Summarizations and Dragons

Stanford CS224N Custom Project, Mentor: Prof. Chris Manning

David Ryan
Department of Computer Science
Stanford University
dpryan@stanford.edu

Abstract

Summarization models are often trained and tested on constrained datasets, which could limit their ability to handle less structured or out-of-domain data, like conversations. Unlike existing datasets, the Critical Role dataset (CRD-3) of roleplaying game transcripts is unstructured and conversational, but it still is centered around a shared game-playing goal and is of sufficient size to properly train and test these models [1]. In this paper, I examine the dataset and identify the characteristics that differentiate it from previous data. Then, I implement an expanded version of the pointer-generator summarization model, and evaluate its performance on this dataset in order to identify which model choices and architectures are well-suited to the dataset, as well as the limitations of this existing model.

1 Introduction

Broadly, neural text summarization condenses input text into a shorter summary. There are two main approaches to this: extractive methods, which tend to directly take quotes from the input text to create the summary, and abstractive ones, which paraphrase the text to generate novel summaries. Abstractive summarization is especially difficult, because the model doesn't just have to identify important information across a text, it also has to paraphrase that information and generate a novel summary. The longer the text and more varied the domain, the more difficult a given dataset is to summarize.

At the moment, one of the main datasets used to train these models is the CNN-Daily Mail (CNN-DM) dataset, which consists of lengthy articles, and relatively long summaries with broad subject matter. However, even this dataset has patterns that models can exploit: although the articles are longer, they tend to be front-loaded with information, and only contain a specific, journalistic type of speech [2].

To avoid these issues, I use The Critical Role dataset (CRD-3), a set of transcripts of Dungeon's and Dragons rolepaying game sessions streamed online which are paired with fan-made summaries [1]. This data is longer and messier, and contains naturalistic dialogue which is missing from the CNN-DM dataset. CRD-3 combines the scale of document datasets with the naturalistic language of conversational ones. In order to get a sufficient number of dialog-summary pairs, an issue with previous conversation datasets, they introduce a data augmentation method to match existing fan summaries to written transcripts. This provides a novel dataset which is noiser and more naturalistic. All of this means that the task is more difficult, and existing models optimized for datasets like CNN-DM may not be able to deal with this new dataset.

In this paper, I examine the specific characteristics of the CRD-3 dataset to see how it differs from the CNN-DM one, and then examine how an existing model - the pointer-generator network from See et. al. [3] - performs on different subsets of the data. From this, I identify a number of limitations of existing methods - particularly when dealing with the longer, less-structured dialogues - and some specific idiosyncracies of the dataset future researchers should be aware of when using it.

2 Related Work

Much of the related work is on the datasets that I describe above. Apart from that, the primary body of related work is on abstractive summarization models. Abstractive summarization focuses on providing summaries of input datasets by generating new sentences not contained in the source documen and/or paraphrasing the information in the source documents.

The model I chose to use is a pointer-generator model, which usually consists of encoder-decoder architectures leveraging attention. Initial models focused on using solely attention[4]. From there additional layers were introduced such as hierarchical networks [5], variational autoencoders [6], and other optimizations. The most immediately relevant work, of course, is from *Get To The Point: Summarization with Pointer-Generator Networks*, the paper which my implementation is based on [3]. Its two primary contributions are using pointer-generator networks to generate summary words from the source document, and coverage attention, which prevents the model from returning to portions of the text that have already been summarized.

Some models modify this basic architecture such as *Bottom-up Abstractive Summarization*, which adds an additional content-selector to prioritize important phrases in the source documents [7]. Others take alternate approaches, like, leveraging large, pre-trained models like T-5 [8] and BART[9].

I chose See et. al.'s model for a two reasons: (1), there are a number of clear, discrete additions to make to the model, which will let us investigate how they affect performance, and (2), the main aspects of the architecture - the pointer-generator and the coverage attention - should be well-suited to the CRD-3 dataset, since it the summaries tend to use a wide range of vocabulary, and there's multiple extended sequences of text that require evenly applied attention. I expand on this below where I introduce my approach.

