Text Summarization

710075868
Department of Computer Science
University of Exeter
Exeter, UK

***Abstract*—Text summarization involves generating a shorter version of a given text while retaining its vital information and meaning. It offers reliable ways to enhance the effectiveness of information retrieval systems, as it enables users to quickly access the relevant content of a text without having to read the complete information. In this project, we aim to predict summaries based on the original text with the help of two different architectures of the Sequence-to-Sequence model encoder-decoder. In the first case, we want to experiment with three unidirectional LSTM (Long Short-Term Memory) layers on the encoder side and one unidirectional LSTM on the decoder side. In the second case, we will utilize one bidirectional LSTM on the encoder side and a unidirectional one on the decoder side. At last, we will compare them with metrics ROUGE -1 (Recall-Oriented Understudy for Gisting Evaluation - 1), ROUGE-2, and ROUGE-L(Recall-Oriented Understudy for Gisting Evaluation - Longest Common Subsequence).**

***Index Terms*—Text summarization, NLP, Sequence-to-Sequence, word embedding, Tokenization, LSTM, Encoder, Decoder.**

I. Introduction

In recent years, there has been an increase in the amount of textual data available in various domains such as finance, research, legal, and news. For instance, when it comes to reading a long article like in a news or research paper, we require to read each line carefully to ensure that we do not miss any crucial points. The sheer volume of information to be processed and analyzed manually can be time-consuming and costly, leading to the need for automated solutions. Things like that have become more manageable in the age of artificial intelligence thanks to Natural Language Processing (NLP) technology with the introduction of Text summarization in 1980s and 1990s.

II. Background

In the early days, text summarization was done using a set of predefined rules. These rules identified essential parts of the text and ranked them according to their significance. As research in the field of summarization techniques progressed, much development was made. One such technique involves representing the text as a set or vector of terms and giving them weights based on their importance [?].

There are two main techniques used for text summarization: Extractive and Abstractive. Extractive techniques involve selecting key sentences and phrases from the original text to create a summary. This is done without modifying the original sentences themselves. On the other hand, Abstractive techniques involve rewriting the text using different words and phrases to create a summary that conveys the same meaning as the original text [1]. Extractive summarization includes the Graph-Based Approach, Fuzzy Logic Based Approach, Concept-Based Approach, and many other techniques [2]. Models for abstractive summarization fall under a larger deep learning category called sequence-to-sequence models, which map from an input sequence to a target sequence. Although sequence-to-sequence models have been successfully applied to other problems in natural language processing, such as machine translation, there remains much room for improvement for state-of-the-art models in abstractive summarization.

Identify applicable funding agency here. If none, delete this.

III. Related Work

This report focuses only on the abstractive summarization models through the neural network by Rush et al. (2015) and Nallapati et al. (2016) research paper. The model in Rush et al. is an encoder-decoder model where the encoder is a convolutional network, and the decoder is a feedforward neural network language model. They integrate attention into the convolutional encoder and use the trained neural network as a feature of a log-linear model [3]. Nallapati et al. built an encoder-decoder model using LSTMs and augmented their model with hierarchical encoders and hierarchical attention, which learns word and sentence level attention [4].

IV. MY WORKING PLAN

In my project, I use similar architecture from the paper ”Sequence to Sequence Learning with Neural Networks” by Sutskever et al. (2014), as they are used for the machine translation encoder-decoder model [5]. Since I am new to NLP, I wanted to do a text summarization project with simple architecture but with a deep learning model. I found that sequence-to-sequence encoder-decoder will initially suit my project of Text summarization. So, that is why I have selected the Seq2Seq encoder-decoder model as my model architecture and not moved to transformers, Bert, and pre-trained models. I will explore two models with different architectures. In Model 1, I will use three unidirectional LSTM encoders and one unidirectional LSTM decoder. Firstly, the reason for choosing LSTM is because LSTM can help address the problem of vanishing gradients in traditional RNNs and allow the model to remember previous inputs, resulting in more accurate outputs. Secondly, the reason for selecting the 3 LSTM layers stacked on the encoder side is that adding more layers to the encoder

can help the model capture more complex relationships in the input text. In model 2, I will use one bidirectional LSTM on the encoder side and the same number on the decoder side, as it allows the model to consider not only the previous words in the sequence but also the future words, resulting in a better understanding of the overall context. This can improve the quality of the summary generated by the model. Additionally, using the same number of LSTM layers on the decoder side as on the encoder side can help ensure that the decoder has enough capacity to capture the information encoded in the input.

V. OBJECTIVE

1. Text cleaning and preprocessing.
2. Understanding the distribution of the sequences.
3. Filter text and summary by maximum length.
4. Preparing "text and summary" tokenization.
5. Model Building

Model 1: 3 unidirectional LSTM encoder and a bidirectional decoder

Model 2: one bidirectional LSTM encoder and a unidirectional LSTM decoder.

