NLP Assignment-2 Report

By- Ishaan Sinha
Roll number: 21CS30064

Aim and Overview of the Assignment:

The aim of this assignment is to develop and evaluate a text summarization model using NLP techniques. The project involved training the RNN (or LSTM) model on the CNN/Daily Mail dataset, which contains articles and their corresponding summaries. After preprocessing the data by cleaning and tokenizing the text, I converted it into sequences suitable for model input.

A sequence-to-sequence model with multiple LSTM layers was developed, incorporating an attention mechanism to enhance summary generation. The model was trained using a portion of the CNN/Daily Mail dataset while monitoring performance on a validation set. For final testing, I utilized a separate Wikipedia summary dataset containing 5000 samples to evaluate the model's effectiveness. The evaluation was conducted using ROUGE scores to analyze the quality of the generated summaries.

Part-1: Data Exploration and Model Building:

Task-1: Data Exploration with CNN/Daily Mail Dataset:

1 a) After inspecting the dataset, I identified three columns: 'article', 'highlights', and 'id'. The 'article' column contains the full text, while the 'highlights' column provides the corresponding summaries for each article. This structure facilitated the development of the text summarization model by using articles for training and highlights for validation and evaluation.

1 c) Before exploring the dataset, I performed basic text cleaning to enhance the quality of the analysis. This involved converting the text to lowercase, removing special characters, and tokenizing the text to ensure only words were retained. The cleaning process was applied to both the headlines (summaries) and the articles. By tokenizing and cleaning the text beforehand, I could obtain a clearer understanding of word frequency and sentence lengths, which are crucial for evaluating the model's performance in text summarization. The cleaned tokens were then flattened for subsequent frequency analysis.

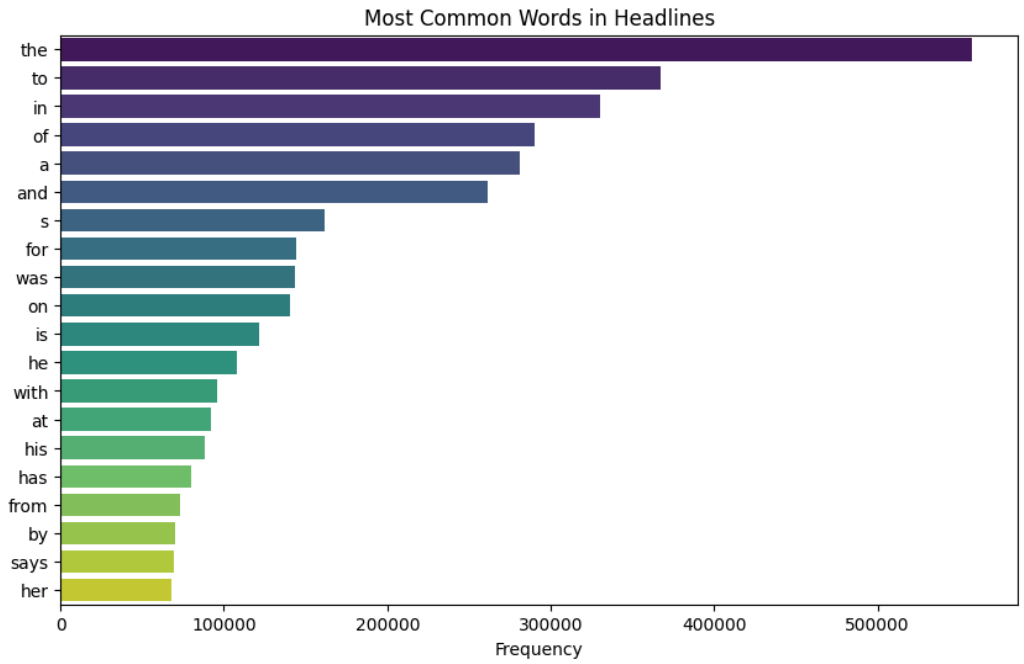
1 b) After exploration I found:

The most frequent word in both articles(text) and highlights(summary) is : the

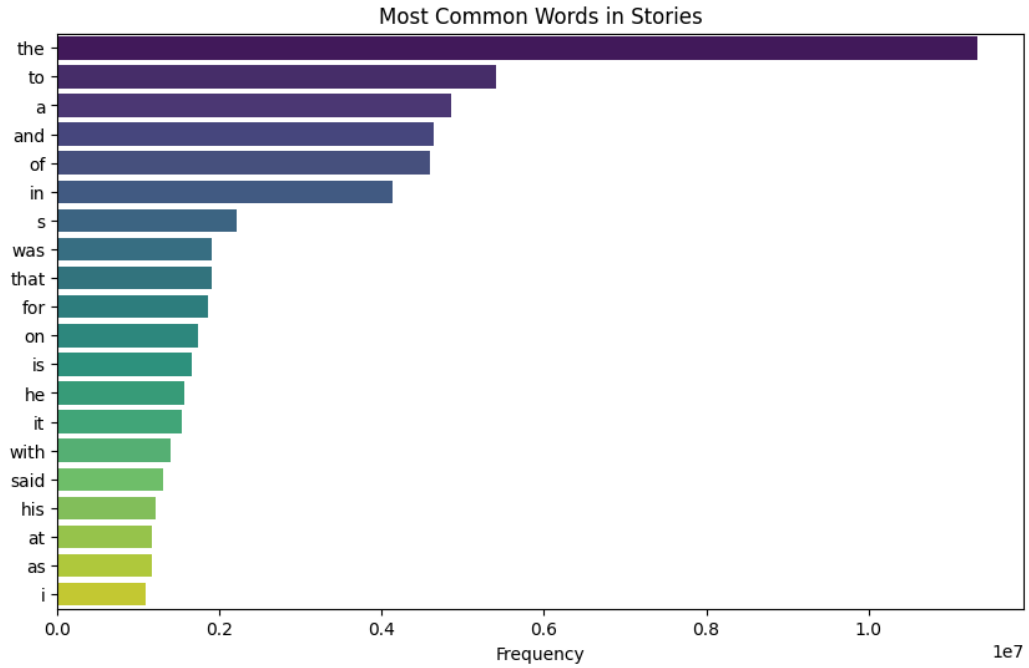
The most frequent article length is : 500-700

