Dan Waters · danwaters@my.unt.edu
University of North Texas
CSCE 5290 · Natural Language Processing
Fall 2021

Abstractive Text Summarization

CSCE 5290 Natural Language Processing | Final Project Report

GitHub: https://github.com/danwaters/nlp-abstractive-text-summarization
Increment 1 video link: [link]
Increment 2 / final video link: [link]

Detailed project reports from each increment can be found in the **Appendix** of this document. The main section is a high-level report, and may contain some useful duplications from the increment documents.

Introduction

Team

This is an individual project researched and implemented by Dan Waters (danwaters@my.unt.edu) and is discussed from my sole perspective.

Motivation and Significance

As the volume of data increases, humans have an increasing need to make decisions faster, with less time available to mentally absorb large swaths of text. Executives are notorious for preferring more concise information over large reports, simply due to time constraints and the need to make quick, well-informed decisions. Is today's technology capable of generating short summaries that an executive could base a decision on? That is the question I am to explore in this paper.

Significant problems still remain in summarization. Some approaches may produce factually incorrect information in sentences which are otherwise grammatically correct. Some of these difficult problems make summarization techniques, in the current state of the art, a "nice to have" and not yet something which can be taken as factual all the time. Accordingly, I will also provide a list of use cases for which abstractive summarization may fulfill a business need, while identifying gaps that prevent it from encompassing other business critical use cases.

Dan Waters · danwaters@my.unt.edu
University of North Texas
CSCE 5290 · Natural Language Processing
Fall 2021

Objective

This project's primary objective is to implement and empirically evaluate a few different approaches to abstractive text summarization, identify the strengths and weaknesses of each, and determine what can be improved.

Features

At a system level, the main feature can be simply described: the system generates a summary of some input text using abstractive summarization techniques. The system will be evaluated using ROUGE metrics[1] where possible.

Datasets and Exploratory Data Analysis

The original dataset under consideration is the CNN/Dailymail dataset (also called the DeepMind Q&A dataset[2]). I quickly learned in my research that this dataset is absolutely massive. Because training a basic Seq2Seq model was taking on the order of days - which made it very hard to iterate - I eventually preprocessed and trimmed down this dataset to only 50,000 examples (from over 280,000) and saved it into a csv file for further experimentation.

While researching Seq2Seq summarization approaches with Tensorflow, I discovered a second, lighter-weight dataset: the News Summary dataset[3] from Kaggle. Much of the project involves trying to apply the methods from this News Summary dataset to the CNN dataset to see what kind of results can be achieved.

Exploratory Data Analysis
Comparing the size and scope of these two datasets, it is not hard to imagine the vast difference in hardware and time requirements to train models based on each. See Increment 2 in the Appendix section for more detailed analysis including word count distributions, histograms, and box plots for both the text and summary.

Statistic	**CNN/Dailymail**	**News Summary (Kaggle)**
Total number of examples	287,113	102,915

[1] "ROUGE (metric)." *Wikipedia*, Wikimedia Foundation, 17 September 2021, https://en.wikipedia.org/wiki/ROUGE_(metric)
[2] Cho, Kyunghun. "DeepMind Q&A Dataset." Accessed 20 Sep 2021, https://cs.nyu.edu/~kcho/DMQA/
[3] Vonteru, Kondolarau. "NEWS SUMMARY / Generating short length descriptions of news articles." *Kaggle*. Accessed 11/01/2021, https://www.kaggle.com/sunnysai12345/news-summary

Dan Waters · danwaters@my.unt.edu
University of North Texas
CSCE 5290 · Natural Language Processing
Fall 2021

Min / max article length	8 / 2,347	1
Min / max summary length	4	11,999
Mean article length	691	71
Mean summary length	51	11

Related Work and Background

The CNN/Dailymail dataset is frequently referenced in the world of abstractive text summarization research. There are several models which have been trained extensively and benchmarked on this dataset; these are efforts which are full-time research projects likely spanning more than a semester of part-time work. These results have been aggregated by Papers With Code[4] :

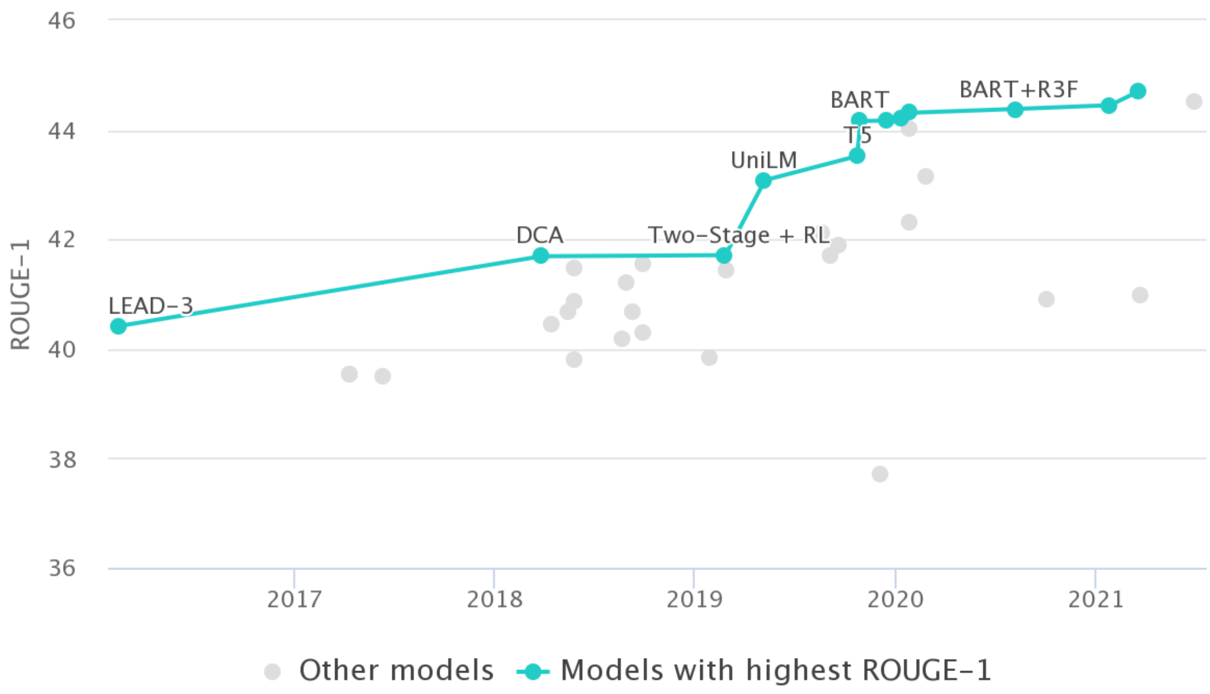


Figure 1. *Abstractive Text Summarization on CNN / Daily Mail leaderboard. Source: Papers with Code*

According to this leaderboard, the top models and their ROUGE-1 scores are:

Model	**ROUGE-1 Score**	**Year**

[4] "Abstractive Text Summarization on CNN / Daily Mail." *Papers with Code*, Facebook AI. Accessed 25 October 2021, https://paperswithcode.com/sota/abstractive-text-summarization-on-cnn-daily

Dan Waters · danwaters@my.unt.edu
University of North Texas
CSCE 5290 · Natural Language Processing
Fall 2021

GMM-XXLarge	44.7	2021
BART + R-Drop	44.51	2021
MUPPET BART Large	44.45	2021

The benchmark score is LEAD-3 with a score of 40.42, based on sequence-to-sequence recurrent neural networks.

