Implementation of Abstractive Text Summarization using T5 Pre-trained Models Fine-Tuned on WikiHow Dataset

Zexian Wang

Abstract

Text summarization is an essential application of text generation in natural language processing, which aims to extract key information from input text and form coherent and readable sentences. With the emergence of numerous pre-trained models such as GPT, BERT, and T5, these models have been extensively applied to text generation tasks. In this project, I implemented a text summarization model by fine-tuning the T5 pre-trained model on the WikiHow Dataset. The model was evaluated using ROUGE metrics, and its performance surpassed the Lead-N and TextRank baseline methods, validating the model's effectiveness. Based on this project's results, we can fine-tune pre-trained models on datasets from specific domains to obtain text summarization models on those specific domains.

1 Introducation

With the advancement of technology, people are exposed to an increasing amount of text information. In daily life, people often faced with a considerable amount of complex text information that requires a significant amount of time and effort to process. To address this, my project aims to develop an efficient text summarization model that quickly provides essential information from a given text. If successful, fully automated and accurate text summarization has the potential to significantly impact daily life and work. In people's daily life, it can help people to effectively process large amounts of text information and save time.

Text summarization is an application of text generation in natural language processing, which aims to extract the key information from the input text and form coherent and readable sentences. At present, the implementation of text summarization mainly includes extractive text summarization and abstractive text summarization. The extractive text summarization selects keywords and key sentences from the original text to form a summary, which has a low error rate in syntax and grammar, ensuring a certain effect. However, this method is different from the process of manual text summarization, which often requires summarizing the text based on understanding the content. At the same time, extractive text summarization also faces problems such as content selection errors, poor coherence, and poor flexibility. Abstractive text summarization, on the other hand, allows for new words or phrases to be included in the summary, making it more flexible.

With the emergence of numerous pre-trained models, such as GPT, BERT, and T5, these models have been widely applied to text generation tasks. Therefore, in this project, I will implement an abstractive text summarization model based on the T5 pre-trained model and fine-tune it on the WikiHow dataset to obtain the final model. Compared to baseline methods like Lead-N and TextRank, this abstractive text summarization method based on pre-trained models achieved better performance on ROUGE metrics. Through this project, I learned that fine-tuning pre-trained models on specific datasets can yield better results. Based on this, we can also fine-tune pre-trained models on datasets from specific domains to obtain text summarization models specific to those particular domains.

2 Data

In the context of text summarization, the selection of an appropriate dataset is important. This section aims to furnish a comprehensive account of the data employed in this project. This includes the source of the dataset, the steps taken for data preprocessing, and the statistics information of the dataset.

2.1 Dataset Source

In this project, I used WikiHow Dataset[1], which is obtained from WikiHow data dump. Traditional news summary databases are usually written by reporters, so they often follow a fixed writing style: they often start with the most striking elements and then add details. WikiHow Dataset is an article written by ordinary people, so the weights of all parts of the article may be more evenly distributed.

2.2 Dataset Statistics Information

For the dataset scale, the WikiHow Dataset has more than 230,000 pairs of articles and summaries. Therefore, in terms of data scale, it can meet the needs of text summarization deep learning model training. As shown in Table 1, the average article sentence length of the WikiHow Dataset is 100.68, the average summary sentence length is 42.27, and the compression ratio is 2.38. For the average article and summary lengths, it is better than the CNN/Daily Mail, DUC and Gigaword corpus. For the compression ratio, as the Table 1 show, it has more compression ratio the CNN/Daily Mail which means high levels of abstraction.(Koupaee and Wang, 2018)

2.3 Dataset Preprocess

As shown in Figure 1, each item in the WikiHow Dataset consists of a headline, a title, and a text. The headline serves as a reference for text summarization. The title pertains to the topic, while the text contains the article that needs to be summarized. The following are the specific preprocessing steps.

2.3.1 Removing of Invalid and Duplicate Data

For data preprocessing, the first step is to remove any null values and duplicate data to ensure that the experiment process will not be compromised by the lack of certain values. Then, we will process the "Text" in the database by first utilizing the nltk library to split it into a list of sentences and counting the number of sentences contained in each "Text". As we are conducting text summarization, we will remove samples with fewer than 3 sentences. After completing the preprocessing, we obtained approximately 185,800 samples.

[1] https://github.com/mahnazkoupaee/WikiHow-Dataset

2.3.2 Dataset Split

After removing invalid and duplicate data, I shuffled the dataset and divided it into a training dataset, validation dataset, and testing dataset with a ratio of 70:15:15 for the training and evaluation of the text summarization model.

