne frank died earlier than previously believed. researchers re-examined archives and testimonies of survivors. anne and older sister margot frank are believed to have died in february 1945.
Model Output: anne frank died of [UNK] in a nazi concentration camp at the age of 15 . she had been abducted in the age of 15 years ago, the [UNK] concentration camp in poland. she had been abducted in the age of 15.

Table 2: Example outputs from the pointer-generator on the test set

We trained our initial transpointer model for text summarization using 8 attention heads with 6 layers each, query and context embedding dimensions $d_k = d_q = 64$, and contextualized word embeddings of size $d_{model} = 512$. During training, we used a vocabulary of size 50,000 and capped the maximum article length at 500 tokens.

During training, we use the Adam optimizer (as suggested in [2]) with batch size of 8 and parameters $\beta_1 = 0.9$, $\beta_2 = 0.98$ and $\epsilon = 10^{-9}$.

We varied the learning rate over the course of training as follows (where d_{model} is defined as above):

$$lr = d_{model}^{-0.5} \cdot \min(\text{stepnum}^{-0.5}, \text{stepnum} \cdot 4000^{-1.5})$$

This corresponds to increasing the learning rate linearly for the first $4,000$ training steps before decreasing proportionally to the inverse square root of the step number.

Throughout this project, our transformer models were prone to underfitting (see Section 5.2). We conducted a hyperparameter search as summarized in Table 4. For each configuration, the model plateaued after 1-2 hours of training. In all our experiments with the transformer model, we found that the mean cross-entropy loss stayed around seven even after many iterations (see Table 4 and Figure 2).

5 Discussion

5.1 Pointer-Generator

5.1.1 Evaluation on test set

As seen in Table 1, we were restricted by computation and time resources to replicate results by See et al. We believe that our trained model can still help pinpoint main sources of errors in generated summaries.

While the model seems to perform well for some inputs, we can identify the following main types of errors from Table 2.

1. We noticed that the summaries of the model tend to be shorter than the "gold" summaries. While this may simply be fixed at decode time, the issue might be that the model is too

Example 4
First Paragraph Snippet: I was 15 when I started smoking, and so were most of my friends. (...) Because cigarettes were both forbidden and easy to get: ten quarters in a cigarette vending machine, which you could still find in most pizza joints and doughnut shops in suburban New Jersey in the early 1990s.
Model Output: [UNK] [UNK] , who was 15 when i started smoking , has been found in the past 1990s . he says the idea of the most people who [UNK] start smoking by the end of the [UNK] .

Example 5
Article Snippet: (...) Wabtec is willing to keep the existing workers at the current average of $35 per hour. (...)
Model Output: new: [UNK] is willing to keep the existing workers at the current average of [UNK] per hour.

Example 6
First Paragraph: The current focus on North Korea's growing arsenal obscures the fact that the most likely trigger for a nuclear exchange could be the conflict between India and Pakistan. Long among the world's most antagonistic neighbors, the two nations clashed again last week before, fortunately, finding the good sense to de-escalate. The latest confrontation, the most serious between the two nations in more than a decade, gave way to more normal pursuits like trade at a border crossing and sporadic cross-border shelling. (...)
Model Output: [UNK] [UNK] : the [UNK] are the fact that the most likely trigger for a nuclear exchange could be the conflict between india and pakistan. he says the most serious between the two nations in more than a decade, the most serious between the two nations in more than a decade.

Table 3: Example outputs from the pointer-generator on NYT opeds

Experiment	**Heads**	**Layers**	d_k, d_q **size**	**Warmup steps**	**Average loss past 2000 batches**
1	8	6	64	4000	7.301
2	16	6	64	4000	7.285
3	8	12	64	4000	7.292
4	8	6	128	4000	7.301
5	8	6	64	2000	7.286
6	8	6	64	6000	7.299
7	16	12	128	4000	7.248

Table 4: Hyperparameter Search on the Transpointer Model

abstractive, and cannot understand multiple distinct concepts from different locations in the input text at once. In example 2, while the model was able to report the main information of the article (Hardy's sentence), it failed to capture concepts from the second half of the article.

