Text Summarization using Recurrent Neural Networks (RNNs)

Hardik Pravin Soni[1]

[1] *20CS30023, 5th Year, Computer Science and Engineering, IIT Kharagpur* Mail

October 20, 2024

1. Introduction

1.1. Objective of the Assignment

The objective of this assignment is to explore and implement advanced techniques for **text summarization**, which refers to the process of condensing a body of text into a shorter version while preserving its key information and overall meaning. Text summarization plays a crucial role in various applications, including information retrieval, content generation, and enhancing user experience by providing succinct insights from large volumes of text.

In this assignment, we will leverage two prominent datasets: the *CNN/Daily Mail* dataset, which comprises news articles along with their corresponding summaries, and the *Wikipedia Summary* dataset, which offers summaries of Wikipedia articles. The primary aim is to develop and evaluate models capable of generating coherent and contextually relevant summaries from the provided texts.

1.2. Approach

To fulfill the objectives delineated above, we will employ a Sequence-to-Sequence (Seq2Seq) model, a sophisticated neural network architecture specifically designed for tasks where both the input and output are sequences, such as in the context of text summarization. The Seq2Seq model harnesses the power of **Recurrent Neural Networks (RNNs)** and their variants, including **Long Short-Term Memory (LSTM)** networks and **Gated Recurrent Units (GRUs)**. These architectures are adept at encoding input sequences (the full text) into fixed-length context vectors, which can then be decoded into output sequences (the summaries).

Moreover, to enhance the performance of our summarization model, we will incorporate the **Bahdanau Attention** mechanism. This attention mechanism allows the model to focus on different parts of the input sequence dynamically, thereby improving the quality of generated summaries by providing contextually relevant information at each decoding step.

The tasks involved in this assignment encompass the following:

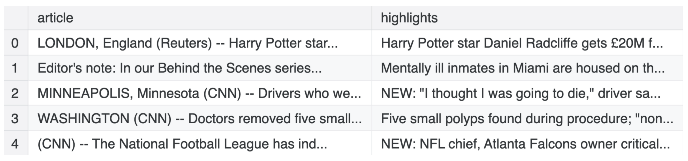
- **Data Exploration**: Conducting a thorough exploratory data analysis (EDA) on the CNN/Daily Mail dataset to gain insights into its structure. This includes inspecting the dataset's features, analyzing word frequency distributions, and visualizing sentence length statistics.
- **Model Building**: Implementing the Seq2Seq architecture, encompassing both the encoder and decoder components. We will also configure hyperparameters to optimize model performance, utilizing the `torch` library for efficient computation and model training.
- **Model Evaluation**: Training the model on the CNN/Daily Mail dataset and rigorously evaluating its performance on the test split using ROUGE scores. These scores will serve as quantitative metrics for assessing the quality of the generated summaries.
- **Testing on New Data**: Applying the trained model to the Wikipedia Summary dataset to evaluate its generalization capabilities and performance on previously unseen data.

2. Data Exploration (CNN/Daily Mail Dataset)

2.1. Dataset Overview

2.1.1. Structure of Data

The CNN/DailyMail Dataset is a large collection of news articles paired with human-written summaries, commonly used for training and evaluating text summarization models. It contains over 287,000 articles from CNN and DailyMail, each accompanied by concise summaries or bullet points highlighting key information. This dataset is valuable for NLP tasks focused on generating summaries from long-form content, such as sequence-to-sequence models.

