TEXT BASED NEWS SUMMARIZATION

Himani Shah
Graduate Student at IU
shahhi@iu.edu

Mahadevan Iyer
Graduate Student at IU
mahiyer@iu.edu

Samardeep Gurudatta
Graduate Student at IU
samgurud@iu.edu

ABSTRACT

Text summarization using Machine Learning is a classic problem with numerous applications in our daily lives. One such application is news summarization. In today's world, multiple versions of the same news article can be found on various websites and magazines, and most of those articles are longer than 1000 words. Therefore, a solution that can condense news stories in a minimum number of words is urgently needed, making it simpler for individuals to ingest the massive amount of information published daily. People can swiftly grasp a high-level knowledge of a news story using such a method. In this report, we present a comparison of several cutting-edge approaches and compare the results. We used extractive and abstractive methods to generate news article summaries, and we compared them to those generated by humans using a metric known as the ROUGE-L (Recall-Oriented Understudy for Gisting Evaluation) score.

Index Terms— Bi-directional LSTM, Encoder-Decoder, Extractive Summarization, Abstractive Summarization, TextRank

1. INTRODUCTION

Do you open news websites and immediately begin reading every news article? Most likely not. People usually skim the short news articles and keep scrolling through and reading more details if the article's summary has piqued the reader's interest. Short, informative summaries of the news turn out to be very useful in such cases. Text Summarization refers to the process of extracting these summaries from the original news article preserving the vital information and relevant context. We aim to develop a solution to this problem by building a model using Natural Language Processing techniques for select categories of news articles i.e sports, politics, business, and entertainment.

The approach toward text summarization can be broadly classified into two – The extractive summarization approach selects a subset of sentences from the original news articles based on a scoring function to form a summary. Generating extractive summaries is a simpler task in comparison to abstractive summaries as the summaries only contain the most important sentences from the input. Abstractive summarization reorganizes the language in the text and adds new words if necessary. Multilayer bidirectional LSTM is used to encode the text and an attention framework is used for decoding.

Initially, we employ the TextRank algorithm to create a summarizer that generates a summary using the sentences taken from the text itself known as Extractive approach for text summarization. Moving on, we implement Encoder-Decoder architecture using Bi-directional Long Short Term Memory neural networks to generate summaries based on the context of the article. Finally, we augment the Encoder-Decoder architecture by applying the attention mechanism using a custom attention layer to further enhance the quality of the summary.

2. DATASET

The dataset used in this project is the CNN / DailyMail Dataset which is an English-language dataset containing just over 300,000 unique news articles as written by journalists at CNN and the Daily Mail. The CNN articles were written between April 2007 and April 2015 while The Daily Mail articles were written between June 2010 and April 2015. The data was originally collected by Karl Moritz Hermann, Tomáš Kočiský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom of Google DeepMind. Tomáš Kočiský and Phil Blunsom are also affiliated with the University of Oxford. They released scripts to collect and process the data into the question answering format. [8]

The data set consists of the following attributes:
id: a string containing the heximal formatted SHA1 hash of the url where the story was retrieved from.
article: a string containing the body of the news article. highlights: a string containing the highlight of the article as written by the article author.

The CNN/ DailyMail dataset has 3 splits: train, validation, and test. The train set consists of 287,113 instances while the validation and test sets contain 13,368 and 11,490 instances respectively.

3. RELATED WORK

Due to the nature of the problem statement this topic has been a prime area of research since a long time. Within academic research there has been a plethora of solutions presented. Even some companies have started using proprietary text summarization algorithms to summarize news articles for their viewers. In most implementations text summarization is broadly divided into two classes Extractive and Abstractive.

Extractive is a more traditional method which picks up sentences directly from the original document. Abstractive summarization is closer to what a human usually does which is conceive the text, comprehend it and re-create its core in a brief text. This method is comparatively more challenging than the previous one. This method demands sophisticated models.

