Text Summarization using NLP techniques

710075868
Department of Computer Science
University of Exeter
Exeter, UK

***Abstract*—In today's digital age, the volume of data produced daily is staggering, with 90% of the world's data generated in just the past two years alone [1]. With ever-growing digital media, there is not enough time to read through entire articles, blogs, or e-books to decide if they contain helpful information. Automatic Text Summarization solves the problem of the overabundance of digital content. By generating shorter, focused summaries of larger documents, individuals can quickly determine if the content contains the information they need. Natural Language Processing (NLP) techniques have proven to be a promising solution to this problem, making it possible to summarize large volumes of digital content efficiently.**

In this project, we aim to predict summaries based on the text with the help of 3 different strategies:- two different architectures from the Sequence-to-sequence model and the Text-to-Text Transfer Transformer (T5). Through these approaches, we aim to gain insights into the characteristics of the resulting summaries and evaluate their quality using metrics.

***Index Terms*—Text summarization, NLP, Sequence-to-Sequence, word embedding, Tokenization, LSTM, Encoder, Decoder, T5.**

I. Introduction

The amount of textual data available in various fields, including finance, research, law, and news, has increased significantly. However, the manual processing and analysis of this data can be highly time-consuming and costly, leading to the need for automated solutions. This can be extremely difficult when processing and analyzing large amounts of information, such as lengthy articles, research papers, or legal documents. The problem this project aims to solve is the ability to automatically generate accurate and concise summaries from these documents using natural language processing (NLP) techniques. By doing so, we can significantly improve productivity, save valuable time and resources, and ensure that no critical information is missed, which can have significant value in industries such as finance, news, and research.

II. RESEARCH CONTEXT

In the early days, text summarization was done using 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 progress was made. One such technique involves representing the text as a set or vector of terms and giving them weights based on their importance[2].

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. 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, and many other techniques [2]. Models for abstractive summarization fall under a larger deep learning category called sequence-to-sequence models, in which an encoder maps a sequence of source document tokens $X = [x_1, \ldots, x_n]$ to a sequence of continuous representation $Z = [z_1, ..., z_n]$ and a decoder then generates the target summary $y = [y_1, \ldots, y_m]$ token-by-token, in an auto-regressive manner, hence modeling the conditional probability: $p(y_1, \ldots, y_m \mid x_1, \ldots, x_n)$ [3].

The neural encoder-decoder architecture was initially used for text summarization by Rush [4] and Nallapati [5]. The model in Rush 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. Nallapati built an encoder-decoder model using long short-term memory (LSTM)s and augmented their model with hierarchical encoders and attention, which learns word and sentence level attention.

Many of the above works benefited from pre-trained language models, which have gained tremendous popularity over the past few years, like T5. T5 [6] is a unified Seq2Seq framework that employs Text-to-Text format to address NLP text-based problems. T5 achieves state-of the- art results and outperforms alternatives like BERT [7] and Generative pre-trained transformer 2(GPT-2) [8].

III. AIM AND OBJECTIVE

This project aims to develop and evaluate different strategies for text summarization using neural network models. Specifically, we will compare the performance of three strategies: a sequence-to-sequence model with three unidirectional LSTM encoders and one unidirectional LSTM decoder, a sequence-to-sequence model with one bidirectional LSTM encoder and

[1]https:/explodingtopics.com/blog/data-generated-per-day

[2]https://medium.com/@prasasthy.sanal/brief-history-of-text-summarization-9d1b3787a707

one unidirectional LSTM decoder and a T5 model. We will evaluate the models based on their ability to generate accurate and concise summaries of input texts.

The step-by-step procedure of the project is as follows:-

(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.

Model 3: T5.

(6) Predict summaries from Model-1,Model-2 and Model-3
(7) Evaluate based on ROUGE-1, ROUGE-2 and ROUGE-L

IV. DATASET

In this analysis, we will extract 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[3]. The first dataset,"news summary csv" contains 4514 data points with six columns including author, date, headlines, read more, text, and ctext. We will focus on the "text" and "headlines" columns for our analysis. The second dataset, "news summary more csv," contains 98401 data points and two columns," Headlines" which is a summary of the news, and "Text" which contains the original document of the news.

V. METHODS

After importing the datasets, we followed the below steps:

1) Text cleaning and preprocessing: After combining two datasets, the resulting dataset had 102915 rows with text and summary columns. However, we identified 118 NULL values in the text column and removed those rows and any other empty rows in the dataset as a part of data cleansing. Further, we performed data pre-processing steps on the remaining dataset.

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

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.

From Fig 1, we can see that the majority length of "text" lies below 50 and the majority length of "summary" below 11.

[3]https://www.kaggle.com/datasets/sunnysai12345/news-summary?select=news_summary_more.csv

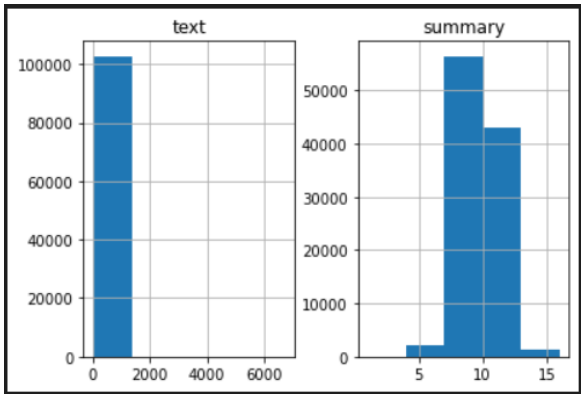

Fig. 1. Frequency Distribution of Text and Summary

So, the maximum length for the summary was fixed at 50 and the text at 11. This decision was made to optimize training efficiency by reducing computation time and memory usage.

