Final report of MSML 651 project

Xiyuan Liu
Xliu12@umd.edu
Instructor: Dr. Paul Rodrigues
MSML 651

Abstract

In the 21st century, there has been an exponential increase in data generation, particularly textual data. This surge is mainly attributed to advancements in technology such as the Internet, social media platforms, and sophisticated storage systems. Consequently, managing and extracting meaningful insights from these massive volumes of text poses significant challenges. Traditional methods for processing and storing data are no longer practical given the sheer scale of information available. As a result, Artificial Intelligence (AI) has emerged as a promising solution for efficient data management and extraction of valuable insights. Within this context, Natural Language Processing (NLP), specifically text summarization techniques have become indispensable for condensing lengthy textual documents into concise representations. However, performing text summarization on enormous datasets presents its own set of unique challenges. One such hurdle is the impracticality of loading massive amounts of data into memory due to resource limitations. To overcome this obstacle effectively without compromising performance or accuracy levels, this paper strategically employs the CNN-DailyMail News dataset while leveraging Azure Databricks - a distributed computing platform designed explicitly for preprocessing large-scale datasets. To tackle the task at hand efficiently with minimal computational overheads, a Seq2Seq model is utilized alongside Tensor Processing Units (TPUs) within Colab's distributed computation environment. By embracing alternative methodologies, the study also evaluates Google's Text-To-Text Transfer Transformer model (T5) as an additional approach that complements existing techniques employed in text summarization tasks. By utilizing Rouge scores as a metric to evaluate model performance, it becomes evident that the T5 model outshines its counterpart - the Seq2Seq model. Remarkably, the T5 model surpasses the Seq2Seq in various aspects such as unigram and bigram overlaps along with longest common subsequence matches. When assessed against alternative models like Scrambled Code + Broken and GPT-2, benchmarking validates the promising potential of using T5 for large-scale text summarization purposes.In essence,this research paper presents an all-encompassing approach towards addressing large-scale text summarization challenges by synergistically combining distributed computing techniques alongside both spark, Seq2Seq and T5 models.

1 Introduction

Since humanity entered the 21st century, the development of technology including the internet, improvement of social media and data storage technology like cloud and decentralized storage, caused the data size of stored text data to grow rapidly. An in-depth analysis of Twitter data storage conducted by Ankush Chavan highlights the enormity of this growth. Each succinct 140-character tweet contributes to a daily generation of an astounding 12 terabytes of data, translating to 2520 terabytes per week and approximately 4.3 petabytes annually (Chavan, 2020).

However, managing and analyzing such vast data sets poses considerable challenges. It is impractical to process or storage this tremendous amount of data with traditional processing and storage methods. Also by the need to extract meaningful insights from this data lake, the reliance on Artificial Intelligence (AI) becomes more imperative than ever since manual processing is no longer viable due to the sheer volume of the task. As one of the NLP tasks, text summarization becomes a crucial tool to assist humans to understand the content of massive data. Text summarization is a process that distills extensive documents into concise representations without compromising essential meaning.

However, executing text summarization on colossal data sets presents its own set of challenges, for example, loading massive volumes of data into

RAM becomes impractical. To simulate and address this challenge, my project strategically utilizes the CNN-DailyMail News data set as the input data. To address the challenges associated with executing text summarization on colossal data sets, my project strategically adopts Azure Databricks for data preprocessing. By using its property of distributed computing, the problem of limitations related to data loading into RAM has been overcome. Additionally, I implemented a Seq2Seq model with distributed computation using TPU in Colab to address text summarization tasks.

In my exploration of alternative methodologies, considering the complexity of the summarization task and the data sample which has about 95 percent samples containing more than 700 words in the article, I have also evaluated Google's Text-To-Text Transfer Transformer model (T5) as a complementary approach. This multifaceted strategy, integrating Azure Databricks, Seq2Seq, and T5, is designed to tackle the complexities of text summarization on a large scale. Through these deliberate choices, my objective is not only to surmount technical constraints but also to extract meaningful insights from large textual data generated daily from CNN News.

2 Methodology

Spark in Azure Databricks for Data Preprocessing

In my initial strategy, I opted to leverage Azure Databricks for data preprocessing using spark, complemented by Colab for model training and prediction. As I explained above, given the sheer volume of data, the utilization of Azure Databricks for big data engineering emerged as a more appropriate choice compared to non-distributed systems. The distributed nature of Azure Databricks allows for parallel processing, enabling efficient handling of large data sets, which is crucial for streamlined data preprocessing (Weber,2019).

