More Abstractive Summarization with Pointer-Generator Networks

Stanford CS224N Custom Project

Michael A. Lin
Department of Mechanical Engineering
Stanford University
mlinyang@stanford.edu

Abstract

Abstractive text summarizations are found frequently in our daily lives from web browser searches to news feeds. Neural approaches to perform this task have made significant progress recently, however, we are still far from producing the level of abstractive summaries humans do. One particular challenge is that handling of out-of-vocabulary words, particularly with the use of pointer generator networks, causes the network to copy words and sentences from the input excessively resulting in less abstractiveness. In this work we investigate the mechanism that allows this copying in the architecture introduced by See et al. [1]. We formulate a new model that produces comparable performance in terms of ROUGE scores and is able to produce significantly more novel n-grams (at least 30% increase) than the baseline model.

1 Key Information to include

- Mentor: Alexandre Matton
- External Collaborators (if you have any): N/a
- Sharing project: N/a

2 Introduction

Text summarization is the task of condensing information from a piece of text to a shorter version while maintaining the main content and meaning of the source text. We encounter summaries frequently in our daily lives: academic abstracts, news feeds, Google search results, library content browsing, etc. Natural language processing systems have approached this problems with two methods: extractive and abstractive summarization. Extractive method construct a summary by picking the words and sentences directly from the source text. Abstractive method extracts the meaning conveyed in the source and summarizes it with similar or novel words or phrases.

Successes of encoder-decoder RNN models in natural language translation has led to the use of generative methods for text summarization [2, 3]. The task of summarization is cast as mapping a long sequence of words in a source document to a target sequence called summary. Learning this mapping of many tokens to few tokens is difficult and some of the main problems that arise are avoiding repetitive decoded sequences or handling out-of-vocabulary (OOV) words. To tackle these problems See et al. introduced some novel changes to the vanilla RNN+attention architecture [1]. In order to handle OOV words, a novel pointer-network architecture is used where a soft switch allows the decoder to choose to take words from the source rather than generating a <unk> token. For reducing the repetitiveness of the output they also introduced a novel method of penalizing for repetitive attention to the same locations of the attention vector. The model produced by this work significantly outperformed the state-of-the-art models at the time in terms of ROUGE points.

Although the method presented by See et al. showed significant improvement over previous implementations of abstractive text summarization, its mechanism to handle OOV also made the results less abstractive as it allowed for a path to copy from the source without much restrictions. As reported, 35% of the outputs were copies of entire article sentences. Even more recent work on using pre-trained transformer models face the same problem [4]. This excessive copying defeats the goal of abstractive summarization of making summaries with novel words or phrases. We are interested in whether it is possible to bias the pointer network to only copy words from source if they are OOV, so as to prevent it from being able to copy entire sentences.

In this work we investigate further into this copying behavior and potentially how the pointer-generator architecture might be related to it. Particularly, we look into the value of the soft-switch p_{gen} and how it affects the rate at which the model copies words from the source. We hypothesize that although the pointer-generator network is intended to activate mostly when the source is an OOV word, it actually learns to excessively copy words from the source. Extractive summarization is generally less expensive and grammatically correct so it would make sense for the network to learn a model that likes to copy and paste. Particularly we look into the relationship between p_{gen} and when n-grams are contained in the source or not.

Inspired by this problem of excessive copying, we also explore a novel method of including a new loss term during training to discourage the model from copy words. This loss term formulation penalizes non-zero values assigned to attention vector elements that are not OOV. Our results show that although this method yields lower ROUGE values, the model is able to increase the amount of novel n-grams it generates.

3 Related Work

3.1 Extractive Summarization

Extractive summarization create a summary by identifying the most salient sentences in the source text. In that view, extractive summarizers models are sentence classifiers that score each sentence on their saliency. SummaRuNNer by Nallapati et al. [5] is an early and successful work in this area that adopted LSTMs with attention to tackle this problem. Narayan et al. approached the problem with Reinforcement Learning by training to directly maximize the ROUGE score of the output [6]. Liu et al. has achieved the state-of-the-art ROUGE score results by adapting the BERT transformer model to extractive summarization [4].