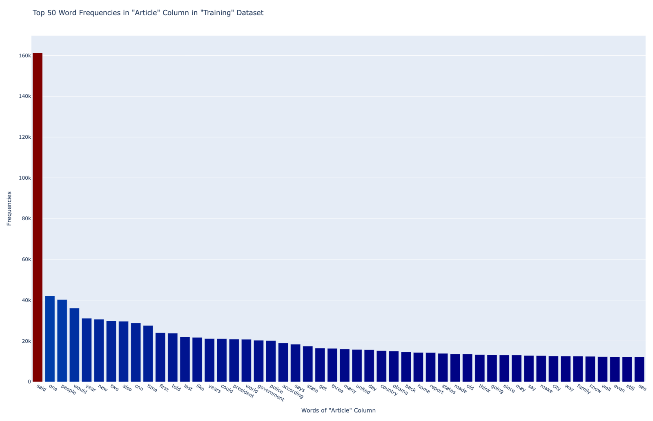


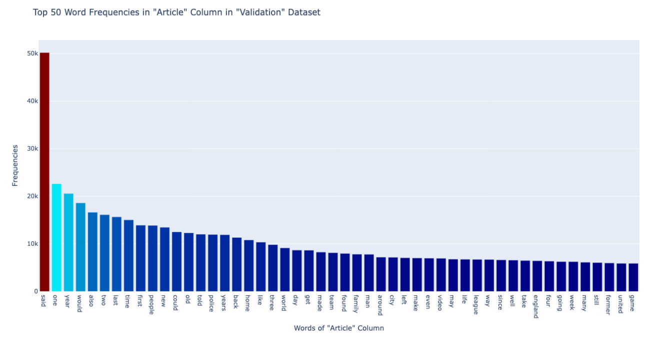
	article	highlights
0	LONDON, England (Reuters) -- Harry Potter star...	Harry Potter star Daniel Radcliffe gets £20M f...
1	Editor's note: In our Behind the Scenes series...	Mentally ill inmates in Miami are housed on th...
2	MINNEAPOLIS, Minnesota (CNN) -- Drivers who we...	NEW: "I thought I was going to die," driver sa...
3	WASHINGTON (CNN) -- Doctors removed five small...	Five small polyps found during procedure; "non...
4	(CNN) -- The National Football League has ind...	NEW: NFL chief, Atlanta Falcons owner critical...

Figure 1. Sample from the **CNN** Dataset

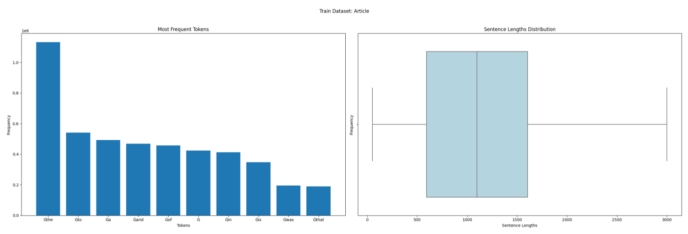
2.1.2. Exploratory Data Analysis (EDA)

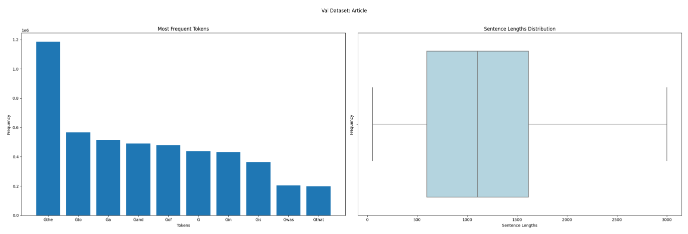
The exploratory data analysis (EDA) of the CNN/DailyMail dataset focuses on the distribution of token frequencies and sentence lengths for both the article and highlights sections of the training dataset. As illustrated in Figure ??, the most frequent tokens in both the article and highlights sections are dominated by common stopwords such as "the", "to", and "and." This suggests that frequent occurrences of these non-informative tokens may skew the model if not handled properly, thereby emphasizing the importance of applying preprocessing techniques, such as stopword removal.