The most frequent highlights length is : 10-150

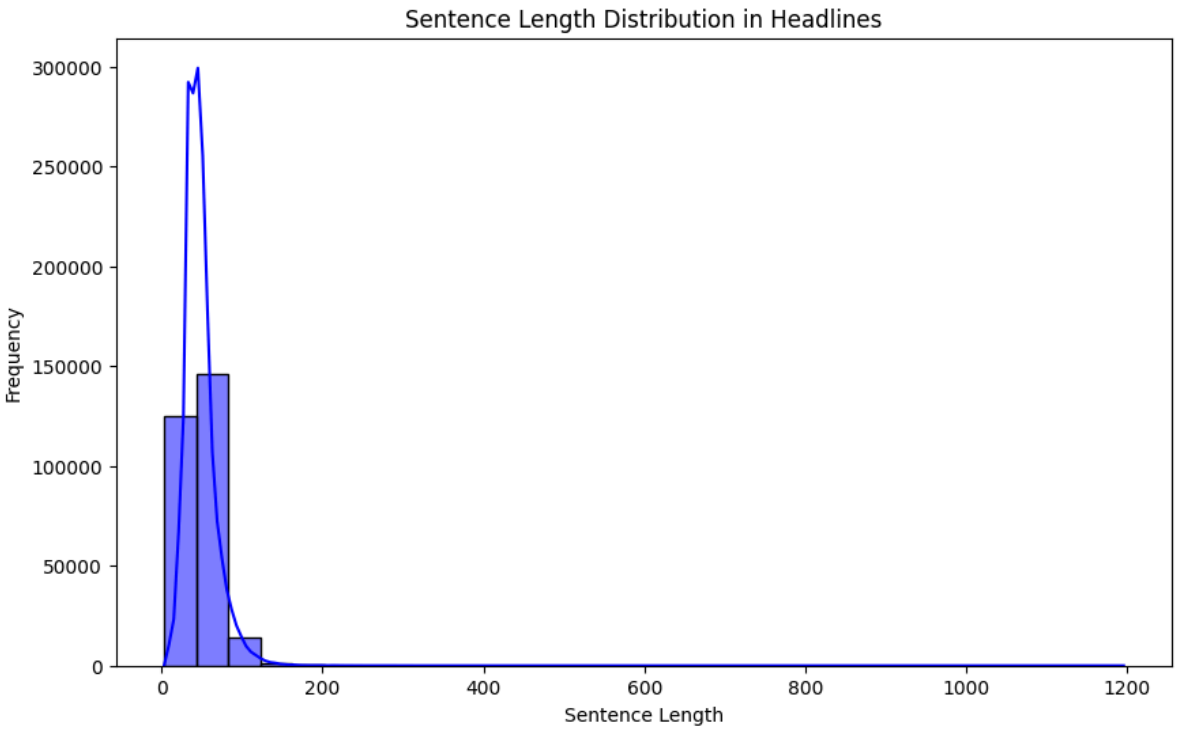
1 d) Visualization of all the findings:

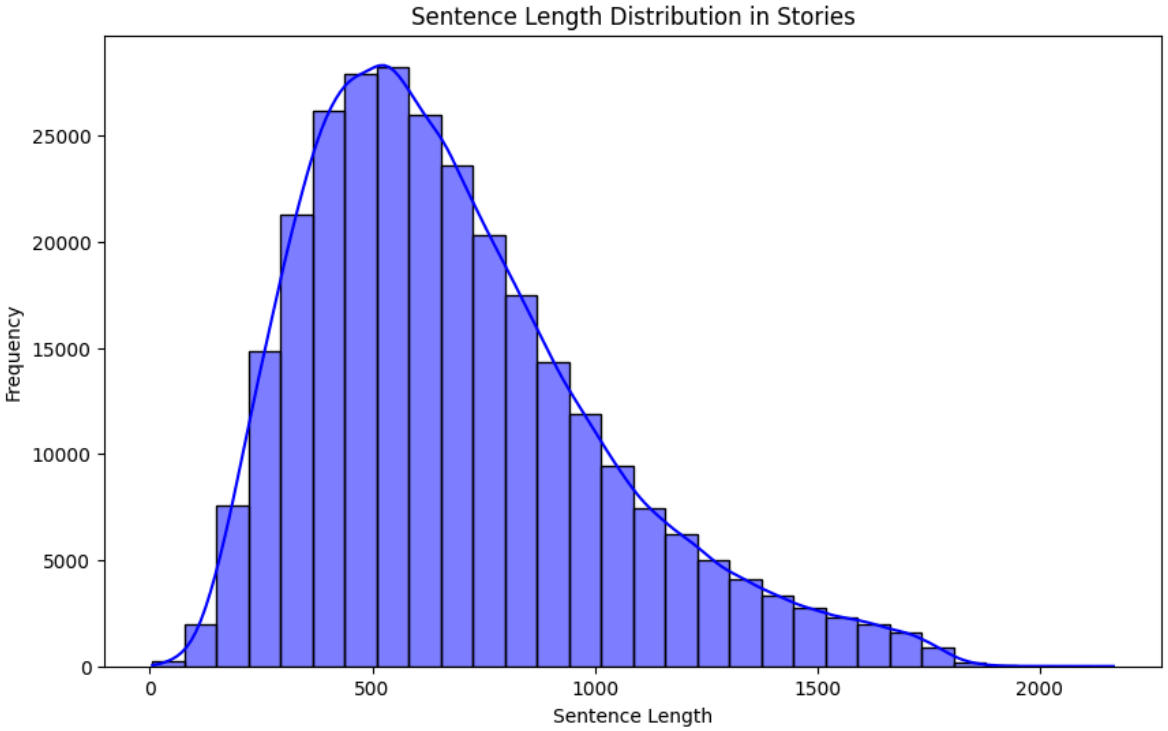


Note: Headlines is basically summary



Note: Stories is basically articles


Sentence Length Distribution in Headlines
Frequency
Sentence Length
0
50000
100000
150000
200000
250000
300000
0
200
400
600
800
1000
1200


Sentence Length Distribution in Stories
Frequency
Sentence Length
0
5000
10000
15000
20000
25000
0
500
1000
1500
2000

Task 2: Build and Train Seq2Seq Model for Text Summarization:

To build the Seq2Seq model for text summarization, I experimented with two approaches. Initially, I developed a standard Seq2Seq model using LSTM layers without incorporating an attention mechanism. However, the evaluation scores were significantly lower than expected. Based on feedback received via email, I decided to enhance the model by integrating an attention layer mechanism along with pre-trained word embeddings. This adjustment aimed to improve the model's ability to focus on relevant parts of the input sequence during decoding, thereby enhancing the summarization performance, however unfortunately the results didn't improve significantly but here is the detailed explanation of what I did in the 2 approaches.

Approach-1 : Without using Attention Layer Mechanism

a) Preprocessing:-

1. Dataset Loading:

- The full CNN/Daily Mail dataset is loaded using the Hugging Face `datasets` library.
- The dataset is reduced to a 10% sample, ensuring that a representative subset is selected for training while reducing the computational load.

2. Data Cleaning:

- A DataFrame is created from the dataset, containing 'text' (articles) and 'summary' (highlights) columns.
- The dataset is filtered to remove duplicate entries and rows with missing values.
- A mapping dictionary is used to expand English contractions (e.g., "he's" to "he is").
- HTML tags, possessive forms, and non-alphabetic characters are removed from the text.
- Optional stopword removal and token filtering are applied to clean the data.

3. Length Analysis:

- The number of words in each text and summary is calculated to determine suitable maximum lengths.
- Articles longer than 700 words and summaries longer than 100 words are excluded, setting these as the maximum lengths.

4. Tagging for Sequence Decoding:

- "START" (`sostok`) and "END" (`eostok`) tokens are added to summaries to aid in the decoding process during training.

5. Data Splitting:

- The dataset is split into training and testing sets (80% training, 20% testing) with shuffling for randomization.

6. Tokenization and Vocabulary Building:

- Tokenizers for both articles and summaries are created using the `Tokenizer` class from Keras.

- A threshold is set for considering rare words (those appearing fewer than a specified number of times), with the code calculating the percentage of rare words and their coverage in the dataset.
- The vocabulary size is adjusted based on the frequency of words.

7. Sequence Conversion and Padding:
- The text and summary sequences are converted to integer sequences based on the tokenizers.
- The sequences are padded to the maximum lengths defined earlier, ensuring uniform input sizes.

8. Vocabulary Size Calculation:
- The final vocabulary size for both articles and summaries is determined.

b) Building the seq2seq Architecture:-

• Encoder: A unidirectional LSTM with an embedding layer was used to capture the meaning of the input text. The encoder has a total of **5 LSTM layers**.
• Decoder: A unidirectional LSTM decoder was employed to help the model focus on relevant sections of the input when generating the summary. The decoder part has only **1 layer of LSTM**.
• TimeDistributed Layer: The overall model architecture also has a TimeDistributed layer to provide the final output as a sequence of tokens.