6. Understanding the plot learning curve for both models.
7. Predict the summaries from Model-1 and Model-2.
8. Evaluate based on ROUGE-1, ROUGE-2 and ROUGE-L.

VI. DATASET

In this analysis, we have extracted two datasets, "news summary csv" and "news summary more csv file" from Kaggle [?], where the person named "Kondalarao Vonteru" gathered the summarized news from the Inshorts app and scraped the news articles only from (Hindu, Indian times and Guardian). The period ranges from February to August 2017. Summary csv 1st dataset consists of 4514 data points with six columns (author, date, headlines, read more, text, and ctext). The columns that we have considered are text and headlines. "News summary more csv" 2nd dataset consists of 98401 data points and two columns(features). Description of columns in the file:

- Headlines:- Summary of the news.
- Text:- original document of the news.

VII. METHODS

After importing the datasets, we followed the below steps:

1) Text cleaning and preprocessing: After concatenating two datasets, we obtained a new dataset containing 102915 rows with text and summary columns. As a part of data cleansing, we detected 118 NULL values in the text column and removed those rows. We also removed any other empty rows present in the dataset. Then, We followed the below steps in data pre-processing: -

- Convert all 'content' field data into lowercase letters.
- Remove HTML tags.
- Contraction mapping like $"ain't" \rightarrow "isnot"$
- Remove ('s)
- Remove any text inside the parenthesis ( )
- Eliminate punctuation and special characters.
- Remove stopwords.
- Remove short words.

2) Understanding the distribution of the sequences: After data cleansing and pre-processing, we will analyze the length of the text and the summary to get over the idea about the distribution of the text.

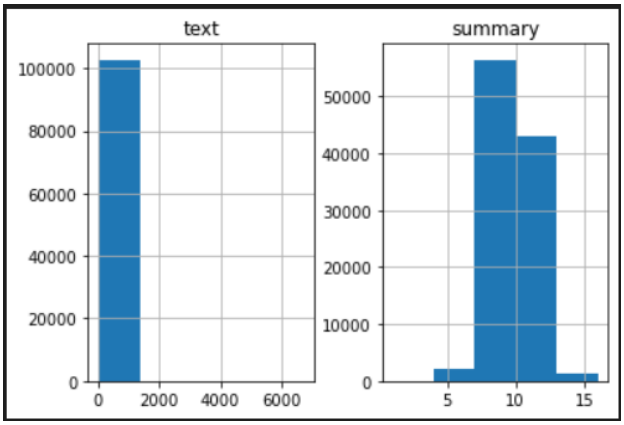

Fig. 1. Frequency Distribution of Text and Summary

From Fig 1, we can see that the majority length of "text" lies below 50 and the majority length of "summary" below 11. So, we are fixing the maximum length of the summary as 50 and the text as 11 because it will help efficiently train the model. If we allow the maximum length to be too high, it will increase the computation time and memory usage required to train the model. By setting the maximum length to a reasonable value, we can reduce the memory required to store the text data and speed up the training process. This is also why we selected a fixed length of summaries and text as 11 and 50, respectively.

3) Preparing tokenization: Tokenization is breaking down a sequence of text into smaller units called tokens, which can be words, phrases, or even individual characters. In this step, we used the Keras tokenizer to prepare the text and summary data for the model. We removed rare words with a frequency threshold of 4 for the text and 6 for the summary to reduce complexity and noise in the data. This improves the model performance by eliminating infrequent words that may not contribute to meaningful patterns. We then converted the text and summary into integer sequences using one-hot encoding and padded them with zeros up to the maximum length. Finally, we obtained the vocabulary sizes for both the text and summary.

VIII. MODEL

The dataset was divided into 90% training and 10% validation data using the train-test split method. I have given an overview of my model's architecture, how it differs from previous working, and why I have selected it in literature in Section 4 (My working plan). In this section, I will define my models to address the text summarization task.

Model 1: The model is a unidirectional encoder-decoder with three stacked LSTM layers in the encoder and a single-layer LSTM on the decoder side. The encoder takes the input text data and processes it through three LSTM layers, each with a latent dimension of 300, initialized with 200-dimensional embeddings. The decoder takes the summary data as input and processes it through an LSTM layer with a latent dimension of 300. The decoder LSTM is initialized with the final hidden and cell states of the encoder LSTM layers. The decoder also uses a dense layer with softmax activation to output the final summary. The model is compiled with the rmsprop optimizer and sparse categorical cross-entropy loss function. During training, the model is fed with the training and validation datasets to monitor its performance on both sets and prevent overfitting. This helps ensure that the model can generalize well to unseen data. A batch size of 128 is used, and the model is trained for a maximum of 10 epochs with early stopping criteria in place if validation loss does not improve after two epochs.