For this project, after investigating a few different methods, I chose to compare the following approaches:

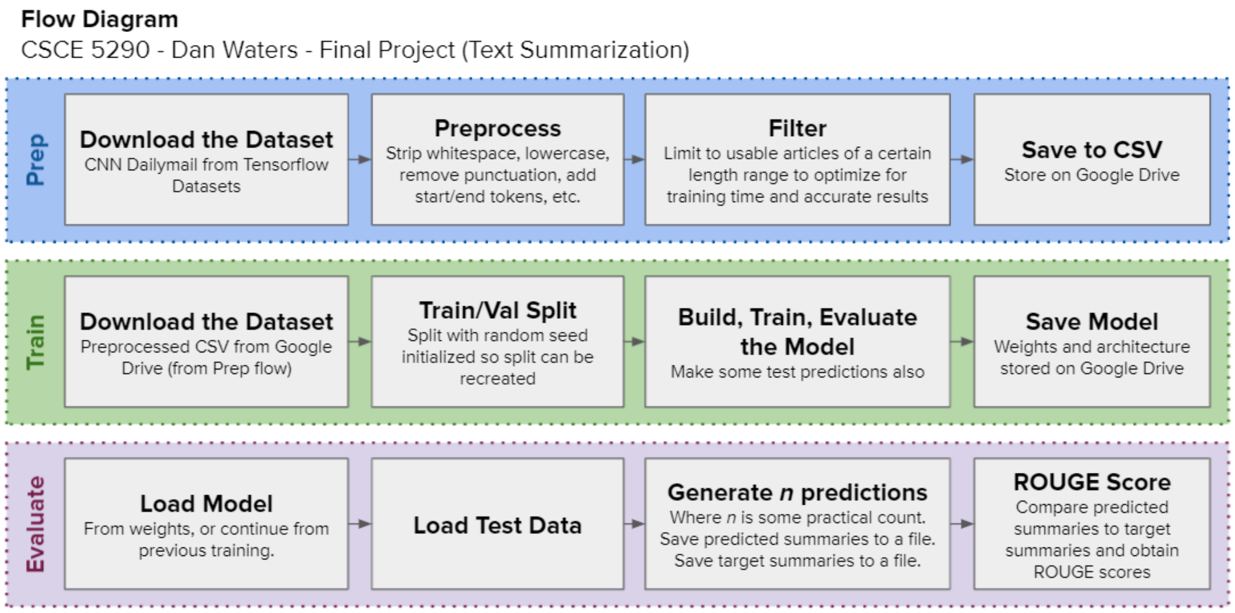
1. Seq2Seq Encoder/Decoder (LSTM) model that I built and trained myself (with much help from an article, linked below)
2. Pre-trained HuggingFace transformer: BART flavor
3. Pre-trained HuggingFace transformer: Google T5 flavor

For the Seq2Seq LSTM model, I trained and evaluated it using my own preprocessed subset of the CNN/Dailymail data set.

Workflow

The following diagram depicts the flow of data and tasks. For the pre-trained models, the ‘train’ swimlane is eliminated and we go straight to the predictions, as these models are already tuned on CNN/Dailymail.

Dan Waters · danwaters@my.unt.edu
University of North Texas
CSCE 5290 · Natural Language Processing
Fall 2021


Flow Diagram
CSCE 5290 - Dan Waters - Final Project (Text Summarization)
Prep
Download the Dataset
CNN Dailymail from Tensorflow Datasets
Preprocess
Strip whitespace, lowercase, remove punctuation, add start/end tokens, etc.
Filter
Limit to usable articles of a certain length range to optimize for training time and accurate results
Save to CSV
Store on Google Drive
Train
Download the Dataset
Preprocessed CSV from Google Drive (from Prep flow)
Train/Val Split
Split with random seed initialized so split can be recreated
Build, Train, Evaluate the Model
Make some test predictions also
Save Model
Weights and architecture stored on Google Drive
Evaluate
Load Model
From weights, or continue from previous training.
Load Test Data
Generate n predictions
Where n is some practical count. Save predicted summaries to a file. Save target summaries to a file.
ROUGE Score
Compare predicted summaries to target summaries and obtain ROUGE scores

Solution 1: Seq2Seq Encoder/Decoder with LSTM

Most of the high-performing models listed above are state-of-the-art and not practical for a student to train with limited resources, so the first approach I took was simply to follow this tutorial on the Seq2Seq method. Thankfully, this is a recent article that ran out-of-the-box. However, it uses the News Summary dataset, and I wanted to try it against the CNN/Dailymail dataset.

Sidebar: Implementation Struggles

As described above in the Exploratory Data Analysis section, the CNN dataset needed to be pre-cleaned and trimmed down in order to stand any chance at all of being trained in a reasonable timeframe. Multiple roadblocks were encountered which led to this, for example:

- Training times were in excess of 16 hours. This required a lot of staying awake, prodding the mouse, so that the Colab session would not disconnect. A Colab Pro+ subscription eventually helped with this, but…
- The runtimes were recycled after the training passes ended, and before being able to save the model weights and architecture. Eventually the model weights did get saved and rehydrated into a new workbook. But…
- Runtime recycling also caused new randomized, shuffled train/validation splits to happen, which caused sequence decoding to use the wrong vocabulary - so the

model weights were trained with a different vocabulary and embedding strategy than how the validation data was processed.

A **key learning** from this very painful experience is to just get the training data processed and stored, along with trained models, one time only. Take those artifacts to a new notebook and start from there.

Sequence-to-Sequence with Encoder/Decoder Model

The main purpose of this approach is to handle dynamic-length strings of text (within some reasonable limitations) which are tokenized into dense vectors of variable length. In actuality, the sequences are padded at the end up to some maximum length, but it relaxes the requirement of some basic encoder/decoder mechanisms that the input and output vector sizes match exactly.

Architecture

Encoders encode the inputs into a new vector representation. That new representation is sent to the decoder to have its input reconstructed.