2. The model also seems to struggle to attribute entities and identify sources. For instance, in example 3 the model has trouble evaluating whether 15 years is the age of Anne Frank when she was abducted or when she died, or how long ago she was abducted.

3. On some inputs, the model outputs a sound summary, but one that does not contain the most relevant information from the input. In example 1, while the article focuses on the movie *Fast and Furious 7*, the model's summary reports mostly information about the details of Paul Walker's accident.

While errors of type 3 are likely due to the undertraining of the model, all errors show that there should be a clear advantage in using stronger attention mechanisms like those deployed in transformers. Such mechanisms could help bring information from different parts of the input text into a single summary (global attention), while ensuring the soundness of the summary (local attention).

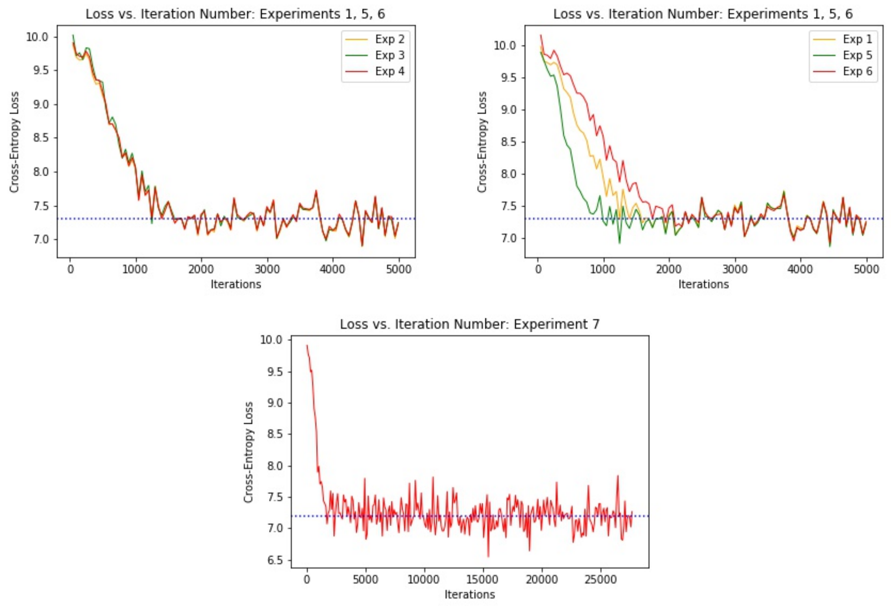

Figure 2: Learning Curves for Different Hyperparameter Values

5.1.2 Evaluation on Op-eds

Moreover, we found that our pointer-generator model did not generalize to op-eds very well. On the 20 input-output pairs we collected, we found the following recurring mistakes displayed in Figure 3.

1. On several inputs such as example 4, the pointer-generator seems to collect only information from the first paragraph. This might come from the fact that in news articles (used to train of the model), most of content of the piece is summarized in the first paragraph.
2. The network seems to struggle with unknown tokens (as in examples 4 and 5). This is odd since the pointer-generator's main goal is to reduce the number of unknown tokens in output summaries.
3. On some pieces like example 5, the model returns a full sentence from the article, even though that sentence is not key to the understanding of the piece. While this article focuses on union protests against several companies, the proposed summary only includes information about a single example. This shows that the generation probability can be improved.

However, we also found that the model was still able to capture some key details for some of the op-eds. In example 6, the pointer-generator picked up on the importance of the conflict between India and Pakistan before getting sidetracked. It is important to note that this op-ed had the most similar structure to other news articles out of the 20 we collected.

Overall, summarizing op-eds is a harder task than summarizing news articles in general, due to the lack of structure of some pieces. This leads See et al.'s model [1] to make some errors, that could potentially be fixed using stronger attention mechanisms.