3.2 Abstractive Summarization

Abstractive summarization approaches the problem more like a sequence-to-sequence mapping where the encoder maps input tokens to a intermediate representation (hidden states) and then a decoder sequentially rolls out the sentence token-by-token. This structure allows us to generate the output based on a language model such that we could potentially create new sentences to describe a meaning or idea. Abstractive summarization is a more useful approach as it is able to compress ideas from multiple sentences into less sentences. In that sense, it technically can achieve more compact summaries than extractive methods. Rush et al. was one of the first to use neural encoder-decoder architecture for text summarization. Nallapati et al. improved on this work by adapting it to larger data sets (CNN/Daily Mail) and by encoding longer sequences of input [2]. As mentioned previously, the work by See et al. has been a break through in this area by reducing repetitiveness and handling OOV words with pointer networks. Other work have approached the problem with reinforcement learning [7], or multi-agent encoders [8]. Recent work on abstractive summarization have produced state-of-the-art performance uses BERT transformer adapted for both extractive and abstractive summarization [4].

4 Approach

4.1 Investigating Copying

The first part to this work investigates the cause of copying by the model proposed in [1]. The architecture used by See et al. builds on top of a single-layer bidirectional LSTM with an attention layer as the one presented in [9]. The component of the architecture that handles copying of OOV words is a hybrid between the baseline encoder-decoder RNN and pointer-generator network [10]. As explained previously, this novel architecture allows pointing from the attention vector in order to copy OOV words that the encoder is unable to map because of their absence in the embedding space. Copying is done through a soft switch variable p_{gen} defined as follows:

$$p_{\text{gen}} = \sigma(w_{h^*}^T h_t^* + w_s^T s_t + w_x^T x_t + b_{\text{ptr}}) \tag{1}$$

where h_t^* is the context vector, s_t is the decoder state, x_t is the decoder input and $w_{h^*}^T$, w_s^T, w_x^T and b_{ptr} are learnable parameters.

The final output of the model is a distribution over the vocabulary

$$P(w) = \underbrace{p_{\text{gen}} P_{\text{vocab}}(w)}_{P_{\text{DEC}}(w)} + \underbrace{(1 - p_{\text{gen}}) \sum_{i:w_i=w} a_i^t}_{P_{\text{PTR}}(w)} \tag{2}$$

This equation's first term is the contribution from the decoder output (we will refer to this as P_{DEC} term) and the second term is the contribution from the pointer from the attention (we will refer to this as P_{PTR} term). Note that for the addition of the two terms to be possible the vocabulary size is augmented to the size of max vocabulary size plus number of OOV words (e.g. 50k + 10 if there are 10 OOV words in the source). In order to find when the model is copying we can compare the values of P_{DEC} and P_{PTR} that correspond the words that are copied from source and the words that are novel. We will look into the relationship of these two terms for 1-gram, 2-gram, 3-gram, 4-gram and full sentences. We also look into the value of p_{gen} for novel vs copied n-grams as this is also an indicator of the tendency to copy words from the input.

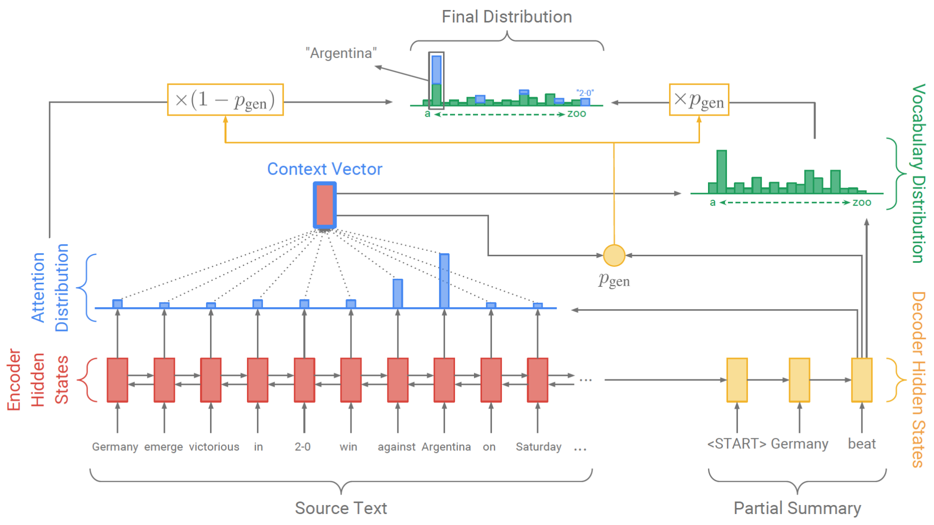

Figure 1: Pointer Network Architecture. Figure taken from See et al. [1]