3) Preparing tokenization: Tokenization involves breaking down text into smaller units called tokens. 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. We then converted the text and summary into integer sequences using one-hot encoding, padded them with zeros up to the maximum length, and obtained the vocabulary sizes for both.

VI. MODEL

The dataset was divided into 90% training and 10% validation data using the train-test split method. This section will define our models to address the text summarization task.

Model 1: We have used 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 Recurrent Neural Networks (RNNs [4]) 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 [9].

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 uses an LSTM layer with 300 latent dimensions and a dense layer with softmax activation to output the summary. The model is trained with the Root Mean Squared Propagation (RMSProp) optimizer and sparse categorical cross-entropy loss. It is trained on training and validation datasets to prevent overfitting, with a batch size of 128 for a maximum of 10 epochs. The decoder also uses a dense layer with softmax activation to output the final summary.

[4]https://www.ibm.com/topics/recurrent-neural-networks

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, 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. It is used with different latent dimensions of 600(twice the size of the encoder's latent dimension).

Model 3: To address long runtime issues, as we saw in the above two models, we wanted to try a pre-trained model. The T5 model architecture is a transformer-based architecture with a stack of 12 self-attention layers and a feedforward neural network (FFN) layer in the encoder. The decoder is identical to the encoder but has a cross-attention layer that enables it to focus on particular segments of the input sequence [6].

Hence, we used a pre-trained T5 model architecture by importing the SimpleT5 class and training it on our dataset. We customized the training process by specifying optional model arguments like source and target max token lengths, batch size, epochs, and early stopping. This fine-tuning approach allowed us to tailor the pre-trained model for our text summarization task.

We wanted to try the T5 pre-trained model architecture for several reasons:

- Pre-trained models have been shown to perform well on various NLP tasks, and On many of these tasks, the T5 model has produced cutting-edge results.
- Compared to training a model from scratch(like we did for models 1 and 2), a pre-trained model can save us time.
- The T5 model is a transformer-based architecture that effectively handles long-range dependencies in text, making it a suitable choice for text summarization tasks.

VII. RESULTS

We will use ROUGE-1, ROUGE-2, and ROUGE-L metrics to assess the models' effectiveness. However, before that, we will compare the predicted summaries generated by all three 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:- us us us to to of in in trump

Model–3 Prediction:- trump threatens to cut us aid to

The predicted summary for Model 3 is way better than Model 1 and Model 2.

The evaluation metrics used in this report are shown below:

ROUGE-1 = (Number of overlapping unigrams) / (Total number of unigrams in reference summary)

ROUGE-2 = (Number of overlapping bigrams) / (Total number of bigrams in reference summary)

ROUGE-L = longest common subsequence (LCS) / Total number of words in reference summary

$$ROUGE\ F1 = \frac{2 \times precision \times recall}{precision + recall} \quad (1)$$

Model	ROUGE-1 F1	ROUGE-2 F1	ROUGE-L F1
1	0.13	0	0.13
2	0.14	0	0.14
3	0.53	0.39	0.53

TABLE I
ROUGE F1 SCORES FOR ALL THREE MODELS

From this table 1, we can observe that Model 3 (T5) performs better than those Seq2Seq models. The T5 model has impressive Rouge F1 scores (0.53, 0.39, and 0.53). Models 1 and 2 have taken a lot of training time; however, in the case of T5, we have trained/fine-tuned T5 models, so it took significantly less runtime. In the case of Models 1 and 2, both took much runtime and performed poorly as per ROUGE scores. So for this report, we will more focused on the ROUGE F1 scores. Furthermore, the best model recommended for the text summarization task will be T5, with the highest ROUGE-1,2 and L scores.

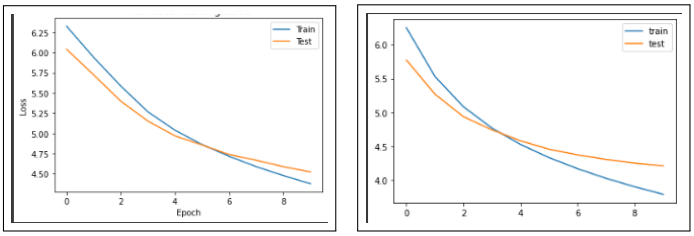

Fig. 2. Loss Rate by Epoch for Model 1 and Model 2

VIII. CONCLUSION

We have compared the performance of three unique models - two different architectures of Seq2Seq and T5. Models 1 and 2 did not perform well compared to Model 3, possibly due to their complex architecture in Model 1 and Model 2. These architectures may have led to longer training times and a higher risk of overfitting the training data.

In contrast, Model 3 utilized a pre-trained T5 model, which saved time and resources during training. Additionally, fine-tuning the pre-trained model for the text summarization task contributed to better performance on the test data. Overall, Model 3 has outperformed Model 1 and Model 2 due to its efficient use of resources and compelling architecture. As a part of future work, we can focus on optimizing the architecture of models, exploring the effectiveness of pre-trained models and fine-tuning strategies, and evaluating performance on different datasets.

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] Y. Liu and M. Lapata, "Text summarization with pretrained encoders," *arXiv preprint arXiv:1908.08345*, 2019.

[4] 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.

[5] 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.

[6] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," *The Journal of Machine Learning Research*, vol. 21, no. 1, pp. 5485–5551, 2020.

[7] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[8] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever *et al.*, "Language models are unsupervised multitask learners," *OpenAI blog*, vol. 1, no. 8, p. 9, 2019.

[9] M. Hermans and B. Schrauwen, "Training and analysing deep recurrent neural networks," *Advances in neural information processing systems*, vol. 26, 2013.

xcolor

NOTE: Please find the code in the below URL:
https://colab.research.google.com/drive/1IP0HYu0LlftT0SLGyvCqELtx0dloPhBe?usp=sharing.