An implementation of extractive and abstractive summarization with sentiment analysis is discussed in [1]. This is a two -tier approach employing two or more news stories and extracting the concerned topic then generating a summary and applying sentiment analysis on the summary. This paper also discusses various other custom implementations of summarization models and their advantages and disadvantages. This paper advocates an abstractive news summarization method as an ideal method using bidirectional LSTM layers and an encoder decoder architecture with an attention mechanism.

Another paper [6] discusses best practices for producing a proper abstractive text summarizer. It outlines the importance of pre-processing steps like vocabulary count, missing word count and word embedding counts. An implementation of two-layered bidirectional RNN with an LSTM using Bahdanau attention is discussed. Decoder with an attention layer performs better than without an attention layer.

Graph-based text summarization method has been discussed in a paper [7] which compares the similarity methods like frequency cosine and cosine in the text rank algorithm and shows the effectiveness of both the methods. It discusses clustering algorithms to avoid any kind of duplication. Paper [2] discusses sequence generative models with RNN variants, such as LSTM, and GRU, showing promising performance on abstractive document summarization. An implementation of bi-directional encoder-decoder architecture with two RNNs has been proposed.

4. EXPLORATORY DATA ANALYSIS

In our data we observed that most of the articles contain between 15- 50 sentences. However, there are few articles that have more than 170 sentences. Highlights of the article contains about 2-5 sentences.

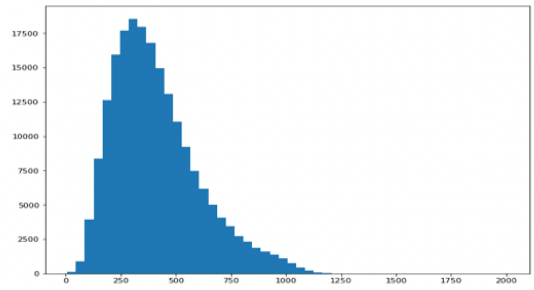

Figure 4.1 Number of sentences in each article.

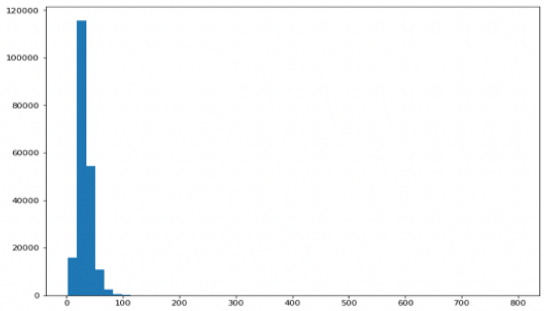

Figure 4.2 Number of sentences in each summary.

5. METHODS

5.1. Text Preprocessing

During the pre-processing phase, we apply the following procedures to clean the articles:

1. Convert articles to lowercase
2. Remove stop-words
3. Remove punctuations and special characters
4. Contraction Mapping

5.2. Extractive Text Summarization using Text Rank

Our first proposed method makes use of the Text Rank algorithm described in [8] for generating extractive summaries from the articles. Text Rank is a graph-based unsupervised algorithm for text processing that can be used to find the most relevant sentences in the articles. However, we use Cosine Similarity as a measure of similarity between the sentences.

The first step is to split an article into its constituent sentences. To convert the sentences to vectors, we used GloVe Embeddings with a dimension of 300. GloVe is an unsupervised learning algorithm for obtaining vector representations of words. Moving on, we calculate the cosine distance between all the sentence vectors from an article to

create a transition matrix. The transition matrix is used to create a graph with sentence vectors as nodes and the similarity scores as edges. We use the networkx library available in Python to create the graph. In the next phase, sentences are sorted in descending order according to the importance score calculated using the graph. Lastly, the final summary is created using the top 20% of the most important sentences from the original article.

5.3. Abstractive Text Summarization using Sequence to Sequence (Seq2Seq) Modelling

The second method models the summary generation problem as a many to many Seq2Seq modeling problem. A Seq2Seq model architecture consists of two components namely Encoder and Decoder.