4.2 Copy Loss Mechanism

For the second part of this work, we propose a method to yield a new model that is discouraged from copying words that are not OOV. The overall architecture is very similar to the one introduced by See et al. shown in Fig. 1. We formulate a new loss term to penalize for the copying of words that are not OOV. We take the sum of output of the pointer-generator $P_{\text{PTR}}(w)$ for each word that is not an OOV word. To do this we build a vector $y_{<\text{unk}>}$ the size of the vocabulary plus number of OOV words where its value is 1.0 at indexes that corresponds to OOV and 0.0 otherwise. We refer to this loss term as "copy loss" and it is formulated as follows:

$$\begin{aligned}\text{copyloss} &= (1 - y_{\text{<unk>}})^T P_{\text{PTR}}(w) && (3)\\ &= (1 - y_{\text{<unk>}})^T (1 - p_{\text{gen}}) \sum_{i:w_i=w} a_i^t && (4)\end{aligned}$$

We add this to the total loss term used by See et al. (including the coverage loss) which results in the following expression for each time step t:

$$\text{loss}_t = -\log P(w_t^*) + \lambda_1 \sum_i \min(a_i^t, c_i^t) + \lambda_2 (1 - y_{\text{<unk>}})^T (1 - p_{\text{gen}}) \sum_{i:w_i=w} a_i^t \quad (5)$$

Where λ_1 and λ_2 are hyper-parameters.

The intuition behind this is that high values of $P_{\text{PTR}}(w)$ causes words to be copied over from the attention vector. However, if the words are not OOV then we should penalize this behavior.

5 Experiments

5.1 Data

We use the *CNN/Daily Mail* dataset collected by Hermann et al [11] which has been widely used in abstractive summarization work[2, 1, 4]. This corpora contains roughly a million news stories collected from scraping the website of the two news media. Articles contain 781 token on average paired with summaries of 3.75 sentences or 56 token on average. We follow the same method as Nallapati et al. and See et al. [2, 1] to split the data into 287,226 training pairs, 13,368 validation pairs and 11490 test pairs. Each pair is one article and one summary. If the source article is longer than 400 tokens we crop it to this maximum length. This source text is the input to the network and the output is a sequence of probability distributions $P(w)$. The output is then compared to the reference summary to generate a loss value. We chose not to anonymize the entities referenced in the source text to avoid prepossessing overhead.

5.2 Evaluation method

For our first experiment on investigating the baseline copy behavior, we want to define metrics that indicates whether the model is favoring copying and whether the final decoded output was indeed a result from copying from source. One metric we will look into is the values of $P_{\text{DEC}}(w)$ and $P_{\text{PTR}}(w)$ as defined in Equation 2. As the model's output distribution is the sum of these two terms, their relative values will determine whether the decoded word was mostly influenced by copying or by the vanilla RNN output. We will also look into the value of $p_{\text{gen}}(w)$ as this is the soft-switch value that scales the decoder output distribution and $1 - p_{\text{gen}}(w)$ scales the attention distribution. $p_{\text{gen}}(w)$ is a less clear metric of whether copying will actually dominate the result as the decoder output and the attention vector are distributions over sets of different cardinality, thus, their values on average are different. However, since this value scales the entire distribution it indicates whether during decoding the model favors copying from source. In contrast, the $P_{\text{DEC}}(w)$ vs $P_{\text{PTR}}(w)$ metric tells us whether the final result was mostly influenced by copy or not.

In the second experiment we want to investigate whether adding the *copyloss* term helps the model (i) generate more novel words, (ii) copy less for words that are not OOV and (ii) improve its performance in terms of ROUGE score. We will evaluate (i) by looking at what percentage of n-grams in the output summaries are also present in the input source, similar it is done by See et al. For (ii) we will use the same metrics as in the first experiments to determine whether this new model still tries to copy for non OOV words. And for (iii) we will measure the overall summary sentence performance using ROUGE (R-1, R-2 and R-L) scores as it is more widely used metric for rating sentence summaries. We used the ROUGE-1.5.5 script with a python wrapper [12] to compute ROUGE F_1 scores.