3 Approach

Using the OpenNMT-py library, I created a series of models based on See et. al.'s pointer-generator networks approach [3], with some minor modifications to the basic architecture. The basic code for the first set of models was based on the OpenNMT-py documentation for using OpenNMT-py as a library, with the individual model components made up of various OpenNMT-py modules that act as wrappers for specific torch modules [10]. To train the different permutations of models, I used a combination of this code and config files based on the OpenNMT-py documentation [11]. My primary modification to See et. al.'s architecture was the types of encoder and decoder used. Whereas they used a single-layer bidirectional long short-term memory (LSTM) as the encoder, and a unidirectional LSTM as the decoder, I chose to use two-layer bidirectional LSTMs for both the encoder and the decoder. In my initial tests, I tried a baseline that replicated the architecture exactly, but found that this two-layer, bidirectional approach tended to be more performant.

3.1 Attention-only

My first variant on the model was the baseline used in See et al., an encoder-decoder with an added attention layer that is modeled after Bahdanau et. al. [12]. Training loss was the negative log likelihood of each word in the vocabulary, $loss = -logP(w_t)$.

This model uses attention to calculate context vectors when determining the next word to generate. The attention is calculated as:

$e_i^t = v^T \tanh\left(W_h h_i + W_s s_t + b_{\text{attn}}\right)$

$a^t = softmax\left(e^t\right)$

Where v, W_h, W_s and b_attn are learnable parameters, and the attention distribution dictates where the decoder looks for each word. This distribution is used to create a context vector (h_t^*), a weighted sum of the encoder's hidden states: $h_t^* = \sum_i a_i^t h_i$

Finally, this context vector is concatenated with the decoder state, fed through two linear layers, and used to create the final distribution to predict our words: $P_{vocab} = softmax\left(V'\left(V\left[s_t, h_t^*\right] + b\right) + b'\right)$.