Before the modeling phase, we clean the articles using the Text Preprocessing steps defined above. After cleaning the articles, we append '<sostok>' and '<eostok>' tokens to each summary to denote the start and end of the summary. Next, we plot the distribution of cleaned articles and summaries using a histogram to understand the number of words in the text. Using the histogram, we decide to keep the maximum number of words in articles and summaries to 1250 and 100 respectively.

The next step in the data preparation phase is to split the data into train and validation sets which we achieve using the train_test_split function in scikit-learn. The train set consists of 90% of the data while the remaining 10% of the articles are assigned to the validation set. Moving on, we use the Tokenizer function in TensorFlow to build the vocabulary and convert an article to words to a sequence of integers.

The Encoder is a Bi-directional Long Short Term Memory network with 400 hidden units. The input to the encoder is an embedding vector of 300 dimensions created using the Embedding Layer in TensorFlow from the tokenized word sequences. The encoder reads the entire article, one word at each timestep and generates the hidden states and cell states. Only the hidden state and cell state generated at the last time step are passed as an input to the decoder.

The decoder is a uni-directional Long Short Term Memory network which reads the entire summary, one word at each timestep. At the end of each timestep, the decoder predicts the probability of the next word in the summary given the previous word. The '<sostok>' and '<eostok>' tokens help the decoder in deciding the start and end of the summary. During the training phase, the decoder takes as input the words from the original summary at each timestep.

After the training phase, an inference architecture is set up to test the model on articles whose summaries are unknown. In the inference phase, the encoder encodes the entire sequence and generates the context vector (hidden state and cell state at the final timestep) which is used to initialize the decoder. At the first timestep, the '<sostok>' token is passed as the input to the decoder. The decoder generates the probabilities of the next word in the summary at the end of the first timestep. The word with the highest probability is selected using the Softmax activation function and is passed as the input to the decoder in the second timestep. Moreover, the weights of the decoder are also updated. The same process is repeated until the decoder generates the '<eostok>' token.

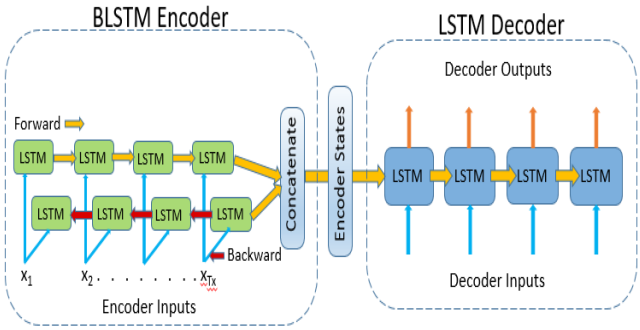

Figure 5.1 Bidirectional LSTMs as Encoder-Decoder

5.4. Limitations of the Encoder Decoder Architecture

A Seq2Seq model only considers the encoder's final cell state and hidden state to generate the context vector that initializes the decoder. This can lead to loss of information as the intermediate states are discarded. Since the context vector is of fixed length, it gets difficult for the encoder network to generate an optimal context vector for long sequences.

5.5. Abstractive Text Summarization using Sequence to Sequence (Seq2Seq) Modelling with Attention

To solve the limitations of the Encoder-Decoder architecture, the attention mechanism was introduced in the paper [10]. An attention-based model differs from Seq2Seq model in the fact that the encoder passes a lot more data to the decoder. There are two different classes of attention mechanism depending on the way the context vector is derived. Global attention mechanism uses all the hidden states of the encoder for generating the context vector while Local attention mechanism makes use on only a few hidden states for generating the context vector. This allows the decoder to increase the importance of specific parts of the news article that generate the summary.

We use an implementation of Bahdanau's Global Attention mechanism available at [11] to augment the Encoder-Decoder Architecture. Since the implementation of attention is computationally expensive, the number of hidden units in the Bi-directional LSTM Encoder network were reduced to 150 and the maximum length of articles and summary were capped at 900 and 70 words respectively.