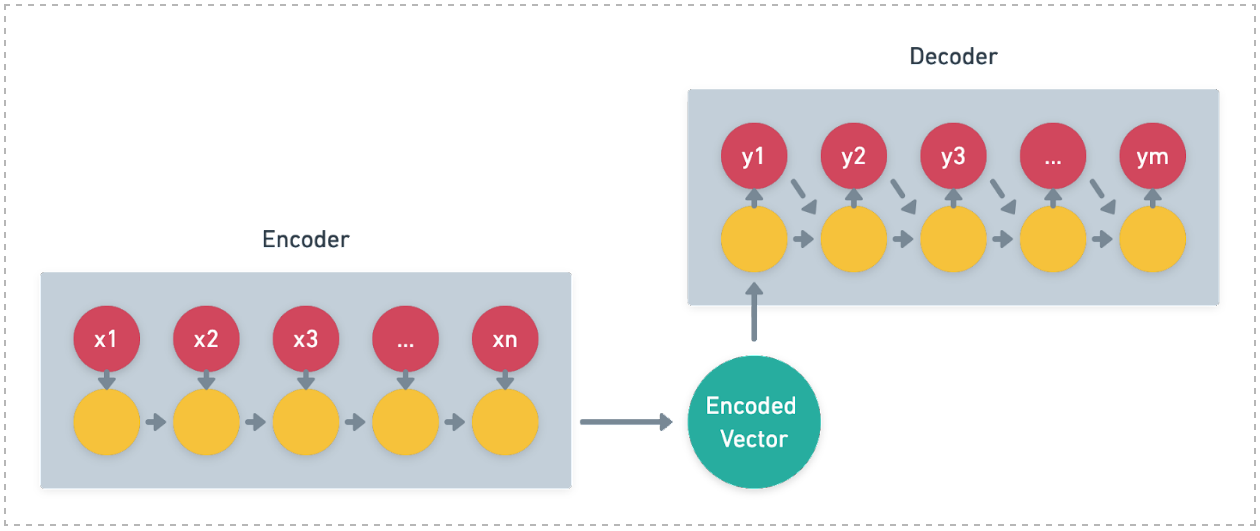
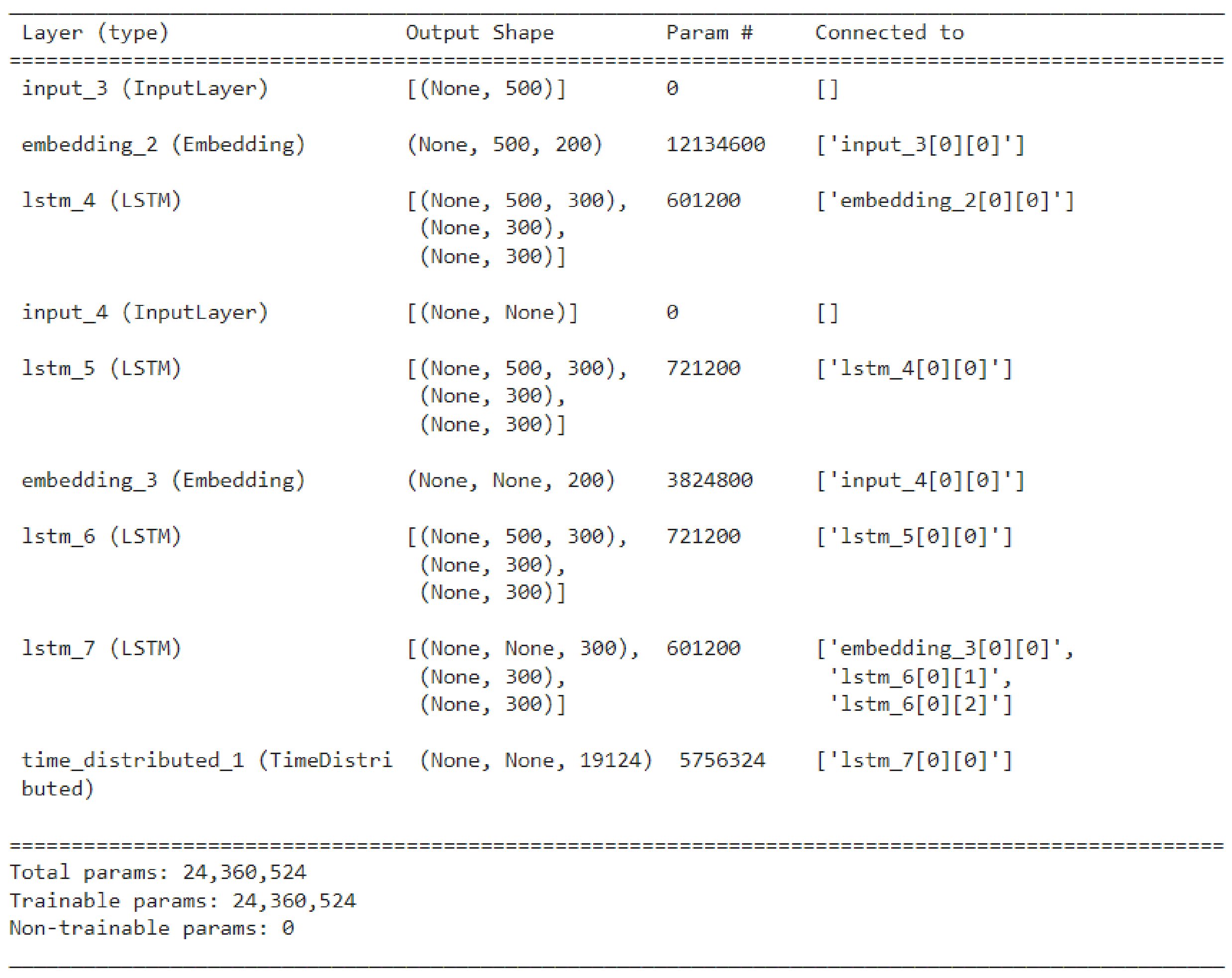


***Figure 2**. Basic Encoder/Decoder architecture. Source:* [*https://blog.paperspace.com/introduction-to-seq2seq-models/*](https://blog.paperspace.com/introduction-to-seq2seq-models/)

This model uses LSTMs for the RNN layers in both the encoder and decoder components. While I did experiment with other hyperparameter values (for example, embedding dimensions, latent states, and so on), the final version here uses the same hyperparameters from the tutorial. There was simply no time to perform a grid search or find the optimal set of hyperparameter values. I did, however, manipulate the batch size (150) and epoch count (15) to get to a reasonable training time in the end. Batch sizes over 150 would cause the training to fail with an out of memory error.

The architecture for the Seq2Seq model is as follows (Figure 3):

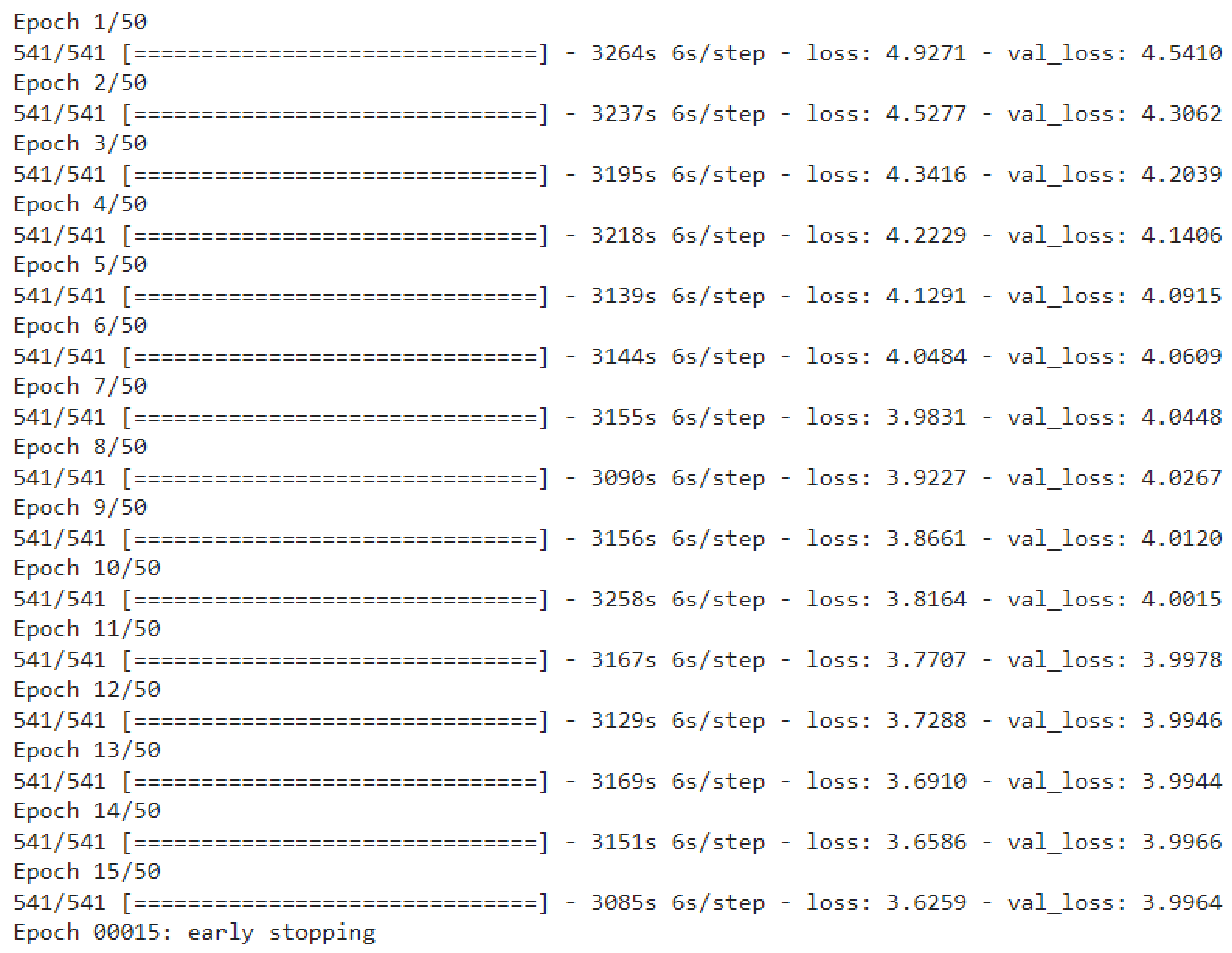


```
__________________________________________________________________________________________________
 Layer (type)                   Output Shape         Param #     Connected to
==================================================================================================
 input_3 (InputLayer)           [(None, 500)]        0           []

 embedding_2 (Embedding)        (None, 500, 200)     12134600    ['input_3[0][0]']

 lstm_4 (LSTM)                  [(None, 500, 300),   601200      ['embedding_2[0][0]']
                                 (None, 300),
                                 (None, 300)]

 input_4 (InputLayer)           [(None, None)]       0           []

 lstm_5 (LSTM)                  [(None, 500, 300),   721200      ['lstm_4[0][0]']
                                 (None, 300),
                                 (None, 300)]

 embedding_3 (Embedding)        (None, None, 200)    3824800     ['input_4[0][0]']

 lstm_6 (LSTM)                  [(None, 500, 300),   721200      ['lstm_5[0][0]']
                                 (None, 300),
                                 (None, 300)]

 lstm_7 (LSTM)                  [(None, None, 300),  601200      ['embedding_3[0][0]',
                                 (None, 300),                     'lstm_6[0][1]',
                                 (None, 300)]                     'lstm_6[0][2]']

 time_distributed_1 (TimeDistri  (None, None, 19124)  5756324    ['lstm_7[0][0]']
 buted)

==================================================================================================
Total params: 24,360,524
Trainable params: 24,360,524
Non-trainable params: 0
__________________________________________________________________________________________________
```