To establish a Databricks workspace within Microsoft Azure, the initial step involves the creation of a new subscription featuring pay-as-you-go pricing. This choice is crucial, as it unlocks the capability for employing multiple nodes. It's worth noting that alternative subscriptions, such as free trials and student trials, impose limitations by allowing the creation of only a single node within the workspace.

Following the subscription setup, the subsequent stage entails the creation of a new cluster, configuring the computational environment of the workspace. For optimal performance, the runtime is set to version 14.0, which encompasses Apache Spark 3.5.0 and Scala 2.12. The chosen worktype is `standard_DS3_v2`, featuring at least 2 workers to harness parallel processing capabilities. Upon the successful creation of the cluster, the next step involves the initiation of a new notebook, closely associated with the cluster. After notebook creation, I upload the raw data set utilizing the Databricks File System (DBFS) and create a table with UI. Up to this point, the configuration of the notebook has been set.

Within the notebook, the initial action involves the creation of a Spark session, followed by the loading of the data set. Subsequently, the focus shifts to data preprocessing The data preprocessing include the removal of empty or duplicate lines, contraction mapping, lowercase conversion, elimination of apostrophes, punctuation, numbers, newline characters, and any unnecessary whitespaces.The subsequent step employs the PySpark machine learning library to tokenize both the articles and highlights and the remove stopwords.As the final step in the preprocessing journey, the processed data set is saved in the Databricks File System (DBFS) directory which can be downloaded by using API.

Colab for Seq2Seq Model Deployment

In the Colab environment, the primary objective is to deploy a Seq2Seq model for text summarization. To harness the power of Tensor Processing Units (TPUs) during training, the notebook has been configured to a runtime tailored for TPUs with high RAM. Initialization of the TPU and configuration of the distribution strategy within the notebook enable the creation and execution of models under the TPU framework, facilitating distributed training.

Once the data is loaded from Google Drive, our initial step involves conducting a comprehensive statistical analysis on the data set. This analysis is crucial for determining suitable lengths for articles and highlights, calculating the total number of tokens for training and testing sets which make a huge impact on the model training. For instance, by establishing proper article and highlight lengths,overly long inputs that might deviate from regular expectations can be eliminated. Addition-

ally, setting a threshold for the total token count in the training and test sets prevents rare tokens from disproportionately influencing the model's dictionary.

Following the statistical analysis and cleanup, subsequent steps involve meticulous tokenization of the input data, converting them into vectors, and padding to a uniform size for streamlined processing within the model. The subsequent stage entails the creation of the Seq2Seq model which is divided into two main parts: the encoder and the decoder. The encoder extracts features from the article vector, while the decoder utilizes the encoder's output as the initial state, predicting the next token given the current input token (Sutskever et al.,2014). The encoder consists of an embedding layer and three LSTM layers, while the decoder encompasses an embedding layer, an LSTM layer, an attention layer, and a dense layer. During the training phase, the model operates with a batch size of 128 over 50 epochs. To optimize training efficiency, an early stop mechanism is implemented, monitoring the validation loss to halt the training process when necessary.

Once the model is defined and trained, the next critical step is the creation of an Inference model for generating summarized text. Notably, the decoder model, trained to use a vector as the initial state and predict the next token based on the current hidden state and input token, is not inherently designed to generate each token in the predicted summary. To the trained model to predict, the inference model becomes essential as a decode tool.

In the decoding process, the inference model employs greedy search algorithms as a unique strategy. This algorithm systematically captures the most probable token at each step, progressing iteratively until it detects the appearance of the end sign, signaling the completion of the summary. By employing this greedy search algorithm, the inference model can generate a coherent and contextually relevant summarized text. It strategically selects the most likely tokens, aligning with the model's training objective, and stops the generation process when the end sign is detected, resulting in a logically structured and meaningful summary.

T5 Model

As mentioned earlier, in the exploration of alternative methodologies, the consideration of the Text-To-Text Transfer Transformer model (T5) comes into play, particularly given the inherent complexity of the summarization task. T5, renowned for its robust capabilities in natural language generation tasks, offers distinct advantages over the previously implemented Seq2Seq model.

One notable advantage is T5's extensive pre-training on a large data set called c4, comprising over 300 million training samples and 300 thousand validation samples. The utilization of such a large, cleaned pre-training data set contributes to enhanced model performance which leads to better performance compared to the models without pre-training. Another significant advantage lies in T5's advancements in attention mechanisms. While both T5 and Seq2Seq models leverage attention mechanisms, T5 benefits from sophisticated variants, such as the self-attention mechanism in transformers. This feature enables T5 to effectively capture long-range dependencies and contextual information, presenting a notable improvement compared to traditional Seq2Seq models.