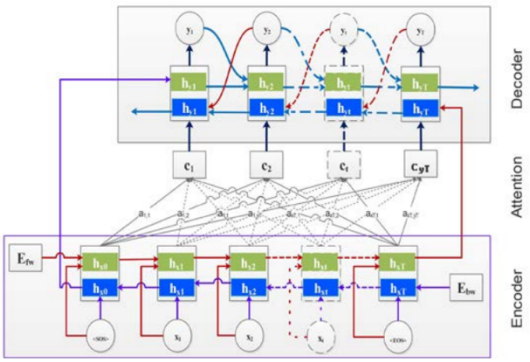

Figure 5.2 Bidirectional Encoder-Decoder with attention mechanism. Where $\boldsymbol{E}_{fw}$ and $\boldsymbol{E}_{bw}$ are the initial embeddings from the two directions and $\boldsymbol{c}_t$ is the context vector at the $\boldsymbol{t}$ decoding time step. [2]

6. RESULTS

6.1. The Complexity Analysis

The suggested solution is complicated in two ways: the first is due to the bidirectional encoder, which adds extra parameters. The basic model comprises 250,501,265 parameters for a vocabulary size of 490,000 words. There are 350,434,722 parameters in the suggested model with attention.

6.2. Evaluation Metrics

6.2.1 ROUGE Score - ROUGE-L

ROUGE Score is a method that determines the quality of the summary by comparing it to other summaries made by humans as a reference. ROUGE-L measures Longest Common Subsequence (LCS) between our proposed model output and highlights. Longer shared subsequence would indicate more similarity between two sequences.

Extractive Method	TextRank
Abstractive Method	Bidirectional LSTM without Attention
	Bidirectional LSTM with Attention

Table 6.1 Proposed Models

6.3. Experiments and Setup

On the joint CNN/ DailyMail dataset with a word embedding dimension of 300 for without Attention model and 500 for with Attention model, the word embedding is learned from scratch. 400 is the size of the model's hidden state for the former and 150 for the later model. For both the source and target, the vocabulary size is limited to 500,000. With a batch size of 32, 64, and 128 and learning rates of 0.001, 0.005, we train the model using RMSprop optimizer. The gradient clipping with a maximum gradient norm of 2 was used instead of regularization.

Below we show the results of the experiments using different approaches of text summarization discussed above.

Precision	13%
Recall	26.3%
F1	16%

Table 6.1 Precision, Recall and F1 ROUGE-L Scores on CNN/Daily Mail test set using Text Rank Extractive Summarization Algorithm

Batch Size	32		64	
Learning Rate	0.001	0.005	0.001	0.005
Precision	13.9%	7.5%	13.6%	9.3%
Recall	10.4%	7.6%	10.9%	7.8%
F1	11.4%	7.2%	11.6%	8.1%
Val Loss	2.2274	2.3249	2.1879	2.2435
Epochs	10	6	12	8

Table 6.3 Precision ROUGE-L Scores on CNN/Daily Mail test set using Bi-Directional LSTM without Attention.

Batch Size	64		128	
Learning Rate	0.001	0.005	0.001	0.005
Precision	13.6%	11%	13.3%	14.1%
Recall	11.1%	9.3%	11.3%	11.4%
F1	11.7%	9.6%	11.8%	12%
Val Loss	3.0655	3.0440	2.9759	2.9170
Epochs	10	10	10	10

Table 6.4 Precision ROUGE-L Scores on CNN/Daily Mail test set using Bi-Directional LSTM with Attention.

ROUGE-L F1 score produced by the Extractive Summarization technique is 16%. Table 6.3 demonstrates that a learning rate of 0.001 and a batch size of 64 produce

the best ROUGE-L F1-Score of 11.6% in the Seq2Seq model without attention. We achieved a ROUGE-L F1-Score of 12% with a learning rate of 0.005 when we employ the attention mechanism with the Encoder-Decoder Architecture with a batch size of 128.