5.3 Experimental details

5.3.1 Baseline and copy loss model

We first try to replicate the results in the work by See et al. [1] but using a PyTorch implementation [13]. We train the model including pointer-generator and coverage mechanism using the same parameters: a vocabulary size of 50k, learning step with Adagrad with a rate of 0.15, an initial accumulator of 0.1, gradient clipping at a norm of 2.0 and no regularization. We also limit the length of the source document to 400 tokens and the length of the summary to 100 tokens during training and 120 tokens during testing. We use a value of 1.0 for the hyper-parameter λ of the coverage loss.

We train the model with the same parameters for 500k iterations on a computer with Intel Core i7-9700k with GPU 8-core NVIDIA GTX-2080. Training time was about 5 days. See et al. trained the model without coverage first and then added coverage for 2000 iterations to get their high ROUGE scores. We found it unnecessary to do this as we only wanted to obtain a model we could use as reference.

For the copy loss model proposed in this work we use exactly the same parameters for training, but we add the additional copyloss term as shown in Equation 5. We use a value of 1.0 for the λ_2 hyperparameter.

5.3.2 N-gram novelty vs. pointer-generator values

In test time, we record the values of p_{gen}, $P_{\text{DEC}}(w)$ and $P_{\text{PTR}}(w)$ for every decoded word in each article. We then group these values by whether the n-gram they correspond is novel or not. For 2-gram, 3-gram and so on, we take the average of the values corresponding to those words.

5.4 Results

5.4.1 Baseline model P_{DEC} vs. P_{PTR}

We have reported our results comparing values of P_{DEC} and P_{PTR} as boxplot for the baseline model in Fig. 3. Plots compare P_{DEC} and P_{PTR} corresponding to n-grams or sentences that are novel (left side of each plot) and corresponding to words that are found in the source text (right side of each plot). Each boxplot shows the median as the horizontal line inside the box, bottom and top of the box correspond to 25th and 75th percentile respectively, and whiskers show the min and max values in the data. An immediately noticeable pattern is that for all n-grams and sentence plots, the values of P_{PTR} is significantly higher than that of P_{DEC} when the decoded word is contained in the source text. This relationship is what we expected and agrees with our hypothesis that the pointer generator network portion of the architecture is related to the copy behavior.

5.4.2 Copy loss model P_{DEC} vs. P_{PTR}

Results for our copy loss model (Section 4.2) comparing values of P_{DEC} and P_{PTR} are shown as boxplots in Fig. 4. We can see that for all n-gram and sentence experiments the words that correspond to copied words do not have a higher P_{PTR} compared to P_{DEC} anymore. This shows that we were able to reduce the tendency for the model to copy words through the pointer network. Another observation is that P_{DEC} is higher than P_{PTR} for all novel n-grams, as it is not the case for the baseline model.

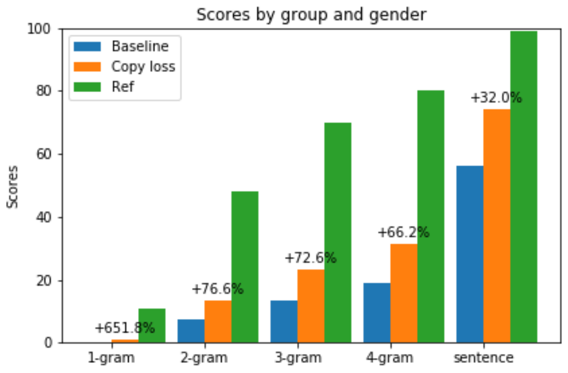

Figure 2: Plot shows percentage of novel [1,2,3,4]-grams in generated summaries from baseline model, our model and reference summary.