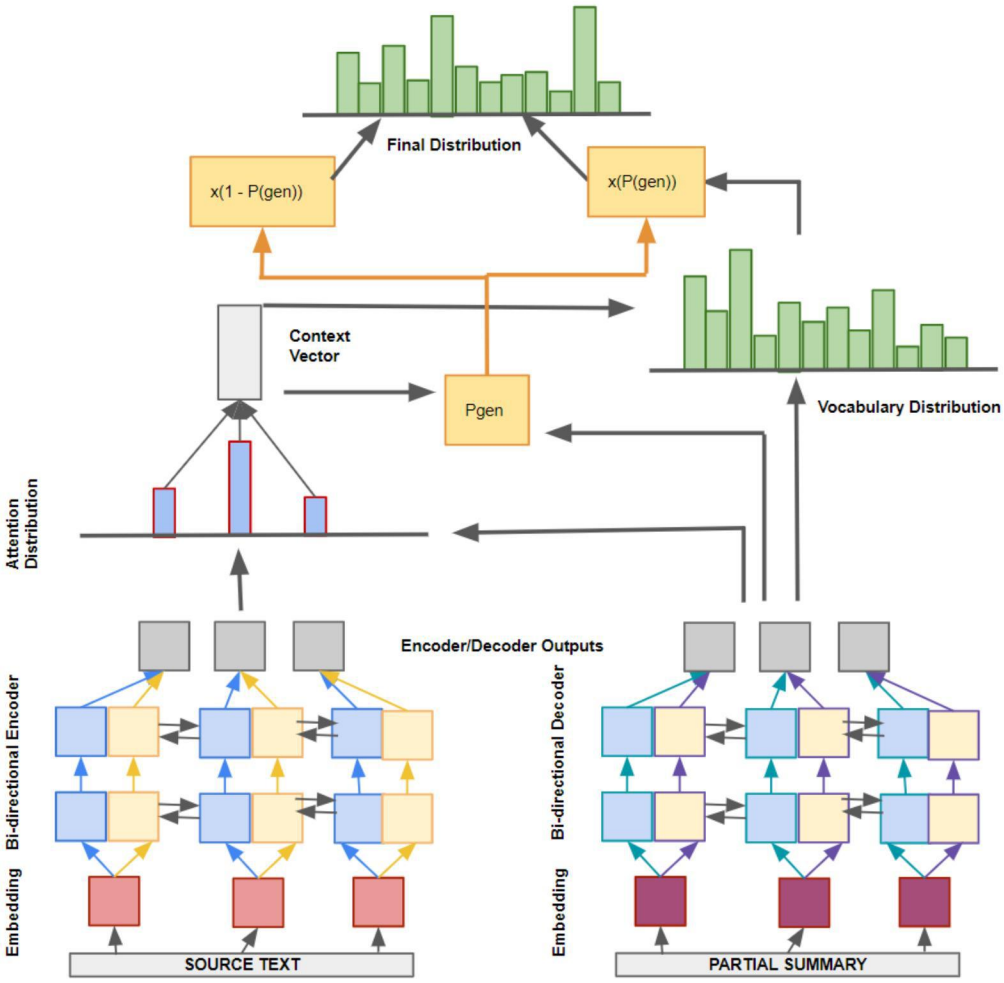

Figure 1: Modified Pointer-Generator Architecture, illustrates the probability of choosing a given word from the source text vs. the vocabulary.

3.2 Pointer-Generator

My next model incorporates a pointer-generator network, as described in See et. al.'s paper, based on Vinyals et al [13]. Figure 1 shows a diagram of this architecture. Basically, this lets the model predict words that are outside of the vocabulary. It calculates attention and the context vector the same way, but at each time step will add in a $generationprobability, p_gen \in [0, 1]$ for each timestep, with: $p_{gen} = \sigma\left(w_{h^*}^T h_t^* + w_s^T s_t + w_x^T x_t + b_{ptr}\right)$

Here, the learnable parameters are the w vectors and b_{ptr} This p_gen will be used to decide if the next word should be chosen from all the words in the source document, or the vocabulary, which is a smaller subset. This approach seemed particularly well-suited to the Critical Role dataset, since each set of dialogues and summaries tend to include fairly unique, and wide, vocabularies.

3.3 Coverage Attention

The final model incorporates the last addition to their full model, a coverage model which tries to account for the attention each word has gotten in previous decoder timesteps to avoid overly repetitive generated text.

This employs a coverage vector, c^t, calculated as:

$c^t = \sum_{t'=0}^{t-1} a^{t'}$. Basically, this measures how much attention each word has already received. This coverage vector is added to the attention calculation in the form of a parameter W_c, resulting in a new equation of:

$e_i^t = v^T \tanh\left(W_h h_i + W_s s_t + w_c c_i^t + b_{\text{attn}}\right)$

This introduces a sense of history to the decoder: each current decision it makes takes into account those in the past, which will, hopefully, allow it to avoid repeating the same words. This change also necessitates a change to the loss function, by adding a coverage loss that penalizes the decoder for repeatedly paying attention to the same words: $covloss_t = \sum_i \min\left(a_i^t, c_i^t\right)$.

Finally, this is incorporated into the overall loss function, modified by a hyperparameter λ:

$loss_t = -\log P\left(w_t^*\right) + \lambda \sum_i \min\left(a_i^t, c_i^t\right)$

This coverage attention seemed especially valuable for the Critical Role dataset, since so many of the dialogue chunks are very long, and (as I validate later on) important information is spread throughout the chunk. This technique prevents the model from fixating on one specific portion of the text, and repetitively drawing from it.

All told, this should mean that the summary will consider the context around each word (attention, from the baseline model), be able to use words outside of the vocabulary (the pointer-generator network), and avoid overly repetitive summaries (the addition of coverage).

4 Dataset Analysis

Before turning to the models, it's important to understand the characteristics of the dataset. My goal is to produce a summary for a set of turns of dialogue – every change in speaker begins a new turn. The full CRD-3 dataset is composed of 159 dialogue transcripts, separated into 398,682 turns. Groups of turns are paired with sentences from fan summaries describing each episode.

To construct the pairs, the authors calculated 'alignment scores' between chunks of dialogue (sets of turns) and summary sentences. This score was a scaled ROUGE score: $B = \frac{|2*t(s)\cap t(a)|^2}{|t(s)|+|t(a)|}$, where t is a tokenization function returning the number of tokens. The scaling term, $t|(s) \cap t(a)|$, gives additional importance to the absolute number of tokens that overlapped between turns and the summary. Essentially, this meant that sets of turns were paired with the summaries they shared the most tokens with. To find the pairings, they used the Needleman-Wunsch algorithm, which imposes strict order constraints on the summaries and turns: a_i and $a_i + 1$ had to be contiguous sets of turns, and the summaries had to be assigned to them in order. It then then finds the set of turn(s)/summary pairs that maximizes the alignment score for each pair, subject to these constraints. However, there was no restriction on different turns being associated with multiple summaries: for example, turns 23 - 24 could be associated with summary 7, and turns 24 - 33 could be associated with summary 8, meaning that turn 24 was associated with both summaries. Although the full dataset generated summaries of 2, 3 and 4 sentences in length, in order to keep the data size manageable I only examine 2-sentence summaries.