This is an initial preprocessing of the data. Before performing text summarization, further data processing is required. Using TextRank as an example. In order to perform text summarization using the TextRank algorithm, several more preprocessing steps must be carried out. First, the text should be divided into a list of sentences. Then, each sentence would be tokenized, with punctuation marks removed, stop words filtered out, and all words converted to lower case. After tokenization, word embeddings, such as Word2Vec or GloVe, can be employed to compute sentence vectors, capturing the semantic meaning of each sentence. Using these sentence vectors, the PageRank algorithm is applied to identify the most important sentences within the text. Finally, the key sentences are combined in their original order to generate a coherent summary. It is important to note that preprocessing requirements may vary depending on the specific text summarization algorithm chosen.

3 Related Work

The following are the related work for the text summarization.

3.1 TextRank

TextRank is a graph based algorithm that summarizes text by treating it as a graph and applying the PageRank algorithm. It treats sentences in text as nodes, treats their relevance as edges, and calculates the relevance score of sentences based on their similarity. Then select the sentences with high importance scores as the key sentences of the text, and combine them to form a summary. However, the results of TextRank are greatly influenced by the results of tokenization and are easily affected by high-frequency words.(Mihalcea and Tarau, 2004)

3.2 Pointer-Generator Networks

Pointer-Generator network (PGN) is a sequence-to-sequence model with attention mechanism applied to text summarization. It uses a pointer mechanism to switch between copying words from input text and generating words from a vocabulary. The PGN decoder generates a summary by considering the

	headline	title	text
0	\nKeep related supplies in the same area.,\nMa...	How to Be an Organized Artist1	If you're a photographer, keep all the necess...
1	\nCreate a sketch in the NeoPopRealist manner ...	How to Create a Neopoprealist Art Work	See the image for how this drawing develops s...
2	\nGet a bachelor's degree.,\nEnroll in a studi...	How to Be a Visual Effects Artist1	It is possible to become a VFX artist without...
3	\nStart with some experience or interest in ar...	How to Become an Art Investor	The best art investors do their research on t...
4	\nKeep your reference materials, sketches, art...	How to Be an Organized Artist2	As you start planning for a project or work, ...

Figure 1: Sample data in the WikiHow Dataset

	WikiHow	CNN/DM
Article Sentence Length	100.68	118.73
Summary Sentence Length	42.27	82.63
Compression Ratio	2.38	1.44

Table 1: Compression ratio of WikiHow and CNN/Daily mail datasets. The represented article and summary lengths are the mean over all sentences.(Koupaee and Wang, 2018)

context and attention weight from the encoder. The pointer mechanism in PGN has improved the accuracy and processing ability of OOV words while maintaining the ability to generate new words.(See et al., 2017)

3.3 BERTSUM

The proposed method, BERTSUM, uses BERT to perform both extraction and generation tasks for document-level summarization. The architecture of BERTSUM separates sentences using the [CLS] token to capture sentence information. Additionally, segment embedding is performed based on the odd and even numbering of sentences.(Liu and Lapata, 2019)

3.4 PEGASUS

PEGASUS is a standard Transformer. In this method, it proposes a new self-supervised pre-training target (GSG) for abstract summary tasks. In PEGASUS pre-training, it deletes several complete sentences in the file, and the goal of the model is to recover the deleted sentences. Its advantage is that it can create a large number of examples without manual annotation. At the same time, it improves the universality of the model through extensive evaluation of data sets in different domains. (Zhang et al., 2019)

3.5 Scoring Sentence Singletons and Pairs

This method assumes that summary statements are obtained by compressing a single statement or merging two statements, so this method maps a single statement and a pair of statements to a unified space for sorting, and then selects a single statement and a pair of statements with higher scores according to this sorting, and finally generates summary statements by compressing a single statement and merging a pair of statements. Therefore, the model is divided into two parts. The first part converts a single statement and a pair of statements into vectors and gets a score. The second part selects a single statement and a pair of statements with a high score to get the final summary (Lebanoff et al., 2019)

After comparing with the related work on text summarization, I aim to develop a text summarization model using T5 transformer pre-trained model, fine-tuned on the WikiHow dataset in this project. As a generative model, T5 model can create new words and phrases instead of merely extracting information from the original text. This makes the generated summaries more flexible compared to extractive text summarization. Besides, the T5 model is pre-trained on a large amount of text data, acquiring extensive language knowledge. T5 demonstrates strong abstraction capabilities in text summarization tasks, enabling it to deeply understand input text and generate more accurate and targeted summaries. By fine-tuning the model on a specific dataset, it can quickly adapt to text summarization tasks and improve summary quality.

4 Methodology

In this section, I would introduce the methodology for implementing the text summarization model, including the selection of the pre-trained model, data processing, and the specific implementation of fine-tuning.