Model: "functional_1"

Layer (type)	Output Shape	Param #	Connected to
input_layer (InputLayer)	(None, 700)	0	-
embedding (Embedding)	(None, 700, 256)	14,496,256	input_layer[0][0]
lstm (LSTM)	[(None, 700, 256), (None, 256), (None, 256)]	525,312	embedding[0][0]
lstm_1 (LSTM)	[(None, 700, 256), (None, 256), (None, 256)]	525,312	lstm[0][0]
lstm_2 (LSTM)	[(None, 700, 256), (None, 256), (None, 256)]	525,312	lstm_1[0][0]
input_layer_1 (InputLayer)	(None, None)	0	-
lstm_3 (LSTM)	[(None, 700, 256), (None, 256), (None, 256)]	525,312	lstm_2[0][0]
embedding_1 (Embedding)	(None, None, 256)	3,159,552	input_layer_1[0]…
lstm_4 (LSTM)	[(None, 700, 256), (None, 256), (None, 256)]	525,312	lstm_3[0][0]
lstm_5 (LSTM)	[(None, None, 256), (None, 256), (None, 256)]	525,312	embedding_1[0][0… lstm_4[0][1], lstm_4[0][2]
time_distributed (TimeDistributed)	(None, None, 12342)	3,171,894	lstm_5[0][0]

Total params: 23,979,574 (91.47 MB)

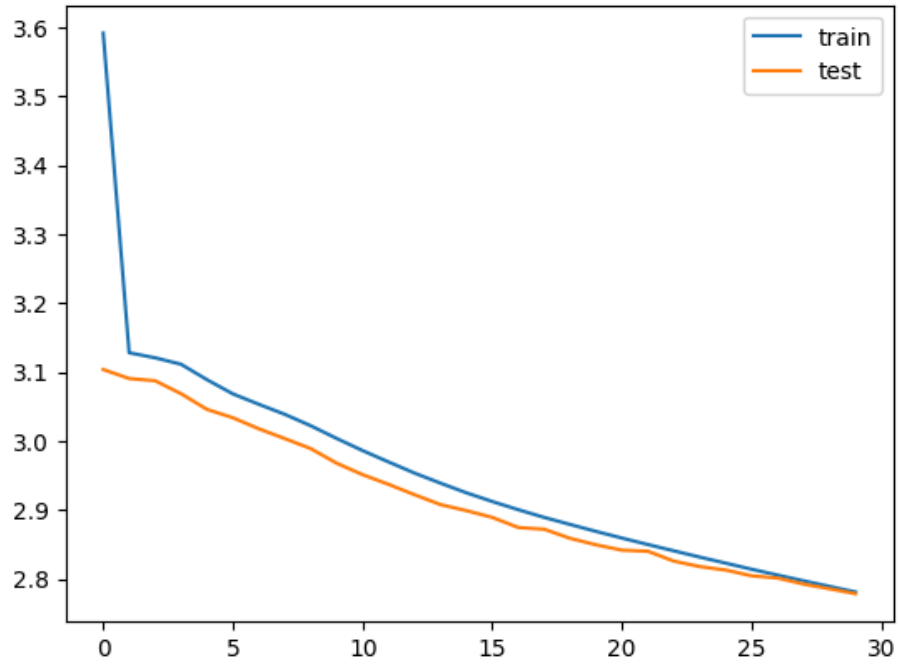
Trainable params: 23,979,574 (91.47 MB)

Non-trainable params: 0 (0.00 B)

• Hyperparameters:
    - o Embedding size: 256
    - o Hidden units (LSTM): 256
    - o Batch size: 128
    - o Optimizer: rmsprop
    - o Loss: sparse_categorical_crossentropy

c) Training and Evaluation:-

• Training Process: The model was trained on 30 epochs with early stopping based on the validation loss. Also, the training data was split 80%-20% into training and validation sets.



• Evaluation Metric: ROUGE-2 and ROUGE-L scores were used to measure the quality of generated summaries.

• Results on Test Data (CNN/Daily Mail):
    - o ROUGE-2 Score: 0.0202
    - o ROUGE-L Score: 0.1113

Approach-2 Using Attention Layer and Pre-trained Embeddings

a) Preprocessing:-

In approach 2, the preprocessing remains largely similar to approach 1, with the main difference being the use of the pre-trained Google News word2vec model. The model was loaded using the `gensim` library from the `GoogleNews-vectors-negative300.bin` file, providing 300-dimensional word embeddings to better capture semantic relationships in the text.

Imp Note: For running the file **nlp-assgn2-part1-task2** please download the .bin file and add it as an input, the link for downloading the file and how to add is mentioned at the end of the report in detail.

b) Building the seq2seq architecture:-

• Encoder: A unidirectional LSTM with an embedding layer was used to capture the meaning of the input text. The encoder has a total of **3 LSTM layers**.

• Decoder: A unidirectional LSTM decoder was employed to help the model focus on relevant sections of the input when generating the summary. The decoder part has only **1 layer of LSTM**.

• Attention Layer: The addition of the attention mechanism (Bahdanau attention) allows the model to dynamically focus on different parts of the input sequence during each decoding step, enhancing the model's ability to generate more contextually relevant summaries.