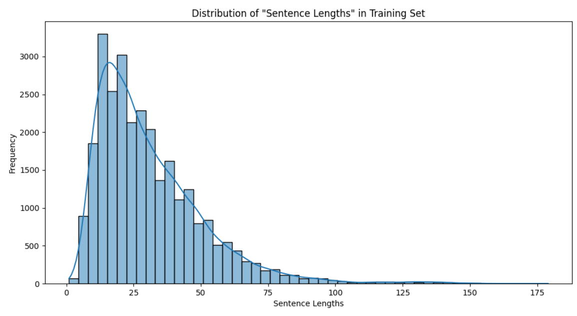


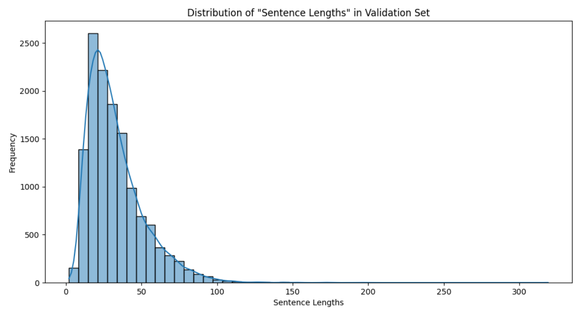
Additionally, the sentence length distributions present distinct patterns for the articles and their associated highlights. For articles, the tokenized sentence lengths follow a long-tailed distribution, with a peak around 500 tokens and some articles exceeding 2,500 tokens. In contrast, the highlights tend to be more concise, with sentence lengths peaking under 100 tokens.





This notable discrepancy in verbosity between articles and summaries underscores the challenge posed by summarization tasks, where long-form content must be effectively condensed into a short, coherent representation.





2.2. Data Cleaning and Vocabulary Creation

The preprocessing pipeline employed for the CNN/Daily Mail dataset is crucial for ensuring the text data is properly formatted and optimized for input into the neural network models. Text data in its raw form typically contains a myriad of inconsistencies such as **contractions**, **special characters**, and **non-standardized capitalization**, which can introduce noise into the training process. Therefore, a series of systematic data cleaning techniques were applied to standardize the text, thereby improving both the quality of the data and the efficacy of subsequent model training.

2.3. Lowercasing and Handling Contractions

Text is converted to **lowercase** to ensure uniformity, treating words like “The” and “the” as the same. **Contractions**, such as "won't" (replaced by "will not") and "can't" (replaced by "cannot"), are expanded using predefined **regular expressions**. This ensures consistency across the dataset, improving both **tokenization** and model comprehension.

2.4. Removal of Special Characters

Non-alphanumeric characters (e.g., punctuation, special symbols) are removed to reduce noise in the text. While this may lead to the loss of certain **semantic cues**, the trade-off helps streamline the data for better tokenization and summarization.

2.5. Tokenization and Vocabulary Building

The cleaned text is tokenized using the **GPT-2 tokenizer**, which also incorporates special tokens like `<pad>`, `<sos>`, and `<eos>`. The tokenizer converts the cleaned text into **subword tokens** while maintaining a **truncation limit** of 3000 tokens to manage sequence lengths efficiently.

In summary, the data cleaning process was methodically structured to transform raw, inconsistent text into a format that is both standardized and optimized for downstream tasks. The techniques applied, from contraction handling to the removal of special characters and detailed tokenization, ensure that the text is ready for effective training and summarization.

3. Seq2Seq Model Architecture with Bahdanau Attention Mechanism

After conducting numerous experiments in TensorFlow with Seq2Seq models, many of which encountered memory limitations, the following model architecture emerged from the initial trials.