***Figure 3**. Seq2Seq model architecture, based on this article but trained on the preprocessed CNN dataset.*

Before drastically trimming down the dataset, training times were upwards of **3 hours per epoch.** This made it extremely difficult to iterate on the project, especially with hardware and session limitations imposed by Google Colab. It was only possible to run two experiments concurrently. With some modifications, I was able to get the training time down to about 54 minutes per epoch. Early stopping certainly helped alleviate some of the wait time, but 15 hours is quite a long time to wait, especially for the results I am about to present.

Dan Waters · danwaters@my.unt.edu
University of North Texas
CSCE 5290 · Natural Language Processing
Fall 2021



```
Epoch 1/50
541/541 [==============================] - 3264s 6s/step - loss: 4.9271 - val_loss: 4.5410
Epoch 2/50
541/541 [==============================] - 3237s 6s/step - loss: 4.5277 - val_loss: 4.3062
Epoch 3/50
541/541 [==============================] - 3195s 6s/step - loss: 4.3416 - val_loss: 4.2039
Epoch 4/50
541/541 [==============================] - 3218s 6s/step - loss: 4.2229 - val_loss: 4.1406
Epoch 5/50
541/541 [==============================] - 3139s 6s/step - loss: 4.1291 - val_loss: 4.0915
Epoch 6/50
541/541 [==============================] - 3144s 6s/step - loss: 4.0484 - val_loss: 4.0609
Epoch 7/50
541/541 [==============================] - 3155s 6s/step - loss: 3.9831 - val_loss: 4.0448
Epoch 8/50
541/541 [==============================] - 3090s 6s/step - loss: 3.9227 - val_loss: 4.0267
Epoch 9/50
541/541 [==============================] - 3156s 6s/step - loss: 3.8661 - val_loss: 4.0120
Epoch 10/50
541/541 [==============================] - 3258s 6s/step - loss: 3.8164 - val_loss: 4.0015
Epoch 11/50
541/541 [==============================] - 3167s 6s/step - loss: 3.7707 - val_loss: 3.9978
Epoch 12/50
541/541 [==============================] - 3129s 6s/step - loss: 3.7288 - val_loss: 3.9946
Epoch 13/50
541/541 [==============================] - 3169s 6s/step - loss: 3.6910 - val_loss: 3.9944
Epoch 14/50
541/541 [==============================] - 3151s 6s/step - loss: 3.6586 - val_loss: 3.9966
Epoch 15/50
541/541 [==============================] - 3085s 6s/step - loss: 3.6259 - val_loss: 3.9964
Epoch 00015: early stopping
```

Figure 4*. Training times and validation loss per epoch.*

Even with early stopping, it could have been beneficial to continue for some time, based on the loss plots. Early stopping was set to monitor validation (test) loss. The patience was set to 2 because I began to run out of patience.

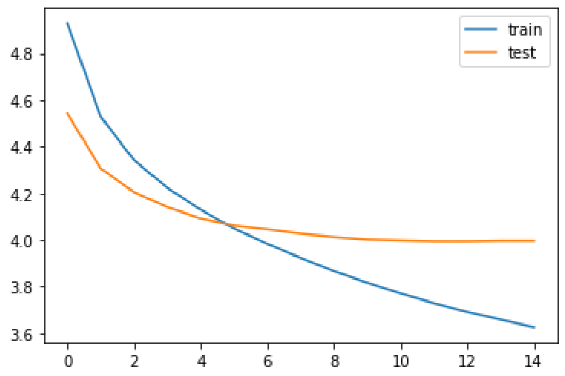
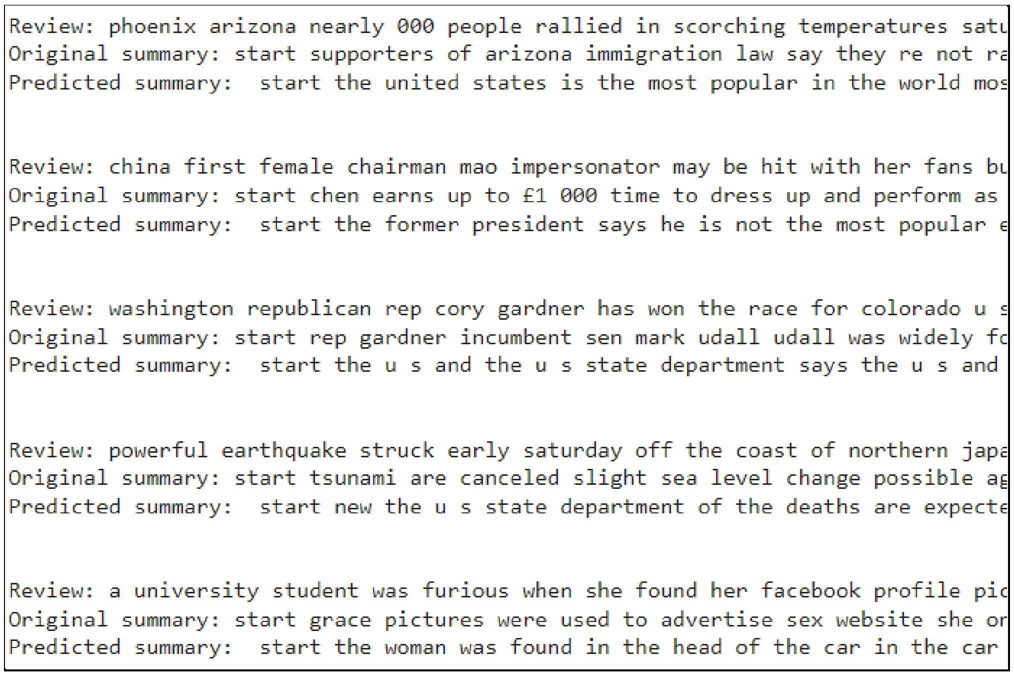