• TimeDistributed Layer: The overall model architecture also has a TimeDistributed layer to provide the final output as a sequence of tokens.

```
Model: "functional_1"
```

Layer (type)	Output Shape	Param #	Connected to
input_layer (InputLayer)	(None, 1000)	0	-
embedding (Embedding)	(None, 1000, 300)	38,653,800	input_layer[0][0]
lstm (LSTM)	[(None, 1000, 256), (None, 256), (None, 256)]	570,368	embedding[0][0]
input_layer_1 (InputLayer)	(None, None)	0	-
lstm_1 (LSTM)	[(None, 1000, 256), (None, 256), (None, 256)]	525,312	lstm[0][0]
embedding_1 (Embedding)	(None, None, 300)	14,100,000	input_layer_1[0]…
lstm_2 (LSTM)	[(None, 1000, 256), (None, 256), (None, 256)]	525,312	lstm_1[0][0]
lstm_3 (LSTM)	[(None, None, 256), (None, 256), (None, 256)]	570,368	embedding_1[0][0… lstm_2[0][1], lstm_2[0][2]
attention_layer (AttentionLayer)	[(None, None, 256), (None, None, 1000)]	131,328	lstm_2[0][0], lstm_3[0][0]
concat_layer (Concatenate)	(None, None, 512)	0	lstm_3[0][0], attention_layer[…
time_distributed (TimeDistributed)	(None, None, 12851)	6,592,563	concat_layer[0][…

```
Total params: 61,669,051 (235.25 MB)
```

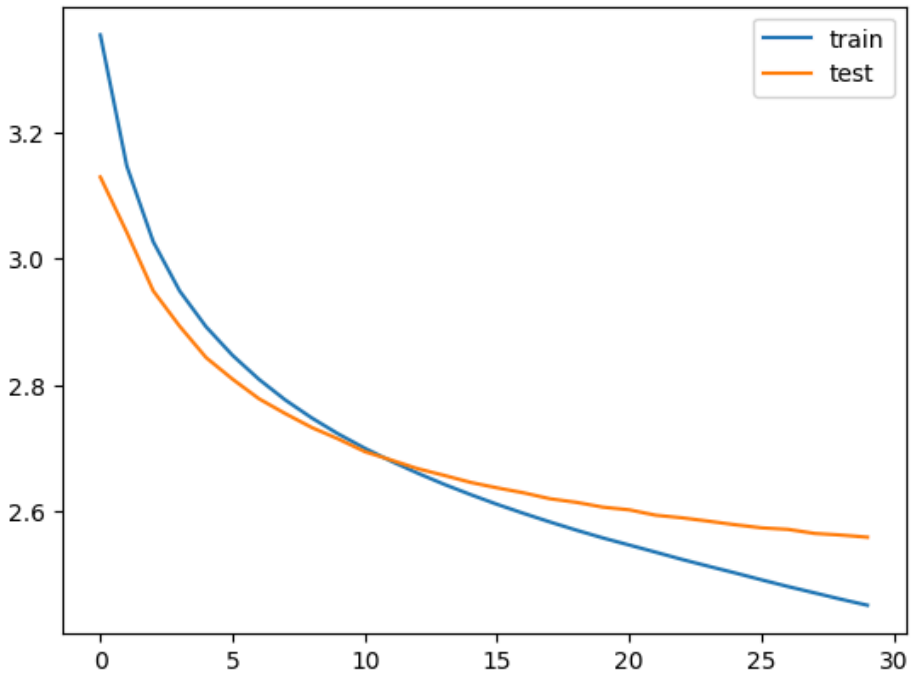
```
Trainable params: 8,783,923 (33.51 MB)
```

```
Non-trainable params: 52,885,128 (201.74 MB)
```

• Hyperparameters:
    - o Embedding size: 300
    - o Hidden units (LSTM): 256
    - o Batch size: 64
    - o Optimizer: rmsprop
    - o Loss: sparse_categorical_crossentropy

c) Training and Evaluation:-

• Training Process: The model was trained on 30 epochs with early stopping based on the validation loss. Also, the training data was split 80%-20% into training and validation sets.



• Evaluation Metric: ROUGE-2 and ROUGE-L scores were used to measure the quality of generated summaries.
• Results on Test Data (CNN/Daily Mail):
    - o ROUGE-2 Score: 0.0116
    - o ROUGE-L Score: 0.1155

Final Observation

The experiments conducted in Task 2 revealed significant insights into the model's performance on text summarization. In Approach 1, which did not incorporate an attention layer, the model achieved ROUGE scores of ROUGE-2 (0.0202) and ROUGE-L (0.1113). While these scores indicate a basic level of summarization capability, the primary limitation of this approach was the generation of identical and irrelevant summaries for each input test sentence. This outcome

highlighted a crucial issue: the model struggled to generate meaningful and diverse summaries, which significantly impacted the quality of the output.

To address this problem, enhancements were made in Approach 2 by incorporating word2vec embeddings and an attention layer. The attention mechanism was added to improve the model's ability to focus on relevant sections of the input during summary generation, potentially leading to more accurate and contextually appropriate summaries. Despite these changes, the ROUGE scores did not show significant improvement. Although there was a slight increase in ROUGE-L from 0.1113 to 0.1155, ROUGE-2 decreased from 0.0202 to 0.0116. While these metrics did not indicate substantial progress, it is important to note that ROUGE scores are not always the most reliable indicators of summarization quality, especially in terms of relevance and diversity.

