Text Summarization using BART-BASE

Balaji Revanth Guduru
Computer Science, 02004098
University of Massachusetts Lowell
balajirevanth_guduru@student.uml.edu

Ashwin Sankarasubramanian
Computer Science, 01987836
University of Massachusetts Lowell
ashwin_sankarasubramanian@student.uml.edu

Abstract

We have trained a pre-trained model facebook/bart-base on cnn-daily mail dataset for the task text summarization. The goal of this task is to summarize large sentences which have tokens more than 512 to smaller sentences of max tokens 128. We preprocessed the dataset for the model and by using pre-trained tokenizers. We used bartforconditionalgeneration from huggingface. We evaluated the model using rouge metric by logging rouge-1,rouge-2,rougeLsum for every 5000 steps in wandb. The training was done for 1 epoch with 20000 training steps, AdamW optimizer, learning rate 5e-5 and weight decay 0.0. The rouge1 score was 40.69.

1 Introduction

In a day-to-day life news reading is important and many don't have time to read long paragraphs. In order to help them we can give a summarized paragraph which is of few lines where they can get brief info of the long paragraph. The amount of data available online is limitless. Think about a normal college student, who has to go through thousands of pages of documents each semester. That is why text summarization is necessary. The main purpose of text summarization is to get the most precise and useful information from a large document and eliminate the irrelevant or less important ones.

Abstractive Text Summarization and Extractive Text Summarization are two approaches for extracting summary from a given text. The system provides a short summary without affecting the meaning of the material from the given text in Abstractive Text Summarization. In the case of Extractive Text Summarization, the system extracts a few key sentences from the given text that make the most sense of the whole.

People are obsessed with emerging technologies and social media in today's world. Everyone in today's generation uses their phone even if they only have a few seconds to spare. The majority of consumers prefer quick, concise news rather than extensive news on television or in the newspaper. As a result, developing a system that extracts summary from a given text or document without affecting the content's meaning is critical.

Text summarization can be done either manually, which is time-consuming, or through machine algorithms and AIs, which takes very little time and is a better option. Text summarization is difficult because, when we humans summarize a piece of text, we normally read it completely to have a thorough comprehension of it before writing a summary highlighting its important points. Because computers lack human understanding and linguistic skills, text summarization is a difficult and time-consuming task.

2 Related Work

The following four works provide insights into how the problem of extractive text summarization is approached and are the baselines for work that has been done in this area.

Extractive Text Summarization Using Deep Learning: This paper [2], proposed in 2018 suggests the use of a lateral combination of a deep neural network and a Fuzzy logic system. The suggested neural network type is the Restricted Boltzmann Machine (RBM). The sentences are independently fed into the neural network and fuzzy logic system to produce two different summaries. After performing a set union operation, we produce a final summary for the document.

Hybrid auto text summarization using deep neural network and fuzzy logic system: This paper [3] proposes a very similar architecture to approach the problem. They also make use of fuzzy logic systems and RBM. The key difference consists of a sequential processing rather than lateral. So the sentences are first classified by the fuzzy rules before being processed by the network. They also make use of distinct sentence features such as title similarity and named entity or numerical data sentence weights to produce a more accurate summary.

Extractive Text Summarization from Web pages using Selenium and TF-IDF algorithm: This paper [4] provides an approach to the data scraping process rather than the processing itself. The key concept is to input a search query rather than a whole input document. The query is then searched using one of the common web search engines. The top web page results of the search engines are retrieved and scrapped together to form the input document. This document is then summarized to produce the short summary of available web information on the search query.

Extractive Text Summarization Using Word Vector Embedding: This paper [5] introduces the use of pre-trained GLoVe vectors to form sentence representations. These pre-trained vectors are trained in an unsupervised manner over thousands or millions of sentences to produce billions of word tokens with varying dimensions. These word vectors are processed using fully connected multi-layer perceptron models to predict the probability of a sentence being included in the summary. This approach had better performance than all previous existing online summarization tools.