For the deployment of T5 in text summarization, the initial step involves loading input data using a data loader provided by the data sets library. Subsequently, an auto tokenizer is created from the transformer library. It's noteworthy that the input data doesn't require preprocessing functions like stopword removal or punctuation removal. The auto tokenizer is then employed to transform the data into vectors of `input_ids` and `attention_mask`. Following this, a `Seq2SeqTrainingArguments`object is created, encompassing various training parameters that dictate how the model is trained. Importantly, the parameter `predict_with_generate` must be set to True to enable the calculation of generative metrics like ROUGE and BLEU.

The subsequent step involves the instantiation of a `DataCollatorForSeq2Seq` object using the tokenizer. Additionally, the ROUGE code is downloaded using the `load_metric` function from the data sets library, thereby creating a metric object. A `compute_metrics` function is then devised, serving the purpose of decoding predictions and computing ROUGE scores using the decoded predictions and labels, with a selection of specific metrics.

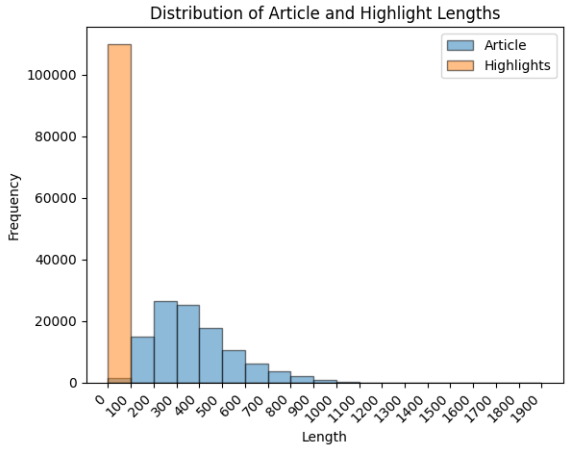
The final phase involves the creation of a Seq2SeqTrainer, incorporating all the previously defined objects – the training arguments, the training and evaluation data, the data collator, the tokenizer, the `compute_metrics` function, and a `model_init` function. Finally, the fine-tuning pro-

cess is initiated with the train method.

3 Statistical analysis on data set

In the context of our project, an integral step involves conducting a statistical analysis on the data to ensure the rationality and efficacy of the parameters set for the project. Two prominent statistical analyses were performed, focusing on the properties of article and summary lengths, as well as the total tokens in the dictionary of tokenizer.

To observe the distribution of article and summary lengths to set proper parameters, a comprehensive statistical analysis was held to the data set. The observations revealed a predominant trend wherein most highlights comprised fewer than 100 tokens, while the majority of articles fell within the range of 200 to 300 tokens.



To systematically filter out outliers, an examination of the distribution proportions was undertaken. Consequently, a reasoned determination was made to establish a maximum article length of 700 tokens and a maximum summary length of 60 tokens, as substantiated by the distribution patterns observed.

Table 1: Table of Article length and Summary length

Article length	**<= 600**	**<=700**	**<=800**
Percentage	0.8806	0.9358	0.9681

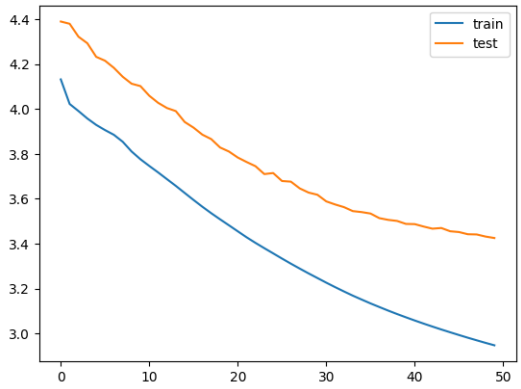
Summary length	**<= 50**	**<=60**	**<=80**	**<=100**
Percentage	0.9377	0.9764	0.9882	0.9941

An additional critical consideration pertains to the total tokens within the tokenizer's dictionary. The significance lies in the potential resource waste during training if there is an excessive presence of rare tokens. To mitigate this concern, a prudent measure has been implemented—a threshold of 10 occurrences. This signifies that in the training dataset, any unique token appearing less than 10 times is designated as a rare token. By applying this criterion and filtering out these rare tokens, the resulting input article vocabulary comprises 72,092 tokens, while the input summary vocabulary consists of 21,180 tokens. This strategic approach ensures that the dictionary is sufficiently rich for training purposes.