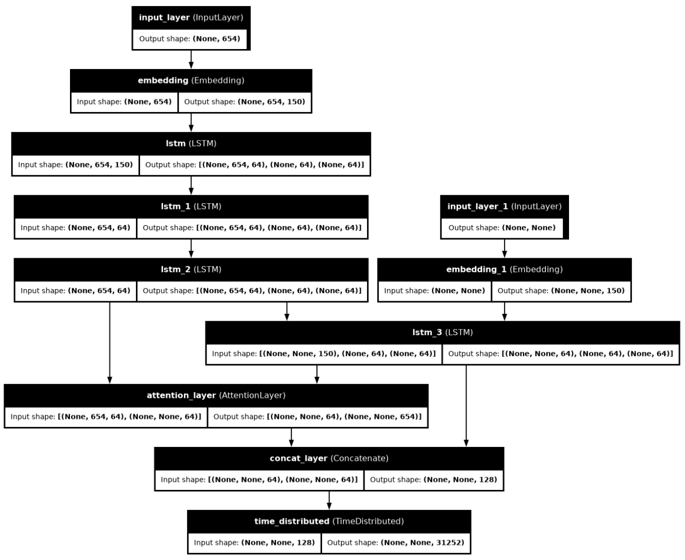

input_layer (InputLayer)
Output shape: (None, 654)
embedding (Embedding)
Input shape: (None, 654) | Output shape: (None, 654, 150)
lstm (LSTM)
Input shape: (None, 654, 150) | Output shape: [(None, 654, 64), (None, 64), (None, 64)]
lstm_1 (LSTM)
Input shape: (None, 654, 64) | Output shape: [(None, 654, 64), (None, 64), (None, 64)]
input_layer_1 (InputLayer)
Output shape: (None, None)
lstm_2 (LSTM)
Input shape: (None, 654, 64) | Output shape: [(None, 654, 64), (None, 64), (None, 64)]
embedding_1 (Embedding)
Input shape: (None, None) | Output shape: (None, None, 150)
lstm_3 (LSTM)
Input shape: [(None, None, 150), (None, 64), (None, 64)] | Output shape: [(None, None, 64), (None, 64), (None, 64)]
attention_layer (AttentionLayer)
Input shape: [(None, 654, 64), (None, None, 64)] | Output shape: [(None, None, 64), (None, None, 654)]
concat_layer (Concatenate)
Input shape: [(None, None, 64), (None, None, 64)] | Output shape: (None, None, 128)
time_distributed (TimeDistributed)
Input shape: (None, None, 128) | Output shape: (None, None, 31252)

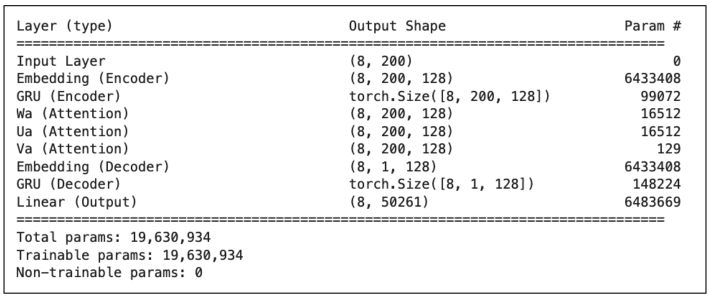
Figure 2. Model Architechture from Inital Experiements

After conducting numerous experiments in ***TensorFlow*** with ***Seq2Seq*** models, many of which encountered memory limitations, the following model architecture emerged from the initial trials. After those trials, I tried implementing the same with 10% of the dataset in ***PyTorch*** to better handle the memory usage in a more flexible way.

Following the data preprocessing steps, the cleaned and tokenized text data is used as input to a ***Seq2Seq*** (Sequence-to-Sequence) model that integrates the **Bahdanau Attention Mechanism**. This architecture is vital for producing high-quality text summaries, as it enables the model to manage sequences of varying lengths and focus on the most relevant parts of the input when generating the output.