The more pressing issue, however, was that the problem of generating the same output for each test sentence persisted. After careful analysis, it was observed that even after training on 10% of the CNN/Daily Mail dataset for 30 epochs, the model's loss function did not converge adequately to produce varied and meaningful summaries. The training process was constrained by resource limitations, with each epoch taking approximately 500-1000 seconds to complete. Given the large number of trainable parameters and the substantial size of the dataset, increasing the number of epochs was not a feasible solution.

To further evaluate the model's capabilities, a smaller dataset was used to alleviate resource constraints. With this new dataset, the model's behavior improved slightly, as it started generating different outputs for each test sentence, with some outputs being more relevant to the input context. Although there was a decrease in the ROUGE scores when trained on the smaller dataset, the variation in the output indicated that the model was at least making some progress in learning meaningful patterns.

To validate the model's generalization ability, experiments were conducted using the Wikipedia summary as input, after training on the new, smaller dataset. The details of the new dataset, the updated ROUGE scores, and the results from testing with the Wikipedia summaries are further elaborated in the explanation of Part 2 below.

In conclusion, the experiments highlighted the challenges in training complex models for text summarization with limited computational resources. The observations suggest that while enhancements like word2vec embeddings and attention layers can contribute to improvements, addressing issues such as model overfitting, training time, and dataset size requires careful tuning and potentially different approaches to achieve optimal results.

Part 2: Applying the Model to New Data:

In the second part of the task, the goal was to evaluate the trained model on a new dataset. Initially, the plan was to use the same model trained on the CNN/Daily Mail dataset to test its performance on the new data. However, given the poor results observed during the training on

the CNN/Daily Mail dataset—especially the major issue of generating identical summaries for all test sentences—it became clear that this approach would not be effective.

To address these challenges, a different strategy was employed: changing the training dataset. The Amazon Fine Food Review dataset was selected as a replacement for the CNN/Daily Mail dataset. This dataset was significantly smaller, making it more manageable given the available computational resources. At the same time, it was diverse and sufficient enough to allow the model to learn meaningful summarization patterns.

After retraining the model using the Amazon Fine Food Review dataset, the ROUGE-L score decreased from 0.1155 (achieved with the CNN/Daily Mail dataset) to 0.08. Although this score was lower, a noticeable improvement in the model's performance was observed: the summaries generated for each test sentence were different, rather than being identical across all test cases. Furthermore, the generated summaries showed some degree of relevance to the input text, albeit limited. This outcome was still a positive step forward compared to the previous case where the outputs were not only identical but also completely irrelevant.

Overall, switching to the Amazon Fine Food Review dataset proved beneficial in terms of producing more diverse and contextually appropriate summaries, even if the relevance was not optimal. This change highlighted the importance of choosing a suitable dataset, especially for training deep learning models with limited computational resources.

Imp Note: For running the file **nlp-assignment2-21cs30064-part2** please download the review.csv file and add it as an input, the link for downloading the file and how to add is mentioned at the end of the report in detail.

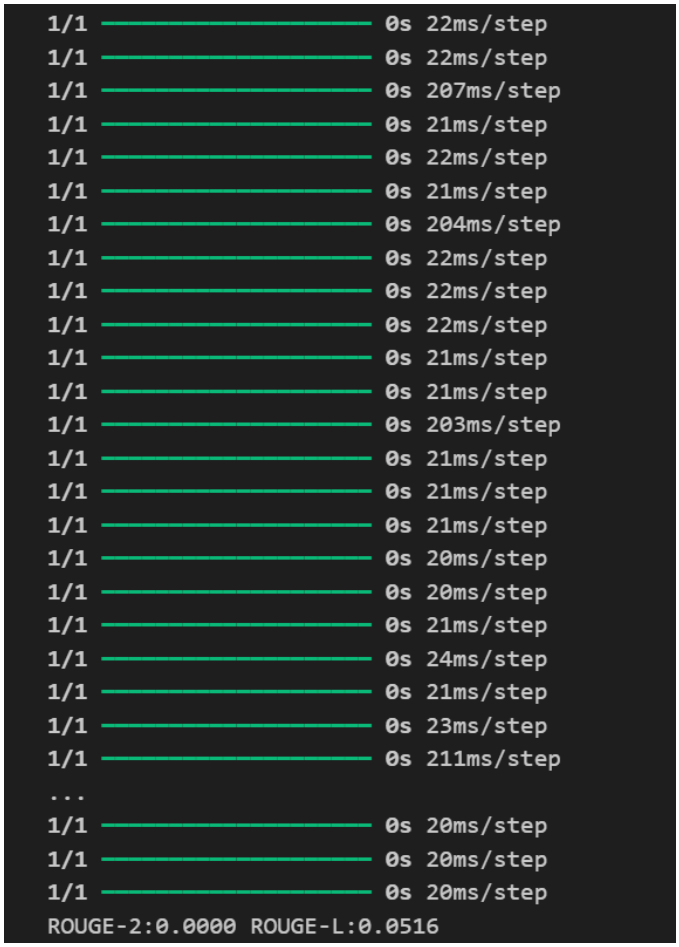
When working with the Wikipedia Summary dataset, loading it directly using the `load_dataset` function proved to be problematic. The available memory on Kaggle was consistently running out, causing the kernel to restart repeatedly. To resolve this issue, the dataset was downloaded separately as a `WikipediaSummary.csv` file and added as an input to the Kaggle environment. This approach allowed for smoother data handling without exhausting the memory.