Figure 5*. Training/validation loss per epoch.*

Dan Waters · danwaters@my.unt.edu
University of North Texas
CSCE 5290 · Natural Language Processing
Fall 2021

Iteration 1 Results - Seq2Seq on CNN Dataset

Because of the runtime disconnection and a new train/test randomized split resulting in different embeddings, the summaries did not make much sense. It was likely they were summarizing other articles because of the data shuffling, which I remedied in a later training run. For example:



```
Review: phoenix arizona nearly 000 people rallied in scorching temperatures satu
Original summary: start supporters of arizona immigration law say they re not ra
Predicted summary:  start the united states is the most popular in the world mos


Review: china first female chairman mao impersonator may be hit with her fans bu
Original summary: start chen earns up to £1 000 time to dress up and perform as
Predicted summary:  start the former president says he is not the most popular e


Review: washington republican rep cory gardner has won the race for colorado u s
Original summary: start rep gardner incumbent sen mark udall udall was widely fo
Predicted summary:  start the u s and the u s state department says the u s and


Review: powerful earthquake struck early saturday off the coast of northern japa
Original summary: start tsunami are canceled slight sea level change possible ag
Predicted summary:  start new the u s state department of the deaths are expecte


Review: a university student was furious when she found her facebook profile pic
Original summary: start grace pictures were used to advertise sex website she or
Predicted summary:  start the woman was found in the head of the car in the car
```

Figure 6*. Mismatched results due to kernel restart and shuffled split with no random seed.*

Ignoring the article mismatches here, a couple of things are apparent. First, the classic "RNN Repetition" artifact is on full display. The full text of the final example reads:

```
the woman was found in the head of the car in the car on sunday morning
the incident happened at the scene of the road and was taken to hospital
the driver was taken to hospital after being taken to hospital
```

Since the articles are mismatched, there is no point in offering a ROUGE score for this attempt.

Iteration 2 Results

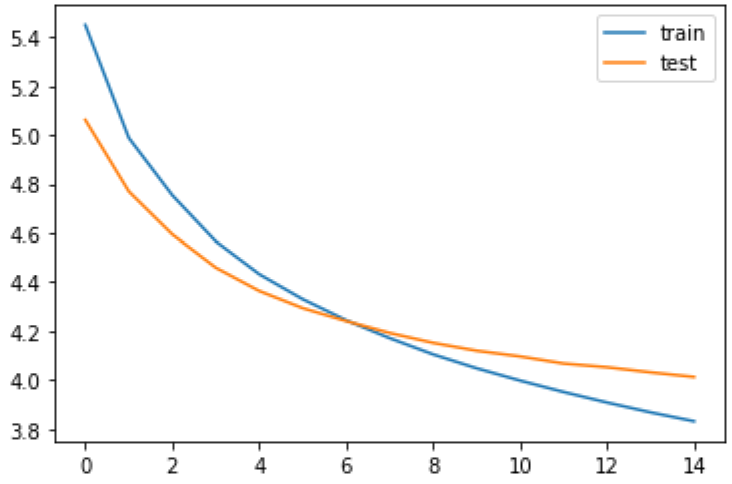
The notebook for this training session can be found here.

Dan Waters · danwaters@my.unt.edu
University of North Texas
CSCE 5290 · Natural Language Processing
Fall 2021

The major improvements in this notebook are: firstly, the summaries are for the correct articles, and secondly, ROUGE scoring metrics are implemented. There is still quite a lot of repetition in the example summary outputs, for example:

```
new york police say they are accused of killing of the murder of the murder
he was arrested in the shooting of the shooting in the capital of the two
year old was arrested in the capital of the attack
```

Due to time constraints, this model was only trained for 15 epochs, but the validation loss is starting to level off by this time. A few more epochs may have been slightly beneficial.



Scoring this summary with ROUGE gets us the following result:

LEAD-3 benchmark (ROUGE-1 F1 score, unigrams): ***40.42***

Score	F1	Precision	Recall
ROUGE-1 (Unigram)	**22.77**	65.14	13.79
ROUGE-2 (Bigram)	08.79	20.67	05.58
ROUGE-L (Longest Common Subsequence)	07.59	21.71	04.60

Needless to say, these results are well below the state-of-the-art benchmark, but this is a hand-built model trained on only a subset of the available data, whereas the BART and T5 models are trained with a tremendous amount of time, resources, and data.

Dan Waters · danwaters@my.unt.edu
University of North Texas
CSCE 5290 · Natural Language Processing
Fall 2021

Solution 2: BART Summarizer

A much easier undertaking than a custom Seq2Seq model, the BART model provided by HuggingFace's Transformers library is already fine-tuned on the CNN/Dailymail dataset. Code for this example can be found in this Jupyter notebook:

- **Dan Waters - CNN summarization with BART and T5 (HuggingFace).ipynb**

Across 100 articles sampled from my test dataset, the average ROUGE scores are presented below:

*LEAD-3 benchmark (ROUGE-1 F1 score, unigrams): **40.42***

Score	F1	Precision	Recall
ROUGE-1 (Unigram)	**53.68**	46.75	63.01
ROUGE-2 (Bigram)	20.92	17.61	25.76
ROUGE-L (Longest Common Subsequence)	47.29	41.19	55.51

The BART model performs better than the baseline, suspiciously so… the reason is that the articles I predicted were actually a subset of the CNN training dataset. It's difficult to know which articles are included in this model, so I performed this experiment using the test dataset from Tensorflow Datasets, hoping that it is the held-out test set. Here are those results:

Score	F1	Precision	Recall
ROUGE-1 (Unigram)	**56.33**	48.33	67.52
ROUGE-2 (Bigram)	21.83	17.85	28.11
ROUGE-L (Longest Common Subsequence)	50.06	42.94	60.00

The results seem better on the test set, surprisingly, so we'll go with that. Again, much of this has to do with the fact that we only had time to average over 100 predictions.

Solution 3: Google T5 Summarizer

The code for this example is in the same notebook as the BART model.

Dan Waters · danwaters@my.unt.edu
University of North Texas
CSCE 5290 · Natural Language Processing
Fall 2021

With 100 sampled articles, the average ROUGE scores are:

*LEAD-3 benchmark (ROUGE-1 F1 score, unigrams): **40.42***

Score	F1	Precision	Recall
ROUGE-1 (Unigram)	**57.40**	49.71	67.92
ROUGE-2 (Bigram)	21.85	18.13	27.49
ROUGE-L (Longest Common Subsequence)	51.72	44.79	61.20

Results Comparison

Taking only the average ROUGE-1 F1 score across 100 examples, we have these final results:

*LEAD-3 benchmark (ROUGE-1 F1 score, unigrams): **40.42***

Seq2Seq (custom trained)	**BART (train)**	**BART (test)**	**T5 (train)**	**T5 (test)**
22.77	53.68	56.33	53.84	**57.40**

The T5 on 100 test set predictions shows the best performance by far.