5.4.3 Copy loss model performance

Results for the output summaries by the copy loss model are reported in Table 1. The results were slightly worse in all ROUGE categories than the baseline model's results. This was expected as many of the reference summaries might be exact phrases from the input source and by reducing the copying behavior those copied phrases might not contribute to the high score anymore. However, if we look at our results on novelty of n-grams reported in Figure 5.4.1, we can see that

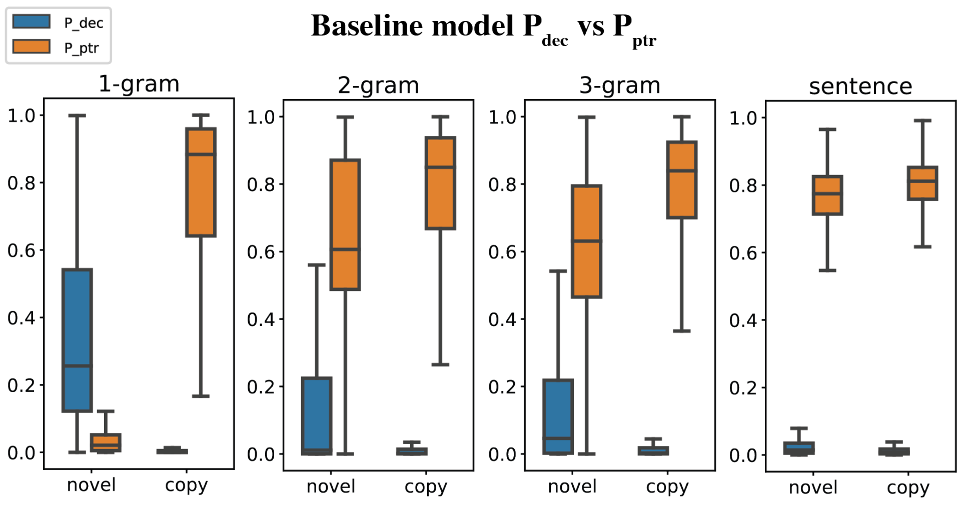

Figure 3: Comparison of P_{DEC} vs P_{PTR} for baseline model [1,2,3]-gram and sentences.

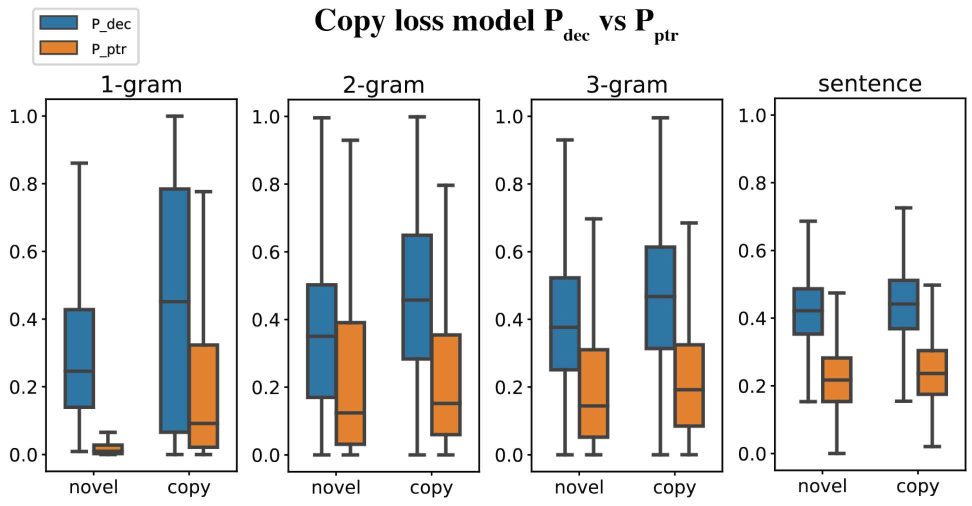

Figure 4: Comparison of P_{DEC} vs P_{PTR} for baseline model [1,2,3]-gram and sentences.

our new model successfully boosts the amount of novel n-grams compared to the baseline model. For example, 75% of the sentences are novel compared to 65% in the baseline which is a significant increase.

5.4.4 p_{gen} values for novel vs. copied words

We also report the results of p_{gen} corresponding to novel and copied words for both baseline and copy loss models. An observation is that for the 1-gram plot, p_{gen} values are higher for novel words and and lower for copied words for the baseline model but not for our model. This is consistent with the results shown in Figure 3 and 4. This higher value of p_{gen} informs us that during beam-search decoding the model is assigning a higher value to the attention distribution across all beams. So the behavior observed from the P_{DEC} vs P_{PTR} results is not just that coincidentally the attention value corresponding to the copied word was higher. It is indeed the pointer network weighing more for the word to be copied from source words. This difference in p_{gen} is not observed for the copy loss model 1-gram output.