4 Model evaluation

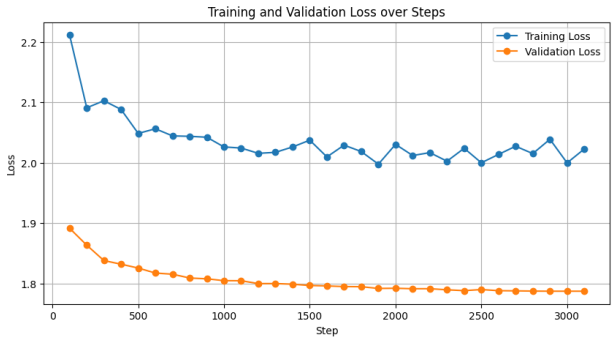
After the training phase, assessing the loss for both the training and validation sets becomes crucial in evaluating potential overfitting or underfitting of the model. For the Seq2Seq model, observing a consistent downward trend in both training and validation losses from the beginning to the end suggests that the model is maintaining a balanced learning process without exhibiting signs of overfitting or underfitting.

Figure 1: training and validation loss over epochs



In contrast, when evaluating the T5 model, we note some fluctuations in the training loss; however, the validation loss continues to decrease. This trend indicates that the T5 model is not overfitting. It's noteworthy that after approximately 1100 training steps, the validation loss stabilizes, suggesting that the model's performance reaches a plateau.

For model performance evaluation, rouge score has been used as the evaluation methods. The point of using rouge score is that it measures the similarity between the generated summary and the reference summaries using overlapping unigrams, bigrams, Longest Common Subsequence that appear in both the generated summary and the reference summaries. Even though rouge score may not fully



capture the semantic quality of the generated content such as Semantic Coherence, it is still a widely used evaluation metric for text summarization.

Table 2: Comparison of F1 Measures and Rouge Scores(Part 1)

Metric	Rouge1	Rouge2
Seq2Seq F1 measure	9.5746	1.3696
T5 F1 measure	27.1999	13.4676
Scrambled code + broken (alter)	48.18	19.84
GPT-2	29.34	8.27

Metric	RougeL
Seq2Seq F1 measure	8.4214
T5 F1 measure	22.6995
Scrambled code + broken (alter)	45.35
GPT-2	26.58

In the provided table, I present an evaluation of the performance of both the Seq2Seq model and the T5 model. Additionally, I include Rouge scores for two benchmark models: Scrambled Code + Broken, recognized for its robustness in Document Summarization on CNN/Daily Mail, and GPT-2, a well-known model developed by OpenAI (META AI). Focusing on the Seq2Seq model's performance, the Rouge1 score reveals a relatively low value of 9.6, indicating a suboptimal capture of key words from the article. The Rouge-2 score, assessing bigram overlap, is notably lower at 1.4, signaling challenges in identifying more extended phrase similarities. While the Rouge-L score, measuring the longest common subsequence, demonstrates a slightly higher value of 8.4, suggesting improved content overlap, the overall F1 measure emphasizes weaknesses in the Seq2Seq model's ability to generate concise and faithful summaries.

Turning our attention to the performance of the T5 model, it demonstrates a significantly more robust performance when compared to the Seq2Seq model. The Rouge-1 score of 27.2 is nearly triple that of the Seq2Seq model's Rouge-1 score, showcasing a substantial improvement in capturing unigram overlap with the reference summaries. In terms of Rouge-2, the T5 model achieves a score of 13.46, outperforming the GPT-2 model in this specific metric. The Rouge-L score of 22.7 further highlights the T5 model's strong capability in extracting semantic content from the source article and generating summaries with notable fluency. The overall performance of the T5 model is notably close to the GPT-2 model, showing its promising potential in the task of text summarization.

The challenges encountered by the Seq2Seq model can be attributed to several factors and one of the primary issues that need to be refined is long input articles. In the test, the input sequence has been expended to 700 tokens, a length necessitated by the desire to retain the majority of information from the articles. However, 700 tokens'length creates a significant challenge to the encoder-decoder architecture, particularly with LSTMs, which possess a limited context window. As the length of the story increases, LSTMs face constraints in referencing words generated earlier in the sequence, impacting their ability to effectively capture and retain information over extended narratives. In contrast, the T5 model, leveraging a transformer architecture with self-attention mechanisms, employs wave frequencies to encode positional information. This unique design enables the T5 model to navigate long input sequences more adeptly compared to the Seq2Seq model (Wydmanski, 2022).