5.2 Transformers

Transformers, which should be able to model long-term dependencies in the input, promise to be the next step in many NLP tasks. Yet as seen in Figure 2, the transpointer failed to obtain good performances even after lengthy training times.

After investigating these errors, we found that while the model was able to overfit to 1% of the training data, it did not manage to learn anything on the whole train set. Indeed, we found that a typical model (trained for 22 hours) predicted the same output summary at decode time regardless of the input that was fed into it. Despite the fact that the encoder states, decoder states, and logits were different for every input, these differences were not large enough to push the model to predict different summaries. More specifically, the model appeared to have learned the average word distribution for the entire dataset, thus never committing to a single prediction. This can be seen in Figure 2, as all experiments struggled to learn past this local optimum.

This underfitting might come from the length of the sequences. As discussed in Section 2, past applications of transformers in NLP have all used these models on shorter sentences. This allows the transformer to receive trimmed and already curated input to make its task easier. However, no text summarization model exists that takes in full input sequences, which seems counter-intuitive since the promise of using transformers is to understand the inter-dependencies in text input. As a result, it seems like transformer models are yet to be fully understood, especially in situations where the input sequence is long and potentially unstructured, while containing a large amount of information. This warrants further analysis on the training and behavior of transformers on such inputs.

6 Conclusion

From our analysis of the subtask of op-ed article summarization, we conclude that while the current state-of-the-art recurrent models perform well on traditional news articles, they do not generalize well to more difficult subtasks. Much related work points towards including extractive steps to trim the input before training an abstractive summarization model. However, as described in this paper, we believe that masking the input beforehand will further degrade performance on more difficult, unstructured pieces such as op-eds and dilute the simplicity of the summarization pipeline.

For this reason, we attempted to implement a transformer for this task. The promise of transformers is to model long-term dependencies in the input. Even in text summarization, one could hope that such architecture is able to understand the structure (or lack thereof) of a given piece of text, and able to leverage this to output a sensible summary. Yet as seen in Sections 4.3 and 5 in this paper, transformer models failed to obtain good performances even after lengthy training times. Thus, we believe that transformers still have many nuances that need to be understood in order to successfully train them on long inputs and adapt them for the task of abstractive text summarization.

7 Future Work

In the future, we would investigate different hyperparameters and architectures for transformers models applied to text summarization. In this paper, we could only experiment with a few of the many possible combinations due to resource and time constraints.

In order to understand how transformers behave on long inputs, we would also consider such architectures applied to the easier task of extractive summarization. It is possible that developing models that can first point to the right place in the text without having to generate new words might provide more insights on how to design effective transformer architectures for the more challenging task of abstractive summarization.

We would also experiment with attention windows, in order to reduce the number of neighbor words that are considered in attention computations. While this will reduce the potential at modeling long-term dependencies, it might provide more expressive representations of the input text. These attention restrictions could use a fixed window size, or process each paragraphs in the input separately.

For these reasons, we are still optimistic about potential applications of transformers to this problem.

Acknowledgements

We would like to thank our mentor Sahil Chopra for his continued support and feedback throughout this project. We would also like to thank the whole course staff of CS224N for building the resources necessary for the completion of this work.

References

[1] C. M. A. See, P. Liu, “Get to the point: Summarization with pointer-generator networks,” *CoRR*, 2017.

[2] A. V. et al., “Attention is all you need,” *Conference on Neural Information Processing Systems*, 2017.

[3] P. L. et al., “Generating wikipedia by generating long sequences,” *ICRL*, 2018.

[4] A. R. S. Gehrmann, Y. Deng, “Bottom-up abstractive summarization,” *Conference on Empirical Methods in Natural Language Processing*, 2018.

[5] A. See, “Taming rnns for better summarization,” 2017.

[6] A. Kumar, “Pointer summarizer pytorch implementation (github repository),” 2017.

[7] Y. Huang, “Attention is all you need pytorch implementation (github repository),” 2017.

[8] R. N. et al., “Cnn/dailymail dataset,” *CoNLL*, 2016.

[9] *New York Times*.