3.1. Encoder

The **encoder** transforms the input sequence into a series of hidden states. These hidden states are passed through an **embedding layer**, which maps each token to a high-dimensional vector, followed by a **GRU (Gated Recurrent Unit)** that processes the embedded tokens and generates contextually rich hidden representations for each token in the sequence. The encoder, therefore, outputs a sequence of hidden states, each representing the information encoded up to a specific point in the input text.



```
Layer (type)                            Output Shape                        Param #
================================================================================
Input Layer                             (8, 200)                                  0
Embedding (Encoder)                     (8, 200, 128)                       6433408
GRU (Encoder)                           torch.Size([8, 200, 128])             99072
Wa (Attention)                          (8, 200, 128)                         16512
Ua (Attention)                          (8, 200, 128)                         16512
Va (Attention)                          (8, 200, 128)                           129
Embedding (Decoder)                     (8, 1, 128)                         6433408
GRU (Decoder)                           torch.Size([8, 1, 128])              148224
Linear (Output)                         (8, 50261)                          6483669
================================================================================
Total params: 19,630,934
Trainable params: 19,630,934
Non-trainable params: 0
```

Figure 3. Model Summary

3.2. Bahdanau Attention Mechanism

The **Bahdanau Attention Mechanism**, an essential component of this model, allows the decoder to selectively focus on different parts of the input sequence at each decoding step. Unlike traditional Seq2Seq models that rely on a fixed-length context vector, Bahdanau attention computes a dynamic context vector based on the current hidden state of the decoder and all hidden states of the encoder. The process involves:

- Applying the `Wa` and `Ua` linear transformations to the decoder's hidden state and each encoder hidden state, respectively.
- Summing the transformed states and passing the result through a non-linear activation function (such as `tanh`).
- The `Va` transformation is then applied to compute a scalar score for each encoder hidden state.
- These scores are normalized through a softmax function to produce **attention weights**, which are used to compute the weighted sum of the encoder's hidden states, yielding a **context vector**.

This attention mechanism enables the model to dynamically adjust its focus on different parts of the input sequence, allowing for improved handling of long and complex input articles, which is crucial in text summarization tasks.

3.3. Decoder

The **decoder** generates the output summary one token at a time. At each step, it uses the attention-derived **context vector** in conjunction with the previous token to predict the next token in the sequence. The decoder includes an embedding layer and a GRU, with the concatenation of the token embedding and the context vector being passed to the GRU. This combined representation helps update the decoder's hidden state, ensuring that each prediction is informed both by the encoder's representations and the decoder's own state. A **log-softmax** layer at the end converts the GRU outputs into probability distributions over the vocabulary.

3.4. Teacher Forcing and Inference

During training, the model uses **teacher forcing**, where the actual target token is fed to the decoder at each step, instead of the predicted token. This accelerates learning by ensuring that the model receives accurate information during early stages of training. During inference, however, the model generates tokens sequentially, using its previous predictions to inform the next one, and continues until either an `<eos>` token is produced or the maximum sequence length is reached.

The Seq2Seq model, coupled with the Bahdanau Attention Mechanism, is trained to minimize the negative log-likelihood loss, promoting the generation of highly accurate summaries by ensuring that the model assigns higher probabilities to the correct tokens at each step.

4. Model Training and Evaluation

4.1. Training

The model struggles to effectively learn from the training data across multiple epochs, likely due to an inadequate number of parameters. However, increasing the parameter count results in an **out-of-memory (OOM)** error, preventing further optimization.