	ROUGE-1	ROUGE-2	ROUGE-L
abstractive model [2]	35.46	13.30	32.65
pointer-generator + coverage	**38.94**	**17.08**	**35.90**
pointer-gen + cov. + copyloss	37.88	16.37	35.19

Table 1: ROUGE F_1 scores achieved by baseline model from See et al. [1] trained by us in PyTorch. We also include, for reference, scores from Nallapati et al. [2] which we obtained from their paper. All ROUGE scores have a 95% confidence interval of at most ± 0.24. The scores achieved by our new model are lower than that of the baseline model. We expected this since we are discouraging the copying behavior so it won't perform as well for those reference summaries that contain a significant amount of n-grams from the source, so it wont achieve high n-gram overlap which is what ROUGE metric quantifies.

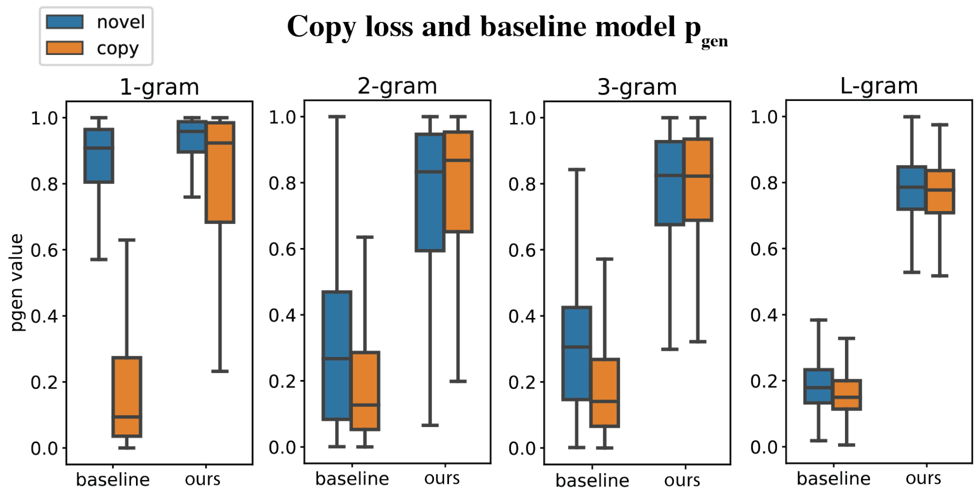

Figure 5: Boxplots show values of p_{gen} corresponding to novel and copied n-grams and sentence. We compare the values from baseline model and ours. Results for 1-gram clearly show that the baseline favors copying with a higher p_{gen} value.

6 Analysis

It is clear from our results that the pointer generator network of the baseline model have a significant effect in copying of words. It is confirmed through not only comparing results of P_{DEC} vs P_{PTR}, but also by looking at values of p_{gen} for novel and copied words shown from Figure 3-5.

The new model is able to correct for the copying behavior with its novel copyloss formulation. Although it performs a little worse in terms of ROUGE F_1 scores as seen in Table 1, it improves the amount of novel n-grams it generates significantly as seen in Figure 5.4.1. But from this plot we can see that there is still a lot of room for improvement as both baseline and new model produce far less novel n-grams compared to the reference summaries. We can see this also from the example shown in Table 2 that the copy loss model only makes small improvements in abstractiveness (replaced *believe* with *say*). This level of abstraction is still far from human capabilities of being able to extract meaning and synthesize new sentences.

We also observed that the copy loss model was more likely to produce more repeated n-grams although the coverage mechanism from the baseline model was used. This is most likely a problem caused by the mutual interference of coverage and copy loss signals.

7 Conclusion and Future Work

In this work we have shown that the pointer generator network introduced by See et al [1] will learn to copy words from the source article if trained using the loss expression proposed in their paper. In