Furthermore, the gradient problem exacerbates the Seq2Seq model's struggle with lengthy sequences. LSTMs, processing sequences sequentially, encounter challenges during backpropagation, where gradients must traverse each time step. As sequence length grows, the multiplication of gradient matrices at each time step can lead to vanishing gradients. This phenomenon results in diminutive gradient updates during training, impeding effective parameter learning and hindering the model's overall training efficacy. In contrast, the T5 model's transformer architecture, built on self-attention mechanisms, allows for more efficient parallelization, mitigating the vanishing gradient issue and contributing to improved model training.

5 Further approaches

The Seq2Seq model demands critical enhancements, with the foremost imperative lying in the

refinement of its attention mechanism. The utilization of Bahdanau attention is not reliable when confronted with excessively long input sequences. However, self-attention mechanisms, which employ wave frequencies to encode positional information, may potentially solve this problem. Self-attention offers the model a more effective means to capture contextual dependencies particularly in mitigating challenges posed by lengthy inputs.

Furthermore, exploring alterations to the model's layer construction, such as the introduction of bidirectional layers, holds potential for notable improvements. Another approach for refinement lies in changing decoding strategies from greedy search algorithms. The use of greedy search algorithms in decoding tends to be sub-optimal solutions, focusing solely on immediate optimal choices. Shifting towards beam search algorithms can substantially improve the model's performance with its capacity to consider a diverse array of candidate sequences which creates a more comprehensive solution. This strategic shift towards a global optimization perspective is instrumental in advancing the Seq2Seq model's proficiency.

The current performance of the T5 model is satisfactory, but there is room for improvement. One potential strategy is to fine-tune the larger T5 Base model instead of the T5 Small model. T5 Base, being a larger model with more layers and parameters, could offer enhanced performance compared to T5 Small. Despite the increased demand for computational power and memory during training and inference, the expectation is that T5 Base may outperform T5 Small after fine-tuning.

It's worth noting that the validation loss of the T5 Small model shows limited improvement even after 1000 steps, suggesting that extending the fine-tuning duration for T5 Small may not yield substantial benefits. Therefore, exploring the larger T5 Base model for fine-tuning appears to be a reasonable approach to potentially achieve better results.

6 conclusion

In conclusion, the rapid expansion of written information in the 21st century necessitates innovative approaches to handle and uncover insights from vast data sets. The limitations of traditional processing and storage methods highlight the significance of Artificial Intelligence (AI) in managing this overwhelming amount of data. Text summarization becomes a crucial tool for condensing lengthy documents into concise representations, addressing the issue of excessive information. This research paper utilizes the CNN-DailyMail News data set strategically and employs Azure Databricks for distributed data preprocessing, effectively overcoming challenges associated with enormous data sets. By integrating Seq2Seq models with distributed computation using Tensor Processing Units (TPUs) on Colab platform, a concerted effort is made to tackle text summarization at scale. Furthermore, alternative techniques such as evaluating Google's Text-To-Text Transfer Transformer model (T5), are employed to navigate through complexities involved in this task. Through Rouge scores measurement that assesses model performance, it is evident that T5 outperforms Seq2Seq model particularly when considering unigram similarity, bigram overlap, and longest common subsequence measures

The T5 model emerges as a promising contender in the realm of large-scale text summarization when compared to other models like Scrambled Code + Broken and GPT-2. Despite its challenges with extensive input sequences and the gradient problem, the Seq2Seq model offers valuable opportunities for improvement. These include enhancing attention mechanisms, exploring modifications in layer construction, and experimenting with decoding strategies. By employing distributed computing and advanced AI models, this study presents a comprehensive approach to large-scale text summarization. The results emphasize that the T5 model exhibits superior performance over traditional Seq2Seq models, opening avenues for further advancements in this domain.

7 Reference

Chavan, A. (2020) Twitter data storage and processing, Medium. Available at: https://ankush-chavan.medium.com/twitter-data-storage-and-processing-dd13fd0fdb30 .

Sutskever, I., Vinyals, O. and Le, Q.V. (2014) Sequence to sequence learning with Neural Networks, arXiv.org. Available at: https://arxiv.org/abs/1409.3215 .

META AI Papers with code - CNN / daily mail benchmark (document summarization) (no date) The latest in Machine Learning. Available at: https://paperswithcode.com/sota/document-summarization-on-cnn-daily-mail .

Wydmanski, W. (2022) What's the differ-

ence between self-attention and attention in transformer architecture?, Medium. Available at: https://medium.com/mlearning-ai/whats-the-difference-between-self-attention-and-attention-in-transformer-architecture-3780404382f3 .

Weber, B. (2019) 3 methods for parallelization in Spark, Medium. Available at: https://towardsdatascience.com/3-methods-for-parallelization-in-spark-6a1a4333b473 .