Once the dataset was available as a CSV file, the text and corresponding summaries were extracted for further preprocessing and text cleaning. The preprocessing steps followed were identical to those used for training the model on previous datasets, ensuring consistency in data preparation.

For training the model on the Amazon Fine Food Review dataset, the same methodology (preprocessing, training and evaluation metrics) from Task 2 was applied, with only the dataset being different. Some slight adjustments were made to ensure compatibility with the new dataset, but the overall approach remained the same, including the architecture and training procedures. This consistency helped maintain a standardized process while adapting to the new dataset.

Imp Note: For running the file **nlp-assignment2-21cs30064-part2** please download the WikipediaSummary.csv file as well along with review.csv mentioned above and add it as an input, the link for downloading the file and how to add is mentioned at the end of the report in detail.

Model Evaluation Results (Wikipedia Data):

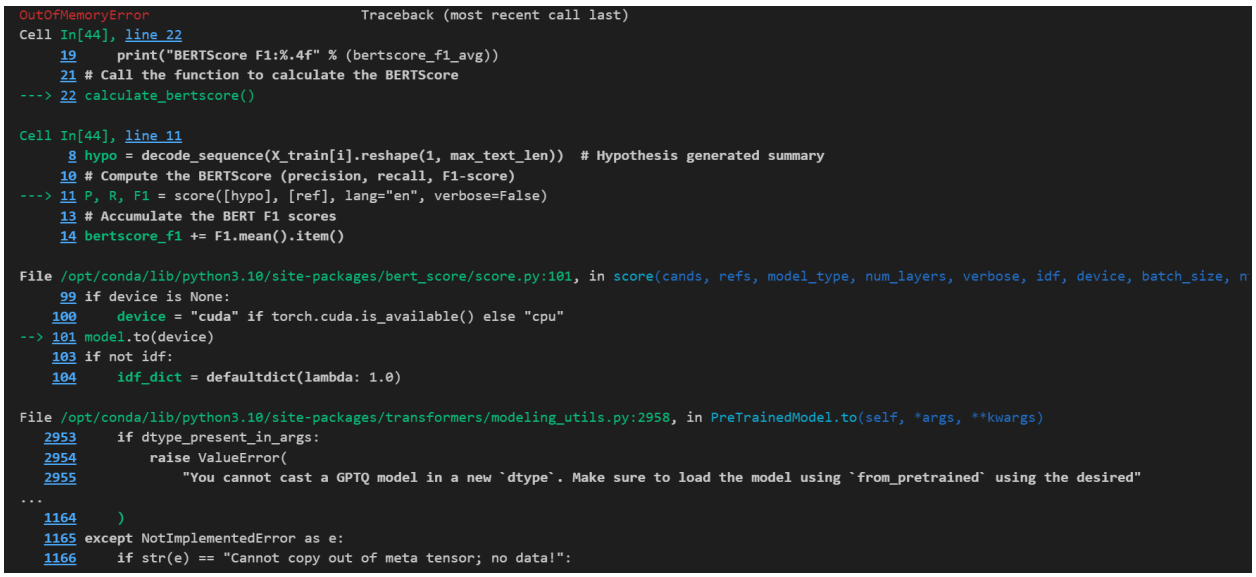


Imp Note: These results were slightly lower than the Amazon Fine Food Review dataset, likely due to the differences in writing style between food reviews and Wikipedia entries, which are often more formal and densely informative.

Drawbacks in the Assignment:

1. A significant limitation of this assignment was the inability to implement BERT/BART score as the evaluation metric, despite recommendations to use these methods over ROUGE score due to their superior accuracy. The large size of the BERT-score and BART-score models resulted in CUDA out of memory errors during attempts to compute these scores, even after several optimization efforts. Consequently, I was constrained to continue using ROUGE scores for evaluation, despite its limitations in accurately

assessing the quality of text summarization. A screenshot documenting the memory error encountered during this process is also provided.



```
OutOfMemoryError                          Traceback (most recent call last)
Cell In[44], line 22
     19     print("BERTScore F1:%.4f" % (bertscore_f1_avg))
     21 # Call the function to calculate the BERTScore
---> 22 calculate_bertscore()

Cell In[44], line 11
      8 hypo = decode_sequence(X_train[i].reshape(1, max_text_len))  # Hypothesis generated summary
     10 # Compute the BERTScore (precision, recall, F1-score)
---> 11 P, R, F1 = score([hypo], [ref], lang="en", verbose=False)
     13 # Accumulate the BERT F1 scores
     14 bertscore_f1 += F1.mean().item()

File /opt/conda/lib/python3.10/site-packages/bert_score/score.py:101, in score(cands, refs, model_type, num_layers, verbose, idf, device, batch_size, n
     99 if device is None:
    100     device = "cuda" if torch.cuda.is_available() else "cpu"
--> 101 model.to(device)
    103 if not idf:
    104     idf_dict = defaultdict(lambda: 1.0)

File /opt/conda/lib/python3.10/site-packages/transformers/modeling_utils.py:2958, in PreTrainedModel.to(self, *args, **kwargs)
   2953     if dtype_present_in_args:
   2954         raise ValueError(
   2955             "You cannot cast a GPTQ model in a new `dtype`. Make sure to load the model using `from_pretrained` using the desired"
...
   1164     )
   1165 except NotImplementedError as e:
   1166     if str(e) == "Cannot copy out of meta tensor; no data!":
```

```
OutOfMemoryError: CUDA out of memory. Tried to allocate 16.00 MiB. GPU 0 has a total capacity of 14.74 GiB of which 14.12 MiB is free.
```