In recent years, improvements to abstractive document summarization models have been developed through the incorporation of pre-training.
BERTSUM: The BERTSUM model [6] has been proposed as a pre-training model for document summarization tasks. After obtaining the sentence vectors from BERT, it builds several summarization-specific layers stacked on top of the BERT outputs, to capture document-level features for extracting summaries. For each sentence, it calculates the final predicted score. The loss of the whole model is the Binary Classification Entropy of predicted score against gold label. These summarization layers are jointly fine-tuned with BERT.

T5: For sequence-to sequence tasks, the T5 model [7] and the BART model [8] have been proposed as part of generalized pre-training models. T5 is an encoder-decoder model pre-trained on a multi-task mixture of unsupervised and supervised tasks and for which each task is converted into a text-to-text format. T5 works well on a variety of tasks out-of-the-box by prepending a different prefix to the input corresponding to each task.

Among the existing pre-training models, the BART model achieves state-of-the-art performance on document summarization tasks.

3 Methodology

3.1 BART:

BART uses the standard sequence-to-sequence Transformer architecture except, following GPT, that we modify ReLU activation functions to GeLUs and initialize parameters from N (0, 0.02). For the base model, there are 6 layers in the encoder and decoder, and for the large model there are 12 layers in each. The architecture is closely related to that used in BERT, with the following differences:

Each layer of the decoder additionally performs cross-attention over
the final hidden layer of the encoder (as in the transformer sequence-to-sequence model)
BERT uses an additional feed-forward network before word prediction, which BART does not. In total, BART contains roughly 10% more parameters than the equivalently sized BERT model.

We do not make any modifications to the existing architecture of the BART model for our implementation. The model architecture specific parameters we provide during model initialization are the maximum input sequence size of 1024 tokens and the maximum output sequence size of 128 tokens.

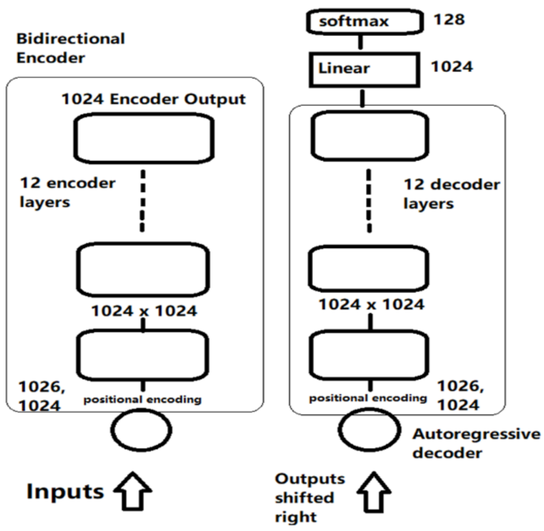

Figure 1: The implemented BART model architecture and specifications.

3.2 Model Initialization

We used BartForConditionalGeneration [9] module to import the BART model. For the baseline, we have initialized a model with a randomly generated BartConfig checkpoint. For the final model, we have imported the facebook/bart-base checkpoint and fine-tuned the model.

3.3 Tokenizer

We have used pre-trained BartTokenizer [10] for converting the dataset values into word tokens.

3.4 Pre-Processing

We added prefix summary to inputs and replaced the labels having value 1 with -100 value.

3.5 Batch Size

We used a batch size of 8 for both the baseline execution as well as the final model execution.

3.6 Optimizer

We have used the AdamW optimizer to fine tune both the pre-trained and the randomly initialized model checkpoints.
We have used a learning rate of 5e-5 for training the model. We used a weight decay of 0.0.

4 Dataset

The CNN / DailyMail Dataset [1] is an English-language dataset containing just over 300k unique news articles as written by journalists at CNN and the Daily Mail. The current version supports both extractive and abstractive summarization, though the original version was created for machine reading and comprehension and abstractive question answering.

The BCP-47 code for English as generally spoken in the United States is en-US and the BCP-47 code for English as generally spoken in the United Kingdom is en-GB. It is unknown if other varieties of English are represented in the data.