This unique method of pairing summaries to dialogue, as well as the (TODO what characteristics)specific characteristics of the dialogue transcripts themselves, mean that the CRD-3 and CNN-DM datasets differ in significant ways. To investigate these differences, I examine descriptive statistics for the dataset as a whole, as well as 60 dialogues I coded by hand[1].

4.1 Overlapping Pairings and Pairing Accuracy

In the CNN-DM dataset, each article is paired only with its headline. That isn't the case in the CRD-3 dataset. Overall, 32% of the dialogue chunks (n=5,903) consisted of only a single turn of dialogue, many of which had multiple summary sentences referring to them. In the hand-coded subset, 10 summaries all described a single turn of dialogue, a 2,476 token turn which summed up the events of the previous game. This overlap isn't unique to these single-turn chunks: among the hand-coded summaries, a substantial number of dialogues contained information relevant to either the next or the previous summary, as described in Table 1. In order to match this structure in the data, the model

[1]See appendix for additional information

would have to learn to pair several different summaries with the same chunk of dialogue, which seems unlikely, especially in the case of the aforementioned single-turn chunks. To see how this may affect performance, I will create a subset with only these single-turn chunks for my experiments.

In the CNN-DM dataset, every article and headline is automatically correctly associated with each other. This isn't the case in CRD-3. The association between the summaries and dialogues is measured by the alignment score - a higher score indicates that the pairing is more likely to be correct. In the hand-coded subsample, I found that 19.4% of the dialogues were missing at least some of the information contained in the summaries, and this tended to happen with chunks with lower alignment scores. This makes it difficult for the model to accurately generate a summary, and adds a certain amount of relatively unavoidable error into the models predictions: even if it perfectly summarized the dialogue, it won't match the reference summary.

Table 1: Chunks in the hand-coded subset that contain dialogue for contiguous summaries as well.

Any Overlap	Next Summary	Previous Summary
55%	45%	38%

4.2 Text Length and the Inverted Pyramid

Before continuing, there are two final aspects of the dataset I want to examine: the lengths of the documents, and where the important information tends to be. Many existing models trained on the CNN-DM dataset truncate their inputs and outputs to decrease the volume of training data. A standard truncation is 400 tokens for the source and 100 tokens for the output [3][2]. The CNN-DM dataset has an average article length of 781 tokens, and summary length of 193 tokens, while the two-sentence summary subset of the CRD-3 dataset has an average of 3,015 tokens per dialogue and 193 per summary, and the 3 and 4 sentence subsets are 50% and 100% longer still. This means that a similar truncation in CRD-3 would remove a greater proportion of the text than in CNN-DM, which may remove key information.

Less information may be lost from truncation of CNN-DM data because articles tend to follow an 'inverted pyramid' structure, where most of the important information is in the beginning of the text [2]. To quantify this tendency, previous researchers have annotators mark the most important sentences in articles (n=100), and found they generally occurred at the beginning of the article [2]. However, the inverted pyramid model doesn't seem to hold as well for the CRD-3 dataset. I carried out a similar procedure for a subset of summary sentences (n=60), and marked at which turns in their associated dialogue chunks their information came from, operating under the assumption that the turns included in the summary are likely the most important. Figures 2 and 3 show histograms of which dialogue chunks summary sentence information came from, with and without the single turn chunks. The x-axis is the percentage into the document in which the sentence was found, measured as the turn it was found in divided by total turns (e.g. a sentence used in the summary located in turn 2 of 4 would be at the 50-percent mark).

This measurement approach doesn't work for single-turn chunks, since all information is found in the first turn, which can often be very long. For the other chunks there's a fairly even distribution across the first three quarters of the dialogue, indicating that the inverted-pyramid model does not hold here. All of this suggests that truncation will be less effective on this dataset, since for single-chunk turns with multiple summaries, the same chunk of dialogue is captured each time, and for multiple turn chunks important information is lost. Comparing the subset without single turns to the full data may illustrate how much this matters – chunks here tend to have a higher token count on average - 3,823 – so truncation may be especially damaging here.