4.1 T5 Pre-trained Model

"Text-to-Text Transfer Transformer" (T5) is an Encoder-Decoder model based on the Transformer architecture(Raffel et al., 2019). The authors filtered about 750 GB of training data from the Common Crawl, which they named the "Colossal Clean Crawled Corpus" (C4). Based on this corpus, they trained the T5 model using a BERT-style denoising objective, Replace Span corruption strategy, 15% corruption rate, and corrupted span length of 3. As a result, the T5 model has acquired extensive language knowledge, making it suitable as a pre-trained model for text summarization tasks. Besides, the T5 model converts various natural language understanding and generation tasks to the text-to-text format by means of a prompt prefix. As Figure 2 shown, for text summarization tasks, "summarize" can be added as a prompt prefix to the input text, and the model can generate summary text conditioned on the prompt prefix. Therefore, in this project, I chose the "t5-small" model on Hugging Face[2] as the pre-trained model.

4.2 Data Processing

In the data preprocessing section, I have already completed the necessary preprocessing steps, such as removing missing values and duplicate samples. Additionally, I have divided the WikiHow dataset into training, validation, and testing datasets with a ratio of 70:15:15. This dataset splitting strategy has been proven effective in practice as it can provide sufficient training data to train the model, while also leaving a certain scale data for validation and testing, in order to evaluate the model's performance on previously unseen data.

Next, I used the datasets library[3] to convert the preprocessed data into training, validation, and testing datasets for further data processing. After obtaining the dataset, I used the T5 tokenizer to encode the text data. For the input text, I prepended "summarize" as a prompt prefix to the input text, signaling to the T5 model that the intended task is text summarization. Then, I used the tokenizer to convert it to input ids and attention masks as the model inputs. Since the maximum input length for the "t5-small" pre-trained model is 512, I set the maximum length for the tokenizer to 512. Input texts that do not meet this requirement would be truncated or padded. For the reference summaries, which are the "headlines" in the WikiHow Dataset, I used the tokenizer to convert them to input ids and attention masks. And I set the max length to 150. Then, I used the input ids of the reference summaries as labels for subsequent model training.

4.3 Fine-tune Training Details

In this part, I will specifically introduce the process of fine-tuning the pre-trained model, including the choice of the loss function, optimizer, and the detailed implementation of training.

4.3.1 Seq2SeqTrainer

Since text summarization is a sequence-to-sequence process, I chose to use the Seq2SeqTrainer[4] for training and the Seq2SeqTrainingArguments[5] for setting the hyperparameters during the training process. In choosing the batch size and gradient accumulation step, I balanced the trade-off between training speed and resource consumption. Therefore, I set the batch size to 4 and the gradient accumulation step to 8 to speed up the training process without overconsuming computational resources. It can also effectively reduces memory usage while maintaining a certain degree of stability in gradient updates.

Besides, since both input and output are sequences during the text summarization training process, I used DataCollatorForSeq2Seq[6] to align the input and output sequences, allowing the model to process different length sequence data in batches. Additionally, to enhance the training performance during the fine-tuning process, I chose a suitable loss function and optimizers for the Seq2SeqTrainer.

[2] https://huggingface.co/t5-small

[3] https://huggingface.co/docs/datasets/

[4] https://huggingface.co/docs/transformers/main_classes/trainer#transformers.Seq2SeqTrainer

[5] https://huggingface.co/docs/transformers/main_classes/trainer#transformers.Seq2SeqTrainingArguments

[6] https://huggingface.co/docs/transformers/main_classes/data_collator

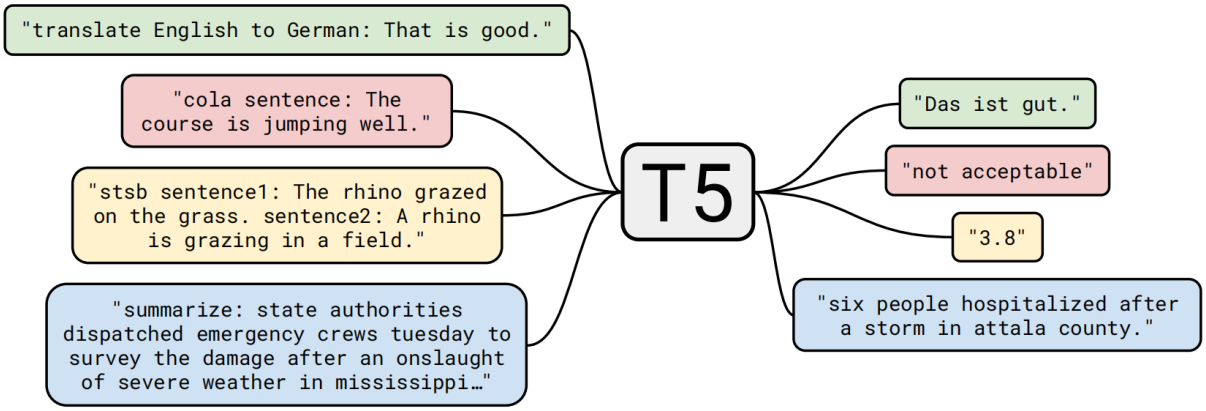

Figure 2: A diagram of our text-to-text framework(Raffel et al., 2019)

4.3.2 Loss Function

In the data processing step, the model's input and label have been determined. The label consists of the tokenized input ids of the reference summaries. During the training process, I used the Cross-Entropy loss function to measure the loss between the generated summaries (model output) and reference summaries (label).