The dataset consists of 287,113 training samples, 13,368 evaluation samples and 11,490 testing samples. The mean token count of the articles or documents in the data set is 781. Similarly, the mean token count of the highlights or ground truth summaries in the data set is 56.

Document:
(CNN) -- An American woman died aboard a cruise ship that docked at Rio de Janeiro on Tuesday, the same ship on which 86 passengers previously fell ill, according to the state-run Brazilian news agency, Agencia Brasil. The American tourist died aboard the MS Veendam, owned by cruise operator Holland America. Federal Police told Agencia Brasil that forensic doctors were investigating her death. The ship's doctors told police that the woman was elderly and suffered from diabetes and hypertension, according the agency. The other passengers came down with diarrhea prior to her death during an earlier part of the trip, the ship's doctors said. The Veendam left New York 36 days ago for a South America tour.'

Ground truth summary:
The elderly woman suffered from diabetes and hypertension, ship's doctors say .\nPreviously, 86 passengers had fallen ill on the ship, Agencia Brasil says .

Figure 2 Sample data from the CNN Daily Mail dataset

5 Experimental setup and evaluation methods

GPU:

We have use Nvidia-A100 GPU available on the google cloud platform to train the model. We created a new VM instance with the GPU and executed the code from the GitHub URL:

https://github.com/Revanth-guduru-balaji/FinalProject-NLP.

ROUGE:

ROUGE, or Recall-Oriented Understudy for Gisting Evaluation, is a set of metrics and a software package used for evaluating automatic summarization and machine translation software in natural language processing. The metrics compare an automatically produced summary or translation against a reference or a set of references (human-produced) summary or translation.
ROUGE-1 refers to the overlap of unigram (each word) between the system and reference summaries.
ROUGE-2 refers to the overlap of bigrams between the system and reference summaries.
ROUGE-L: Longest Common Subsequence (LCS) based statistics. Longest common subsequence problem takes into account sentence level structure similarity naturally and identifies longest co-occurring in sequence n-grams automatically.

In the ROUGE paper, two flavors of ROUGE are described:
1. sentence-level: Compute longest common subsequence (LCS) between two pieces of text. Newlines are ignored. This is called rougeL in this package.
2. summary-level: Newlines in the text are interpreted as sentence boundaries, and the LCS is computed between each pair of reference and candidate sentences, and something called union-LCS is computed. This is called rougeLsum in this package.

We used rouge metric from datasets library to evaluate the model. For every 5000 steps we evaluate the model on whole evaluation dataset.
We considered mid value of rouge-1, rouge-2, rougeLsum.

BEAM SEARCH:
Beam search is an algorithm used in many NLP and speech recognition models as a final decision-making layer to choose the best output given target variables like maximum probability or next output character. We used beam search with beam size 5.

6 RESULTS:

The results for the final model are as below:
Rouge1: 40.69
Rouge2: 18.89
RougeL: 28.36
RougeLsum: 37.89

We compare this with the baseline of the same model that has been randomly initialized and trained from scratch without pretraining. The baseline model metrics were:

Rouge1: 9.23
Rouge2: 0.15
RougeL: 8.61
RougeLsum: 8.10

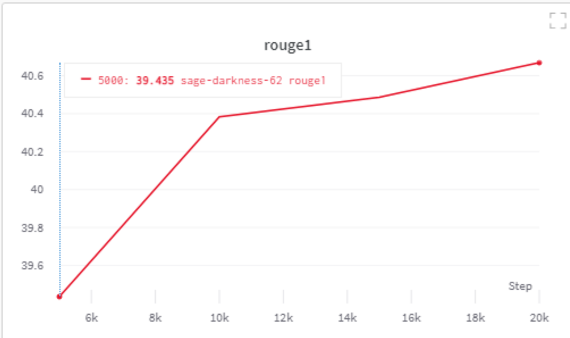

Figure 3 ROUGE-1 Score

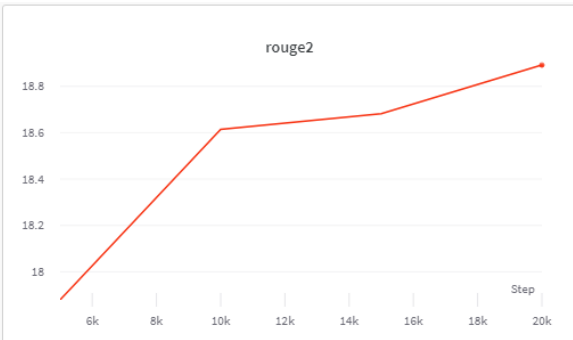

Figure 4 ROUGE-2 Score

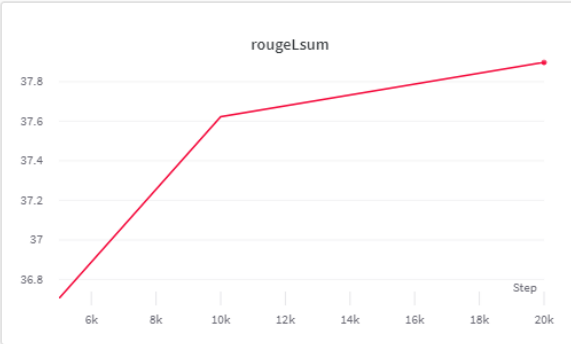

Figure 5 ROUGE-Lsum Score

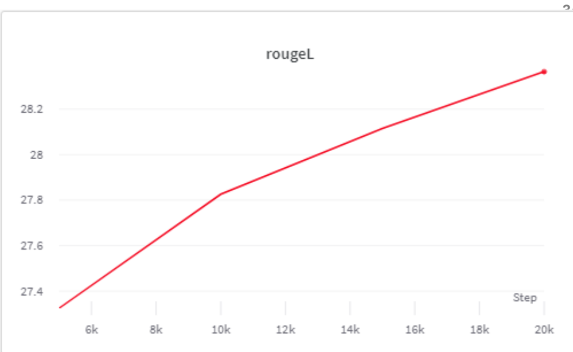

Figure 6 ROUGE-L Score

CONCLUSION:

The pre-trained model performed way better than the baseline randomly initialized model checkpoint. We were able to see the ROUGE-1 score increase from 9.2 to 40.7. The ROUGE-2 score improved from 0.15 to 18.9. The ROUGE-L score increased from 8.6 to 28.4. The ROUGE-Lsum score increased from 8.1 to 37.9.

REFERENCES:

[1] https://huggingface.co/datasets/cnn_dailymail

[2] Nikhil S. Shirwandkar and Samidha Kulkarni, “Extractive Text Summarization Using Deep Learning“ 2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA)

[3] Heena A. Chopade and Meera Narvekar, “Hybrid auto text summariza-

tion using deep neural network and fuzzy logic system“ 2017 Interna-

tional Conference on Inventive Computing and Informatics (ICICI)

[4] K Usha Manjari, Syed Rousha, Dasi Sumanth and J Sirisha Devi,

“Extractive Text Summarization from Web pages using Selenium and

TF-IDF algorithm“ 2020 4th International Conference on Trends in

Electronics and Informatics (ICOEI)

[5] Aditya Jain, Divij Bhatia and Manish K Thakur, “Extractive Text

Summarization Using Word Vector Embedding“ 2017 International

Conference on Machine Learning and Data Science (MLDS)

[6] Yang Liu, “Fine-tune BERT for Extractive Summarization” 5 Sep 2019

[7] 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” 28 Jul 2020

[8] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov and Luke Zettlemoyer, “BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension” 29 Oct 2019

[9] Bart for Conditional Generation:

https://huggingface.co/docs/transformers/model_doc/ bart#transformers.BartForConditionalGeneration

[10] Bart tokenizer:

https://huggingface.co/docs/transformers/model_doc/ bart#transformers.BartTokenizer