Model 2: The main difference between Model -1 and Model 2 is that Model 2 utilizes a bidirectional LSTM encoder and a unidirectional LSTM decoder, with different latent dimensions of 600(twice the size of the encoder's latent dimension). Model 2 also uses embeddings of size 200 for both the input and output sequences, which are trained alongside the model using the same optimizer and loss function as Model 1. It is also trained with a batch size of 128 on both the training and validation datasets to prevent overfitting and has early stopping criteria in place if validation loss does not improve after two epochs.

My initial plan was to implement the same architecture (like model 1) in model -2 with three bidirectional layers on the encoder side and one unidirectional layer on the decoder side. However, during experimentation, it was challenging for me as the training time for this architecture was long, possibly due to the complexity of the architecture and the large dataset. As a result, a smaller architecture was chosen, with one bidirectional layer on the encoder side and one unidirectional layer on the decoder side for my 2nd model.

IX. GENERATING SUMMARIES

To generate a summary, the input sequence is first encoded using the encoder network. Using a loop, the decoder network then takes the encoded output and generates the summary one word at a time. The initial target sequence is set to start with the 'sostok' token. The decoder predicts the next word in the summary sequence at each loop iteration using the previous word and the encoded input sequence. This process continues until either the maximum summary length is reached or the end-of-sequence token 'eostok' token is predicted. The generated summary is then returned as output.

X. RESULTS

To evaluate the performance of both models, we will use the ROUGE-1, ROUGE-2, and ROUGE-L metrics. However, before that, we will first see the predicted summaries from both models for a given original text.

Source text: - us president donald trump threatened cut us financial aid countries vote favour resolution rejecting us recognition jerusalem israel capital united nations general assembly meeting thursday vote called protest recent us veto similar measure un security council meeting nnn

Ground truth summary:- threatens to cut aid to nations over un jerusalem vote

Model-1 Prediction:- us prez trump calls us for jerusalem

Model–2 Prediction:-

The results of each model are shown in below table:

Model	*ROUGE-1*	*ROUGE-2*	*ROUGE-L*
Model 1	0.85	0.75	0.90
Model 2	0.89	0.81	0.92

From this table, we can observe that CNN and RNN models have better accuracy rates of 94.17 and 94.53. RNN model has a higher recall rate of 94.20. CNN and RNN models both have better precision rates of 95.85 and 95.14. CNN and RNN models also have better F1-Scores of 94.17 and 94.66. CNN model has the lowest runtime and RNN model cost the highest runtime among these three models. For CVE vulnerability type, it is more interesting in True Positive and True Nagative as the CTI systems are depending on the correct labels. So for this report, we will be more focused on the accuracy rate. And the best model recommended in this report is RNN model which has the highest accuracy rate of 94.17.

XI. CONCLUSION

In this project, we apply the Sequence-to-sequence encoder-decoder for text summarization. The result is not promising in Model -1; the reason could be that using multiple unidirectional LSTMs on the encoder side may not effectively capture the important information in the input text. Additionally, The stacking of multiple LSTMs may make the model more complex and harder to train, leading to lower performance. In contrast, Model -2 performs slightly better than Model -1 and uses a bidirectional LSTM on the encoder side, which allows the model to consider both past and future words in the input sequence. This can help the model better capture important contextual information, resulting in higher-quality summary outputs. Additionally, using the same number of LSTM layers on the decoder side as on the encoder side can help ensure that the decoder has enough capacity to capture the information encoded in the input, leading to more accurate and coherent summaries. Overall, the architecture of Model 2 appears to be better suited for the task of text summarization. As a part of future work, we can improve the Model's performance even further by using attention mechanisms, which can help the Model to focus on the most important parts of the input text when generating the summary. Another approach could be to incorporate pre-trained language models, such as BERT or GPT, which have been shown to improve the performance of text summarization tasks significantly.

REFERENCES

[1] C. Khatri, G. Singh, and N. Parikh, "Abstractive and extractive text summarization using document context vector and recurrent neural networks," *arXiv preprint arXiv:1807.08000*, 2018.

[2] N. Moratanch and S. Chitrakala, "A survey on extractive text summarization," in *2017 international conference on computer, communication and signal processing (ICCCSP)*. IEEE, 2017, pp. 1–6.

[3] A. M. Rush, S. Harvard, S. Chopra, and J. Weston, "A neural attention model for sentence summarization," in *ACLWeb. Proceedings of the 2015 conference on empirical methods in natural language processing*, 2017.

[4] R. Nallapati, B. Zhou, C. Gulcehre, B. Xiang *et al.*, "Abstractive text summarization using sequence-to-sequence rnns and beyond," *arXiv preprint arXiv:1602.06023*, 2016.

[5] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," *Advances in neural information processing systems*, vol. 27, 2014.