Exact Error Message: **CUDA out of memory. Tried to allocate 16.00 MiB. GPU 0 has a total capacity of 14.74 GiB of which 14.12 MiB is free. Process 3419 has 14.72 GiB memory in use. Of the allocated memory 999.23 MiB is allocated by PyTorch, and 4.77 MiB is reserved by PyTorch but unallocated. If reserved but unallocated memory is large try setting PYTORCH_CUDA_ALLOC_CONF=expandable_segments:True to avoid fragmentation. See documentation for Memory Management** (https://pytorch.org/docs/stable/notes/cuda.html#environment-variables)

2. Another drawback encountered in this assignment was the significant time required for generating summaries and preprocessing the data, especially when dealing with large datasets. In Part 2 of the assignment, I limited the evaluation to 5,000 rows of the Wikipedia summary dataset because processing the full dataset was infeasible given its size. After filtering out text and summaries that exceeded the maximum length constraints (`max_text_len` and `max_summary_len`) used in training the model on the Amazon Fine Food Review dataset, I was left with only 45 rows for evaluation. The Amazon dataset's texts and summaries were relatively short, and keeping the maximum length parameters the same for the Wikipedia summary dataset helped ensure consistency. However, if I had adjusted `max_text_len` to accommodate longer texts from the Wikipedia summary dataset, the generated summaries for inputs exceeding the original length constraints would likely have been poor, with a high frequency of `<unk>` tokens. This would have further reduced the quality of the summaries and negatively impacted the ROUGE scores. Ultimately, I calculated the ROUGE score on only 20

sentences, following the same approach used in Part 1. Evaluating on 10-20 sentences was the most feasible solution given the time constraints, as each iteration took a substantial amount of time.

Conclusion:

In conclusion, the exploration and training tasks conducted throughout this assignment highlighted significant limitations in the Seq2Seq model's ability to generate coherent and meaningful summaries, particularly when applied to the CNN/Daily Mail dataset. The primary challenges included the generation of repetitive and irrelevant summaries, which indicated that the model struggled to capture the essence of the input texts. Despite attempts to enhance the model's performance by incorporating Word2Vec embeddings and an attention layer, the improvements in ROUGE scores were minimal, and the core issue of generating uniform outputs remained unresolved.

The transition to the Amazon Fine Food Review dataset yielded some improvements in generating varied and slightly more relevant summaries, but it was evident that the limitations of the current architecture hindered the achievement of more sophisticated summarization. Additionally, resource constraints, such as GPU memory limitations when attempting to implement BERT or BART scoring, restricted the evaluation process, further complicating the model's assessment.

Moving forward, it is crucial to explore and implement more advanced models, such as BART, T5, or other transformer-based architectures, which are designed to handle complex text generation tasks more effectively. These models possess the potential to achieve improved accuracy and fluency in summarization, addressing the shortcomings observed with the current Seq2Seq approach. Furthermore, optimizing training procedures, including better data management and preprocessing techniques, could enhance performance. Overall, future work should focus on leveraging more sophisticated methodologies to unlock the potential for high-quality summarization in natural language processing tasks.

Few Important Points to be noted regarding running few of the files:

1. Here is the link for Wikipedia Summary: (Name of file: WikipediaSummary.csv)
https://drive.google.com/file/d/17TzqryDMK0AaWdnRGKBIU1h4_28BGFRv/view?usp=sharing
Please download it and add it as an input in **nlp-assignment2-21cs30064-part2.ipynb** file while running on kaggle/colab. I ran it on kaggle by creating a new dataset and then copying the path of this file which is saved on kaggle and then pasting it accordingly in the correct place in the code.

2. Here is the link for Amazon Fine Food Review: (Name of file: Reviews.csv)
https://drive.google.com/file/d/1z2iD1q7uiVRpMwUvNF7SX0VTWVff_H6w/view?usp=sharing
Please download it and add it as an input in **nlp-assignment2-21cs30064-part2.ipynb** file while running on kaggle/colab. I ran it on kaggle by creating a new dataset and then copying the

path of this file which is saved on kaggle and then pasting it accordingly in the correct place in the code.

3. Here is the link for the Pretrained Embeddings: (Name of the file: GoogleNews-vectors-negative300.bin)
https://drive.google.com/file/d/1rwzXPrnhcT3CTJ4_AEaFgPL4qEMGzZo1/view?usp=sharing
In this case please add the .bin file by creating a new model and then similarly copy pasting the path of the .bin file (which is inside the newly created model) at the appropriate place in the code. And this needs to be added in the **nlp-assgn2-part1-task2-withattention.ipynb** file.

4. No need for adding anything as input in **nlp-assginment2-21CS30064-part1-task1.ipynb and nlp-assginment2-21CS30064-part1-task2.ipynb** files.