Dan Waters · danwaters@my.unt.edu
University of North Texas
CSCE 5290 · Natural Language Processing
Fall 2021

Conclusion and Takeaways

There is really no match for a state-of-the-art, pre-trained transformer model such as BART or T5. It is incredibly time-consuming to train even a basic sequence-to-sequence LSTM model, and the time it takes to iterate and make changes is the stuff of full-time ML engineering. Not having that luxury right now, it's very easy for me to conclude that while BART and T5 have some subtle differences (the BART model chosen is fine-tuned on the CNN data set, whereas T5 is more generalized), one of the pretrained models would likely suit any summarization need better than the basic LSTM model I constructed.

Additionally, checkpointing, saving artifacts, and pre-processing data ahead of time are all very valuable time-saving tricks that should be identified as best practices.

Key Learnings:

- NLP at scale is non-trivial. As described at the beginning of this course, language is confusing, ambiguous, and difficult to represent in a way that computers can understand it.
- Training on a large amount of text data takes a very long time. Even other data processing techniques, such as using the SpaCy NLP pipeline to add start/end tokens, can take a very long time when you are dealing with 200k items.
- Selecting a tokenization and embedding scheme that is uniform and reproducible is extremely useful. For tasks such as this, bag-of-words makes no sense; such sparse vectors would take even longer to train.
- There is almost no chance that an individual part-time researcher will achieve the level of summarization quality offered by Google's T5 transformer model or the BART models offered by HuggingFace.
- This field is constantly growing and evolving, with new breakthroughs always on the horizon.

Dan Waters · danwaters@my.unt.edu
University of North Texas
CSCE 5290 · Natural Language Processing
Fall 2021

Appendix: Increment Reports

Increment 1

Overview

Since the original project proposal, I have learned a lot more about this task, and it became necessary to overhaul my research plans. We have also recently explored some key concepts in class that I did not previously know about, such as N-grams (just covered on October 27), which are critical to this project. In fact, understanding N-grams forms a basis for the ROUGE-1 and ROUGE-2 evaluation metrics, and I previously did not know how I would measure summarization results empirically. As a result of this pivot, I don't have any preliminary results of my own work yet, but will spend the next two increments doing so.

While I did experiment with my originally proposed GAN approach[5], it was yielding terrible results and is rather outdated, so I will not be discussing it here or pursuing GANs any further for this task. More modern approaches are discussed next.

Dataset

The dataset for this project is the containing stories and questions for CNN and Daily Mail news corpora.

Related Work and Background

During this increment, I discovered a helpful meta-resource on Papers with Code containing modern approaches and results specific to this task and dataset. This resource clearly documents the ROUGE performance of various models (Figure 1).

[5] Liu, Linqing et al. "Generative Adversarial Network for Abstractive Text Summarization." Accessed 20 Sep 2021, https://arxiv.org/abs/1711.09357

Dan Waters · danwaters@my.unt.edu
University of North Texas
CSCE 5290 · Natural Language Processing
Fall 2021

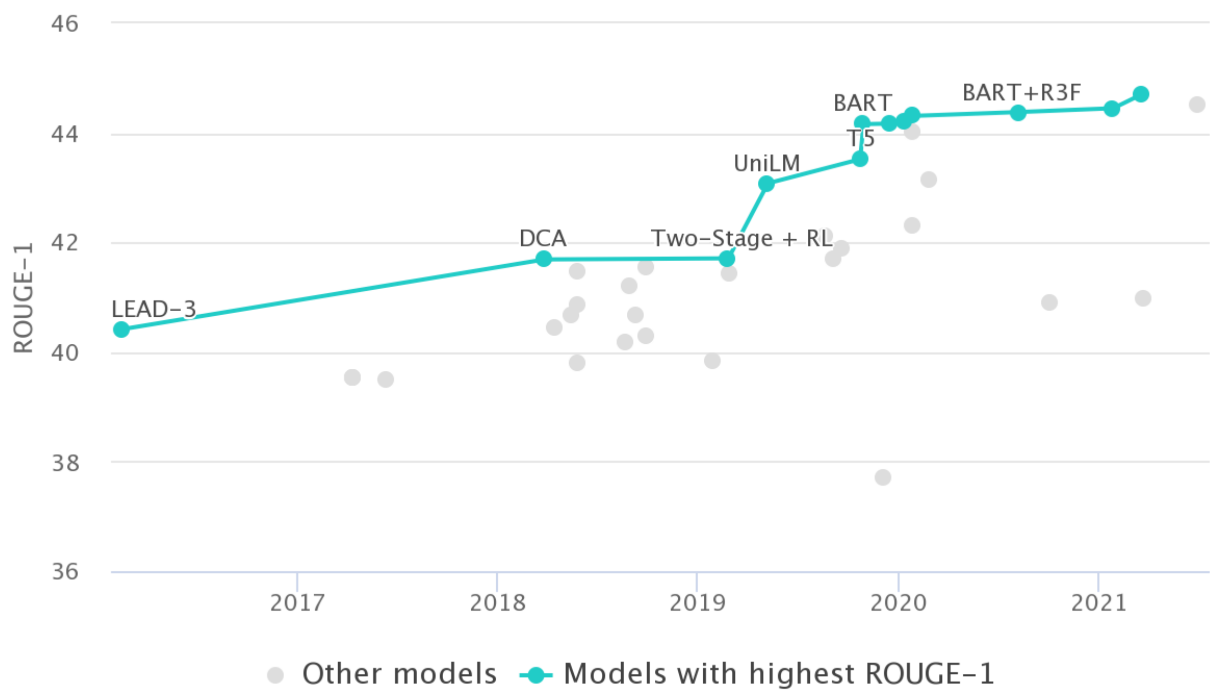


Figure 1: Abstractive Text Summarization on CNN / Daily Mail leaderboard. Source: Papers with Code

According to this leaderboard, the top models and their ROUGE-1 scores are:

Model	**ROUGE-1 Score**	**Year**
GMM-XXLarge	44.7	2021
BART + R-Drop	44.51	2021
MUPPET BART Large	44.45	2021

The benchmark score is LEAD-3 with a score of 40.42, based on sequence-to-sequence recurrent neural networks.

The papers supporting the top-performing models are far beyond my current understanding and involve some pre-trained models, so for the next increment I will be focusing on something less complex that I can implement myself in the time available.

Other approaches to this problem include sequence-to-sequence RNNs, pointer-generator networks, and transformers with attention. Transformers seem to be the favorite approach. Over the next two increments, I aim to explore all three approaches, starting with the techniques in the LEAD-3 model.

Limitations of Existing Solutions

Several limitations exist with the models available today. I have observed summarizations which:

- are grammatically correct, but factually incorrect,
- miss key context,
- are too long or too short,
- fall into repetitive sequences (particularly with the RNN approach).

A more detailed analysis of which models exhibit which behaviors will be provided in the next increments, as experimental results become available.

Detailed Design

To be completed in the next increment.

Analysis

To be performed in the next increment.

Implementation

To be completed in the next increment.

Preliminary Results

To be discussed in the next increment.

Project Management

High-Level Status Update

There is much more to do. As of October 31, I have only been able to research techniques, set up the data sets, and do some initial data cleaning, which I cover in my video. However, I believe I am set up to be successful in the next two increments of this project.

Work Completed

Task	**Description**
Dataset acquisition	Downloaded the four dataset components (CNN Stories, Daily Mail Stories, CNN Questions, Daily Mail Questions) and hosted them in my Google Drive where they can be downloaded into Google Colab
Data preprocessing	Using some ideas from this article[6], put together a notebook which preprocesses a given story. This is organized to be flexible so that other preprocessing techniques can easily be applied and modified.
Method investigation	Researched more modern techniques for abstractive text summarization using recent learnings about n-grams.