Article (truncated): Police in the Indian city of Malegaon , in the western state of Maharashtra, are requiring identity cards for an unusual group of residents: Cattle. Following a recent state-wide ban on the sale and consumption of beef, authorities in the city have asked residents to take a 'mugshot' of their cattle and submit it to the police . Along with the photograph, the residents have to give information about their animal's ' unique features , ' such as the coloring and age of the cow , along with the length of its tail and other distinctive characteristics . Police officials believe this is the only way to solve cow slaughter cases and enforce the law . ...
Reference: authorities in the indian city of malegaon have asked residents to take a ' mugshot ' of their cattle . cows are revered by the majority hindu population , and many parts of the country have laws banning the slaughter of cattle . officials in malegaon believe this is the best way to solve cow slaughter cases and enforce the law .
baseline: residents have asked residents to take a ' mugshot ' of their cattle and submit it to the police . along with the photograph , the residents have to give information about their animal 's ' unique features , ' such as the coloring and age of the cow . police officials believe this is the only way to solve cow slaughter cases and enforce the law .
copy loss: police in the indian city of malegaon , in the western state of maharashtra , are requiring identity cards for an unusual group of residents : cattle . the residents have to give information about their animal 's ' unique features , ' such as the coloring and age of the cow , along with the length of its tail and other distinctive characteristics . police say this is the only way to solve cow slaughter cases and enforce the law .

Table 2: One example of decoded output using baseline and copy loss model. As can be seen from baseline output, most words are copied from the source article. Copy loss model output has some novel n-gram instances such as "police say this is" rather than copying "police officials believe this is". However, this result is still far from the level of abstraction that humans can produce.

order to help the model produce more abstractive summaries, we introduced a novel modifications to the training is able to significantly reduce the copy effect while not compromising too much on the summary quality in terms of ROUGE F_1 score.

Although we increased the number of n-grams our model is able to generate compared to the baseline, the output sentences are still very similar to the input articles sentences with maybe some words replaced with synonyms or simple shortening of a sentence as seen in Table 2. These models are still not able to, for example, take multiple sentences and summarize their meaning in one sentence. Another limitation of our model is that is it not easy to balance the loss values between coverage loss and copy loss. See et al. [1] also had this problem with coverage loss and teacher forcing loss term. The strategy used there was to train without coverage for 600k steps and then strain with coverage loss for 2k steps more. We did not find a strategy similar to this that worked well, so a potential future direction is look into a more systematic way to determine the optimal training schedule to balance losses.

Other interesting future directions would be to include other datasets such as XSum [14] where summaries contain less details and are more concise. The CNN/Daily Mail datasets provide multiple highlight sentences for each news article where some sentences are summaries and others are details meant to be eye catchy. If this trend is consistent across the current data set, it might cause the model to generate random details instead of information compression.

Another limitation of this work is that it is still not able to take very long sequence inputs (we crop at 400 tokens). An interesting future direction would be to explore the possibility of encoding tokens conditioned on their sentence location within a paragraph. This could allow us to leverage the fact that the more important sentences appear in certain parts of a paragraph and to pay more attention to those tokens than those in other sentences that may be details.

References

[1] Abigail See, Peter J Liu, and Christopher D Manning. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083, 2017.

[2] Ramesh Nallapati, Bowen Zhou, Caglar Gulcehre, Bing Xiang, et al. Abstractive text summarization using sequence-to-sequence rnns and beyond. *arXiv preprint arXiv:1602.06023*, 2016.

[3] Sumit Chopra, Michael Auli, and Alexander M Rush. 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, 2016.

[4] Yang Liu and Mirella Lapata. Text summarization with pretrained encoders. *arXiv preprint arXiv:1908.08345*, 2019.

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

[6] Shashi Narayan, Shay B Cohen, and Mirella Lapata. Ranking sentences for extractive summarization with reinforcement learning. *arXiv preprint arXiv:1802.08636*, 2018.

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

[8] Asli Celikyilmaz, Antoine Bosselut, Xiaodong He, and Yejin Choi. Deep communicating agents for abstractive summarization. *arXiv preprint arXiv:1803.10357*, 2018.

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

[10] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. In *Advances in neural information processing systems*, pages 2692–2700, 2015.

[11] Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. Teaching machines to read and comprehend. In *Advances in neural information processing systems*, pages 1693–1701, 2015.

[12] Python wrapper for rouge-1.5.5. `https://github.com/bheinzerling/pyrouge`. Accessed: 2020-02-20.

[13] Pytorch implementation of "get to the point: Summarization with pointer-generator networks". `https://github.com/atulkum/pointer_summarizer`. Accessed: 2020-02-20.

[14] Shashi Narayan, Shay B Cohen, and Mirella Lapata. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. *arXiv preprint arXiv:1808.08745*, 2018.