5 Experiments

5.1 Data Used

To reduce training time I used only the subset of the data containing turns paired with 2-sentence summary chunks, with any pairs that contained empty summaries excluded (1/3 of the total data). This resulted in 13,861 training, 2,485 test, and 2,094 validation pairs. I also trained models on two

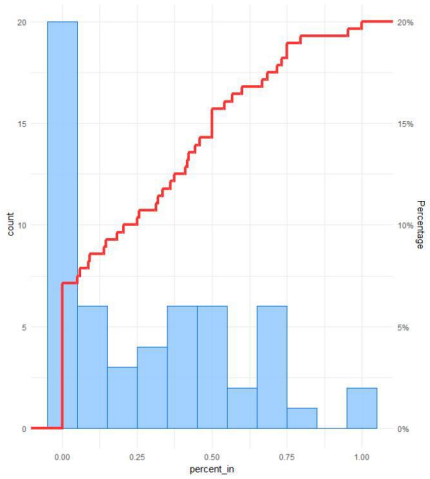

Figure 2: All Chunks

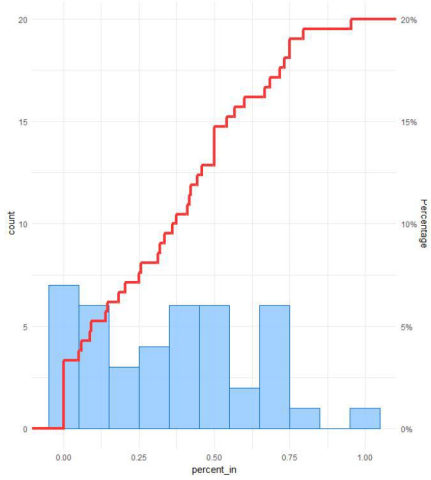

Figure 3: Single-turn Chunks Excluded

secondary subsets of the data: (1) only the chunks with alignment scores at or above the median for the dataset (training pairs = 6,931, test = 1,243, validation = 1,047) and (2) only multi-turn dialogue chunks (training pairs = 9,600, test = 1,603, validation = 1,334).

5.2 Evaluation method

The main metrics were ROUGE-1, ROUGE-2, and ROUGE-L scores for each dataset. Each of these metrics measures the overlap in tokens between the reference summaries and the model predictions, ROUGE-1 at a unigram level, ROUGE-2 at a bigram level, and ROUGE-L using the longest common sub-sequence between the two. I measured the proportion of repeated summaries predicted by the model as the number of contiguous, repeated summaries in the predictions divided by the total number of summaries. This is intended to measure the impact of multiple summaries being based on the same dialogue, and to see if this encourages the model to (wrongly) repeat summaries when it's presented with these overlapping chunks.

5.3 Experimental details

For every experiment, I used 512-dimensional (rather than See et al.'s 256-) hidden states, and used 200-dimensional pre-trained word embeddings from GloVe. I used an Adagrad optimizer with an initial accumulation of 1.0 and max gradient norm of 2, and a learning rate of 0.15, as per See et. al. [3]. For the baseline and attention-only model I used a source vocabulary of 90k, and a target vocabulary of 22k. For all the other models I used a source and target vocabulary of 50k. Dialogue lengths were truncated to 800, and the summary lengths to 200. Due to memory limitations of cards being used to train, the baseline and attention-only models both had batch sizes of 12, and the other models had batch sizes of 8. The baseline model trained for 12 hours and 20,000 iterations, the attention-only model trained for 12 hours and 20,000 iterations, the full pointer-generator models trained for 40,000 iterations and 15 hours, and the single-turns excluded model ran for 10 hours and 30,000 iterations. To add coverage, I trained the pointer-generator model an additional 5,000 iterations and 2 hours.

Model	ROUGE-1	ROUGE-2	ROUGE-L	Repetition
Baseline	11.9	.06	.05	64%
+Attention	14	.09	11.1	42%
+Pointer-Generator	17.4	2.7	14.4	30%
+Coverage	17	2.8	14.2	29%
Single-turn	18.4	2.4	14.8	21%
Chen and Bansal 2018	27.38	9.2	25.2	N/A

5.4 Results

My results are found in the table above. As described earlier, I calculated the ROUGE-1, ROUGE-2 and ROUGE-L F1 scores for my models, calculated with the pyrouge package[2] , as well as the performance of the baseline model from Rameshkumar and Bailey [1].