Work To Be Completed

Task	**Description**
Validate preprocessing	Make any iterative improvements to the preprocessing algorithm (e.g. embeddings, lemmatization or stemming if necessary, as required by the LEAP-3 approach)
Build sequence-to-sequence model	Build a LEAP-3 style Seq2Seq RNN
Evaluate sequence-to-sequence model	Evaluate LEAP-3 model against ROGUE-1 metric benchmark of 40.42
Build pointer-generator model	Use pointer-generator technique
Evaluate pointer-generator model	Evaluate pointer-generator technique against an acceptable benchmark

[6] Brownlee, Jason. "How to Prepare News Articles for Text Summarization." *Machine Learning Mastery*, 7 August 2017. Accessed 15 October 2021, https://machinelearningmastery.com/prepare-news-articles-text-summarization/

Build transformer model	Select a transformer / attention model to try
Evaluate transformer model	Evaluate transformer model against benchmark ROGUE-1 metric
Comparative analysis	Compare metrics and example summaries from each model to inform conclusions
Write conclusion	Use data from comparative analysis to make definitive statements about the methods explored during this project.

Increment 2

Overview

Working with the CNN/Dailymail dataset for the last several weeks has been somewhat frustrating due to its scale. Not only are the articles very numerous, but each article itself is very long on average (mean of 691 words per article). The headline sections are also very long for summaries. This means that preprocessing, tokenization, and embedding take some time. Training also takes a long time, as smaller batches are required to avoid running out of memory.

As a result, I took a step back and conducted an exploratory data analysis with the CNN/Dailymail dataset versus another popular source, the News Summary dataset from Kaggle.

Dataset

The original dataset for this project is the DeepMind Q&A dataset[7] which contains stories and questions for CNN and Daily Mail news corpora.

However, a smaller dataset has proven extremely useful as well: the News Summary Dataset from Kaggle.

I performed an exploratory data analysis between these two datasets to understand some basic statistics about scale. Here, I am comparing the entire News Summary Dataset with the training set from Tensorflow Data Sets.

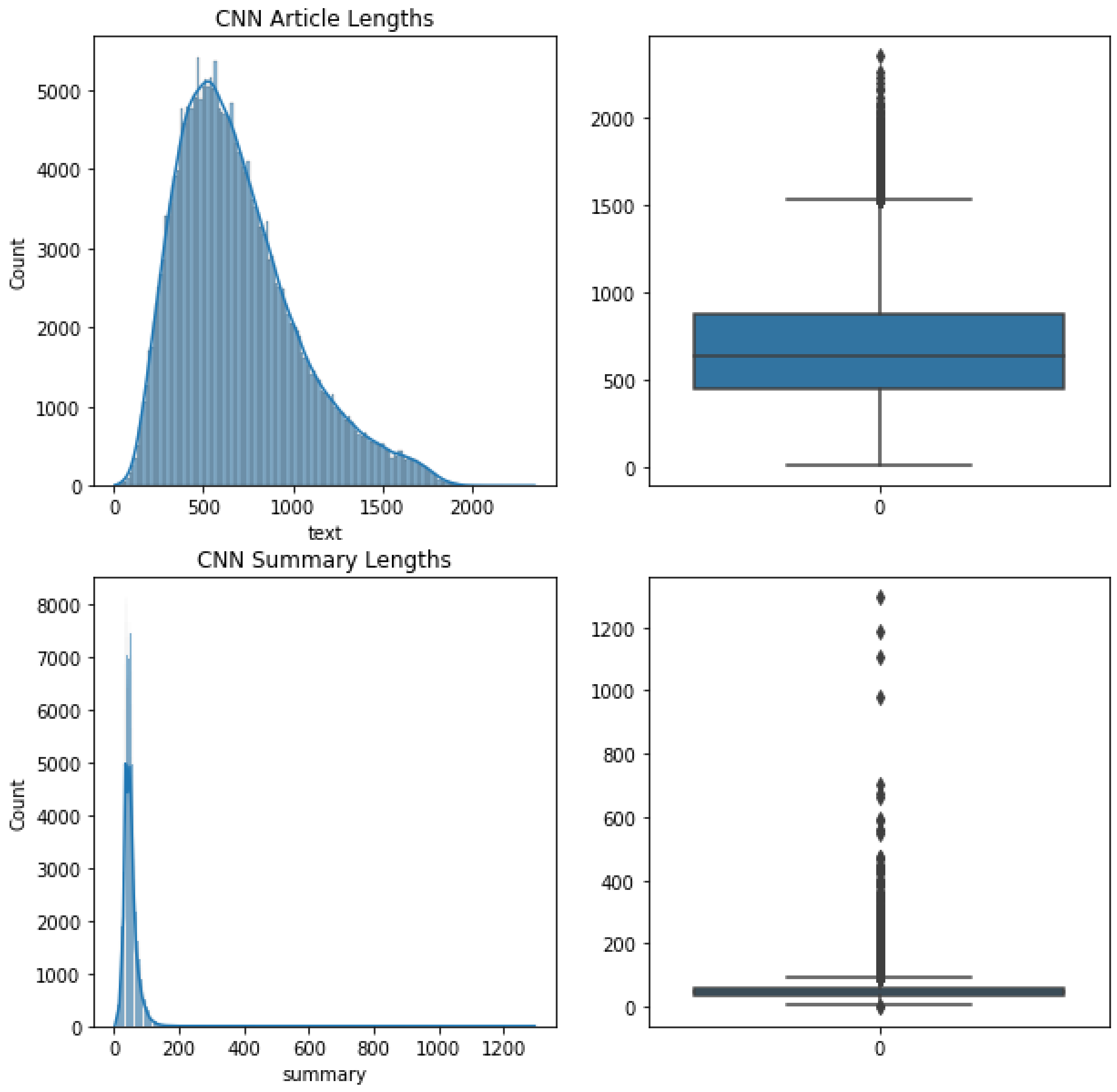
[7] Cho, Kyunghun. "DeepMind Q&A Dataset." Accessed 20 Sep 2021, https://cs.nyu.edu/~kcho/DMQA/

Dan Waters · danwaters@my.unt.edu
University of North Texas
CSCE 5290 · Natural Language Processing
Fall 2021

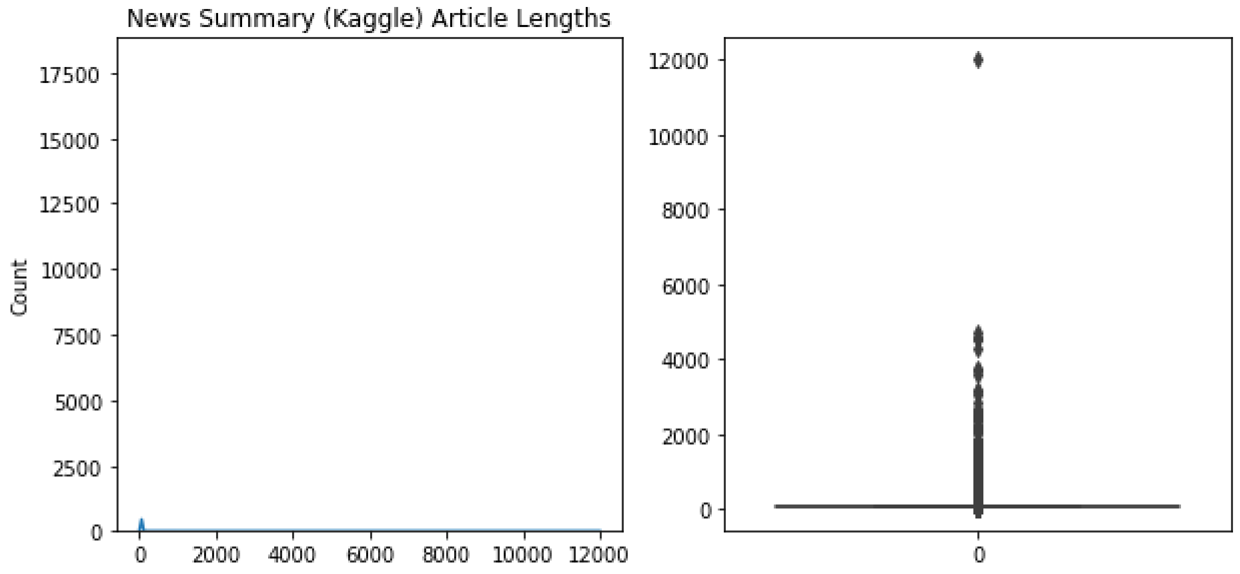
Statistic	**CNN/Dailymail**	**News Summary (Kaggle)**
Total number of examples	287,113	102,915
Min / max article length	8 / 2,347	1
Min / max summary length	4	11,999
Mean article length	691	71
Mean summary length	51	11