When comparing the outcomes of abstractive summarization, the model with the attention mechanism comes out slightly better. Finally, when the two approaches for text summarization are compared, we can see that the Extractive technique outperforms the Abstractive method. However, we believe that this is due to the evaluation metric's limitations.

7. CONCLUSION

7.1. Summarize Approach and Future Directions

In this study, we developed an extractive summarizer using the Text Rank algorithm to extract summaries from the original articles. In order to replicate human generated highlights, we implemented a bidirectional encoder-decoder architecture with two recurrent layers, one for learning past textual context and the other for learning future textual context. The encoder's output was fed into the decoder. Then, one by one, tokens for the final summary are generated using a unidirectional LSTM. Finally, we use the ROUGE-L score to compare the predicted summaries with the expert generated summaries.

The next step to improve the predicted summaries is to augment the Seq2Seq model with attention with a Pointer Generator neural network that allows both copying words via pointing and generating words from a fixed vocabulary. Furthermore, coverage mechanism can be implemented to keep track of what has been summarized.

7.2. Project Takeaways

The generated summaries are readable and make sense, however they contain repetitions and sometimes skip over important facts or get the plot wrong altogether. Seq2seq model without Attention Mechanism performs like Extractive Text Summarization when their ROUGE-L scores are compared. Extractive summarizer had a higher recall than the Seq2Seq model. Seq2seq model with Attention Mechanism performs slightly better than Seq2seq model without Attention Mechanism.

7.3. Challenges Faced

Size of the dataset - to reduce computational complexity and time we randomly sampled 150,000 articles from the original set of approximately 300,000 articles. When the number of sentences in the input article is very large, extractive summarization fails to summarize accurately. Extractive summarization extracts a set of phrases from the input article which has the highest similarity score and combines them, which might not make sense at times. There is a possibility that output may differ from ground truth in some cases.

8. REFERENCES

[1] S. R. K. Harinatha, B. T. Tasara and N. N. Qomariyah, "Evaluating Extractive Summarization Techniques on News Articles," 2021 International Seminar on Intelligent Technology and Its Applications (ISITIA), 2021, pp. 88-94, doi: 10.1109/ISITIA52817.2021.9502230.

[2] Al-Sabahi, Kamal, Zhang Zuping, and Yang Kang. "Bidirectional attentional encoder-decoder model and bidirectional beam search for abstractive summarization." arXiv preprint arXiv:1809.06662 (2018).

[3] See, Abigail, Peter J. Liu, and Christopher D. Manning. "Get to the point: Summarization with pointer-generator networks." arXiv preprint arXiv:1704.04368 (2017).

[4] Narayan, Shashi, Shay B. Cohen, and Mirella Lapata. "Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization." arXiv preprint arXiv:1808.08745 (2018).

[5] Song, S., Huang, H. & Ruan, T. Abstractive text summarization using LSTM-CNN based deep learning. Multimed Tools Appl 78, 857–875 (2019).

[6] Mohammad Masum, Abu & Abujar, Sheikh & Talukder, Md & Rabby, Akm Shahariar Azad & Hossain, Syed. (2019). Abstractive method of text summarization with sequence to sequence RNNs. 10.1109/ICCCNT45670.2019.8944620.

[7] Mallick, C., Das, A.K., Dutta, M., Das, A.K., Sarkar, A. (2019). Graph-Based Text Summarization Using Modified TextRank. In: Nayak, J., Abraham, A., Krishna, B., Chandra Sekhar, G., Das, A. (eds) Soft Computing in Data Analytics. Advances in Intelligent Systems and Computing, vol 758. Springer, Singapore.

[8]https://www.kaggle.com/gowrishankarp/newspaper-text-summarization-cnn-dailymail

[9]https://web.eecs.umich.edu/~mihalcea/papers/mihalcea.emnlp04.pdf

[10] Bahdanau, Dzmitry & Cho, Kyunghyun & Bengio, Y.. (2014). Neural Machine Translation by Jointly Learning to Align and Translate. ArXiv. 1409.

[11] https://github.com/thushv89/attention_keras/blob/master/src/layers/attention.py