The F1 scores did improve consistently with each addition, with the exception of coverage. It's possible there was an implementation issue, or it just wasn't as effective for this particular data. They follow the same general trends as the baseline approach: R-1 is highest, followed by R-L, and then, distantly, R-2. On face this makes sense; unigram overlap is easier to achieve than bigram overlap, and when your longest shared subsequences are largely unigrams it is to be expected that the R-1 and R-L scores would be similar. Considering the fact that I used one third of the training set, it isn't particularly surprising that my ROUGE scores are lower across the board. I think the results for the model trained without the single turns are most compelling; as expected the number of repeated summaries dropped, and the ROUGE scores are either close to or exceed the full models score despite being trained on a dataset that's half the size. The subset of pairs with the highest alignment scores ended up with ROUGE scores close to 100, suggesting an issue with the data splits.

6 Analysis

One of the most telling indicators of the difference in model performance was the percentage of repeated summaries each model generated. These were almost entirely failed predictions, generally occuring when there were multiple summaries generated from overlapping dialogue. The fact that this percentage fell - from 64% for our baseline, to 42 % in our model with attention and finally around 30% for our final models is a good indicator that the pointer-generator network in particular is doing a good job of varying these summaries based on the source text. For example,

The most common issue with baseline model was it simply extracting most of the first portion of a dialogue. For example, the baseline model's summary for one dialogue was just "jester and caleb find the house fenced with pointed-top iron bars, locked gates, crownsguards, and a mastiff on a chain. there are some tall trees near it, however.", which is exactly the beginning of the chunk of dialogue. The reference summary, on the other hand, ignores this initial turn entirely, and focuses on later portions. The model with added attention captures the correct event, with "the party take a heroes' feast before they leave to sleep in the woods while *vax* puts on one of the party with his holy arrows and on the *vax*. *vax* suggests that they have a feast before they row up the rift." This matches the general flow of events in the reference summary, but (emphasis mine) replaces the out-of-vocabulary proper nouns (the second two repetitions) with the name that was in the vocabulary (vax). The pointer-generator model tended focus on the correct turns more often, like the model with attention, but also tended to get proper nouns, or vocabulary that tended only appeared in one section of the dialogue.

One general issue with analyzing their comparison from summary to summary is that the performance of the models, as a whole, just wasn't that great - the unigram overlap was low, and the bigram overlap was lower. Since the chunks of dialogue were so long, the different models could end up giving summaries so different that they weren't even comparable. For example, on one dialogue, which consisted of a general overview of the show's logistics, the baseline model captured a portion of the QA: "q: how much do you write and how much do you improv? everything is improv.", the model with attention extracted a tangent about how well a t-shirt was selling, "a critical role t-shirt design of grog's cask that was created by the audience", the pointer-generator model captured the recap of events, "" carr had encountered yasha who joined back up with the party. you had encountered yasha who joined yasha who joined with the party." and the model with coverage mentioned a giveaway, "the rest of the twitch giveaway is <unk>". I think this speaks to the specific difficulties in working with a much more difficult dataset, with a model that isn't as well-suited to it. Because the predictions as a whole are less accurate, it's harder to qualitatively examine specific summaries, and compare model versions in that way.

Instead I will turn to other questions, namely to what extent is the model abstractive on this dataset, and how effective was taking a subset excluding single-turn dialogue chunks?

[2]https://pypi.org/project/pyrouge/

6.1 Abstractive or Extractive?

I can investigate to what extent the model is extractive versus abstractive, compared to the reference summaries, by looking at the percentage of unigrams in each model's generated summaries that are novel, shown in Table 2. We can actually see that the percentage of novel unigrams is higher in our models without the pointer-generator network than those with the network. This makes sense: the models not using the pointer generator appeared to generate incorrect unigrams frequently, and a summary not related to the source text is likely to have less overlap with that source text. In contrast to this, both models with pointer-generator networks were actually more extractive on this measure, but this may be because they were more accurate.

Table 2: Comparing the percentage of novel unigrams (don't appear in source text) by model, with reference summaries.