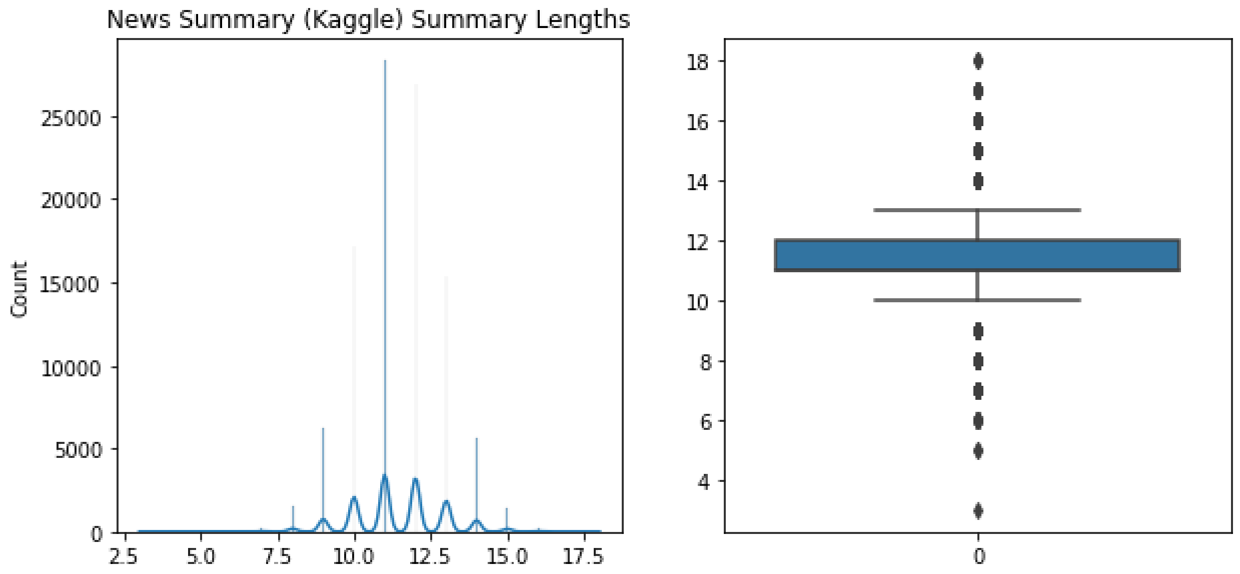
Below is more information about the distributions for each data set (summary and article lengths):

Dan Waters · danwaters@my.unt.edu
University of North Texas
CSCE 5290 · Natural Language Processing
Fall 2021



Dan Waters · danwaters@my.unt.edu
University of North Texas
CSCE 5290 · Natural Language Processing
Fall 2021





Using an arbitrary threshold of articles between 20 and 500 words and summaries between 10 and 50 words, the CNN corpus contains 65,902 examples, whereas the News Summary dataset contains 94,087 such examples. I proceeded to train the Seq2Seq model with the CNN corpus. As soon as training finished, about 14 hours later the following afternoon, I saved the entire model architecture, but lost connection to the runtime before I could run a full set of predictions using the same tokenizer settings that had been created from a random split. As a result, I was not able to recreate the correct summaries on the CNN dataset, but I was able to recover the summaries from the News Summary dataset and score them using ROUGE. Given more time (and computing power), retraining the model on the CNN data using a fixed split would be ideal.

Dan Waters · danwaters@my.unt.edu
University of North Texas
CSCE 5290 · Natural Language Processing
Fall 2021

Related Work and Background

Increment 2 was mostly spent working around Seq2Seq using the tutorial, but applying this approach to the truncated CNN/Dailymail dataset.

Project Management

High-Level Status Update

With all of the work and time that went into training the sequence-to-sequence encoder/decoder LSTM models, there was no time remaining to build transformers or pointer-generators from scratch, but I was able to evaluate BART and T5 transformers from the HuggingFace library.

Work Completed

Task	**Description**
Dataset acquisition	Completely retooled dataset acquisition workflow, using Tensorflow Datasets instead.
Data preprocessing	Completely reworked the preprocessing and snapshotted the CNN data at a smaller scale (50k examples, already processed). This can be experimented with much more easily.
Method investigation	Researched Seq2Seq in great detail. Implemented BART and T5.
Models trained	● Seq2Seq Tutorial (News Summary dataset) ● Seq2Seq with CNN dataset and ROUGE metrics ○ Several versions iterated here… only kept the final one as a notebook as the earlier ones had very poor results and small mistakes. ● Implemented BART and T5 predictions using CNN test set, and offered ROUGE metrics for each.

Dan Waters · danwaters@my.unt.edu
University of North Texas
CSCE 5290 · Natural Language Processing
Fall 2021

Resources

All work thus far has been committed to this GitHub repository:
https://github.com/danwaters/nlp-abstractive-text-summarization

Project proposal can be found here:
https://github.com/danwaters/nlp-abstractive-text-summarization/blob/main/CSCE%205290%20Project%20Proposal%20-%20Draft%20(Waters).pdf

This document (Increment 1):
https://github.com/danwaters/nlp-abstractive-text-summarization/blob/main/CSCE%205290%20Project%20Increment%201%20(Waters).pdf

References

1. "ROUGE (metric)." Wikipedia, Wikimedia Foundation, 17 September 2021, https://en.wikipedia.org/wiki/ROUGE_(metric)
2. Liu, Linqing et al. "Generative Adversarial Network for Abstractive Text Summarization." Accessed 20 Sep 2021, https://arxiv.org/abs/1711.09357
3. Vonteru, Kondolarau. "NEWS SUMMARY / Generating short length descriptions of news articles." Kaggle. Accessed 11/01/2021, https://www.kaggle.com/sunnysai12345/news-summary
4. Cho, Kyunghun. "DeepMind Q&A Dataset." Accessed 20 Sep 2021, https://cs.nyu.edu/~kcho/DMQA/
5. "Abstractive Text Summarization on CNN / Daily Mail." Papers with Code, Facebook AI. Accessed 25 October 2021, https://paperswithcode.com/sota/abstractive-text-summarization-on-cnn-daily
6. Brownlee, Jason. "How to Prepare News Articles for Text Summarization." Machine Learning Mastery, 7 August 2017. Accessed 15 October 2021, https://machinelearningmastery.com/prepare-news-articles-text-summarization/