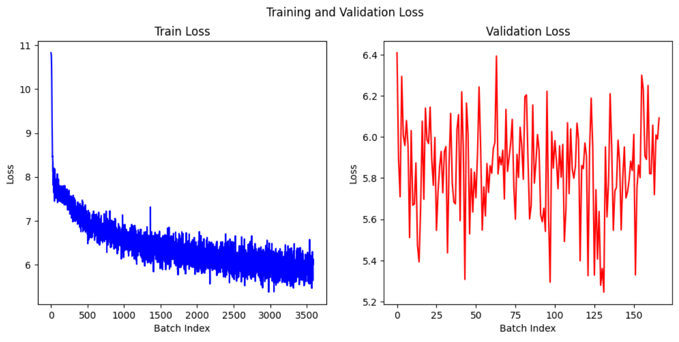


Figure 4. Epoch 1

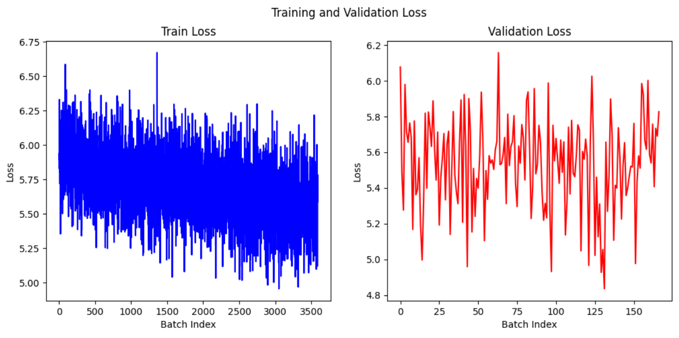


Figure 5. Epoch 2

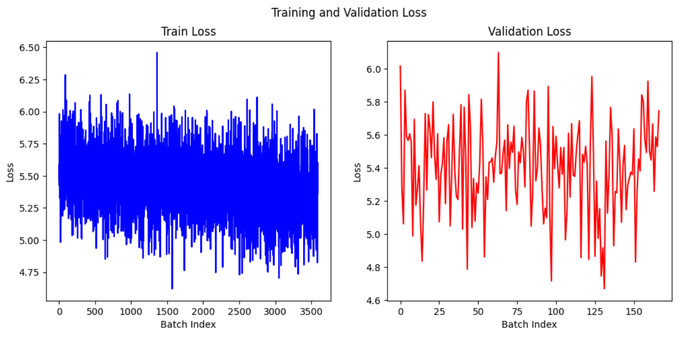


Figure 6. Epoch 3

4.2. Evaluation

4.2.1. CNN-Dailymail Dataset's Test Split Results:

```
1437/1437 [03:41<00:00,  6.49it/s, loss=4.96]
Test ROUGE-L: 0.1243, Test ROUGE-2: 0.0105
```

4.2.2. Wiki Dataset Evaluation Results on first 10, 000 rows:

```
1250/1250 [25:35<00:00,  1.23s/it, loss=6.89]
Wiki ROUGE-L: 0.1272, Wiki ROUGE-2: 0.0233
```

5. Conclusion

Through experimentation with Transformer models on the CNN/DailyMail dataset, it became evident that the **tokenizer** plays a pivotal role in the successful training of Seq2Seq models. Thus, employing robust tokenization algorithms such as **SentencePiece**, **Word2Vec**, or **Byte Pair Encoding (BPE)** is critical for developing effective text summarization models. Furthermore, smaller **RNN-based models with attention mechanisms** demonstrate competitive performance relative to larger Transformer-based models, highlighting their efficiency in certain scenarios. Notably, numerous Kaggle notebooks illustrate the training of **large language models (LLMs)** on this dataset, further reinforcing the suitability of the CNN/DailyMail dataset for **large-scale models** capable of generating **extended sequences**.

■ Resources and Libraries

Libraries & Tools

1. **Hugging Face Datasets**
    - CNN/Daily Mail Dataset: https://huggingface.co/datasets/abisee/cnn_dailymail
    - Wikipedia Summary Dataset: https://huggingface.co/datasets/jordiclive/wikipedia-summary-dataset
2. **PyTorch**
    - PyTorch Library (Official): https://pytorch.org/
    - Seq2Seq Tutorial: https://pytorch.org/tutorials/intermediate/seq2seq_translation_tutorial.html
3. **TensorFlow/Keras**
    - TensorFlow/Keras Library (Official): https://www.tensorflow.org/
    - Keras Sequential Models: https://keras.io/guides/sequential_model/
4. **ROUGE Scoring**
    - ROUGE Package (GitHub): https://github.com/google-research/google-research/tree/master/rouge

Research Papers

- Sutskever et al. (2014), "Sequence to Sequence Learning with Neural Networks" : https://arxiv.org/abs/1409.3215
- Lin (2004), "A Package for Automatic Evaluation of Summaries (ROUGE)" : https://aclanthology.org/W04-1013/