Baseline	Attention	Pointer-Generator	Coverage	Ref.
64%	65%	42%	42%	32%

My models generated a higher percentage of novel unigrams than See et. al.'s models did in the CNN-DM dataset. However, this may be because there was a very different prevalence of novel unigrams in the reference summaries– novel unigrams were around 10% of the CNN-DM dataset but more than 40% of the CRD-3 [3]. This difference is also reflected in the percentages of novel bi- and trigrams. In every model - as well as the reference summaries - the percentage of novel bi- and trigrams was between 98% and 99%. This is markedly different from the equivalent statistics See et. al. found for the CNN-DM dataset, where the percentage of novel bi- and trigrams was well below 10% for the pointer generator model, and between 50% and 70% for the reference summaries. One of the conclusions they drew from their equivalent examination was that the pointer-generator network tended to write summaries extractively. The fact that this dataset contains so many reference summaries with novel ngrams presents a very different context to evaluate the model in, where the reference summaries are much more abstractive. In this case, it seems that the models matched that level of abstraction much more closely than in the CNN-DM dataset, suggesting that some of the tendency towards extractive summarization the model exhibited could be, in part, a function of the dataset it was applied to, not the model itself.

6.2 Subset Without Single Turns

I think one particularly interesting result was that the dataset without any single turn chunks performed substantially better than the full dataset, despite having a significantly reduced dataset. This suggests that the issue I predicted earlier: the model being given the same dialogue chunks, and therefore returning the same summary each time, was happening. This is corroborated by the significantly lower incidence of repeated summaries: 21%, down from 29% in the coverage model. This is particularly remarkable when we recall that this subset had longer chunks of dialogue than the dataset as a whole, so an even larger percentage of the tokens were being truncated. Future research on this dataset should either make this the defacto filter for the data, or find another technique to deal with these single-turn chunks. One option would be arbitrarily splitting these chunks based on the number of summaries that share them: i.e., if there are 4 associated summaries, splitting the turn into fourths and assigning the fourths consecutively. This has its own issues, of course, but finding some automatic way to adapt this data would certainly help.

7 Conclusion

I successfully implemented the pointer-generator model, validated that it improved with each addition, and specifically identified some of the unique characteristics and challenges that the CRD-3 dataset comes with. As the first attempt to train a summarization model on this dataset apart from the paper introducting it, I think this initial exploration is very valuable.There are a number of limitations to this work: my performance on the ROUGE metrics were below those achieved by the original CRD-3 paper, I only trained on a subset of the data, and I wasn't able to fully explore the effects of the dataset's unique structure and which model architectures best suited to dealing with them. Hopefully future work can build on this, more fully explore the dataset, and develop text summarization models to meet this new challenge.

References

[1] Revanth Rameshkumar and Peter Bailey. Storytelling with dialogue: A critical role dataset. In *Association for Computational Linguistics (ACL)*, 2020.

[2] Nitish Shirish Keskar McCann Bryan Xiong Caiming Kryscinski, Wojciech and Richard Socher. Neural text summarization: A critical evaluation. *CoRR*, abs/1908.08960, 2019.

[3] 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)*, 2017.

[4] Alexander M. Rush, Sumit Chopra, and Jason Weston. A neural attention model for abstractive sentence summarization. *CoRR*, abs/1509.00685, 2015.

[5] Ramesh Nallapati, Bing Xiang, and Bowen Zhou. Sequence-to-sequence rnns for text summarization. *CoRR*, abs/1602.06023, 2016.

[6] Yishu Miao and Phil Blunsom. Language as a latent variable: Discrete generative models for sentence compression. *CoRR*, abs/1609.07317, 2016.

[7] Deng Yuntian Gehrmann, Sebastian and Alexander Rush. Bottom-up abstractive summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018.

[8] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *CoRR*, abs/1910.10683, 2019.

[9] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *CoRR*, abs/1910.13461, 2019.

[10] OpenNMT py Documentation: Library. https://opennmt.net/opennmt-py/examples/library.html.

[11] OpenNMT py Documentation: Summarization. https://opennmt.net/opennmt-py/examples/summarization.html.

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

[13] Fortunato Meire Vinyals, Oriol and Navdeep Jaitly. Pointer networks. In *Neural Information Processing Systems*, 2015.

8 Acknowledgment

Thank you to Chris Manning for his mentorship and guidance throughout the project.