4.3.3 Optimizer and Scheduler

For the optimizers of the Seq2SeqTrainer, I chose AdamW as the optimizer. AdamW is an improved adaptive learning rate optimizer, which outperforms the traditional Adam optimizer in terms of weight decay handling. This helps the model to converge faster. In the parameter setting process, I set the learning rate to 1e-5 and the weight decay to 0.01. Besides, I also divided the model parameters into two groups: with weight decay and without weight decay. For the "bias" and "LayerNorm.weight" parameters, not using weight decay would prevent the performance of the pre-trained model from declining.

Additionally, I used a scheduler to adjust the learning rate during the training process. Firstly, the learning rate is linearly increased during the warm-up process, and then linearly decreased throughout the training process. Using a scheduler helps stabilize the model parameter updates in the early stages of training, thereby improving convergence performance. The warm-up steps and training steps of the scheduler are calculated based on training parameters, dataset size, and training epochs.

After setting the Seq2SeqTrainer's batch size, loss function, optimizers, and other parameters, I completed two training epochs and obtained the final text summarization model, which is based on the T5 pre-trained model and fine-tuned on the wikiHow Dataset.

5 Evaluation

5.1 Evaluation Metrics

ROUGE is an important evaluation metric for evaluating the generation of text summaries. It contains: $Rouge_N$ and $Rouge_L$. For $Rouge_N$, it compares the summaries generated by the comparison algorithm with the reference summaries in the dataset and calculates the corresponding scores to measure the performance of the summary generation. The denominator is the number of n-grams, and the numerator is the number of n-grams shared by the reference summary and automatic summary $Rouge_N = \frac{\sum_{S \in ReferenceSummaries} \sum_{gram_n \in S} Count_match(gram_n)}{\sum_{S \in ReferenceSummaries} \sum_{gram_n \in S} Count(gram_n)}$ And for $Rouge_L$, it refers to the longest common subsequence. The specific calculation formula is as follows

$$R_{lcs} = \frac{LCS(X,Y)}{m}$$

$$P_{lcs} = \frac{LCS(X,Y)}{n}$$

$$F_{lcs} = \frac{(1+\beta^2)R_{lcs}P_{lcs}}{R_{lcs} + \beta^2 P_{lcs}}$$

LCS(X,Y) represents the length of the longest common subsequence of X and Y, m and n represent the length of the reference summary and the generated summary respectively, and R_{lcs} and P_{lcs} represents recall rate and accuracy rate respectively, beta is a hyperparameter, and finally F_{lcs} is the $Rouge_{lcs}$

Model	WikiHow								
	ROUGE-1			ROUGE-2			ROUGE-L		
	R	P	F1	R	P	F1	R	P	F1
TextRank	34.34	22.07	23.70	8.31	5.05	5.52	32.19	18.96	21.72
Lead-N	42.54	22.29	26.00	10.82	5.45	6.44	36.85	18.79	22.75
T5 (End to End)	20.37	24.57	20.20	3.94	5.30	3.98	18.89	22.72	18.71
T5 (Fine-tuned)	26.14	46.52	30.23	11.04	19.44	12.64	25.49	45.42	29.47

Table 2: The ROUGE metrics of different methods on WikiHow dataset

In this project, I chose to use the ROUGE evaluation metric. Based on the testing dataset, I compared the summaries generated by the model with the reference summaries in the dataset, and calculate the recall, precision, and F1 score on $Rouge_1$, $Rouge_2$, and $Rouge_L$ metrics by using the rouge[7] library.

5.2 Baseline

In this project, I have chosen two baseline methods: TextRank and Lead-N. These baselines can serve as a foundation for evaluating the effectiveness of our model.

Lead-N is a very intuitive and straightforward baseline method. It selects the first N sentences of the source document as the summary. This method is based on the assumption that the most important information in an article is usually placed at the beginning. However, for the wikiHow dataset, the most important information is not usually placed at the beginning. Therefore, I slightly modified the Lead-N method, changing it to segment the text content and select the first sentence of each segment as the summary. In this way, the summary can better capture the important information in the Text.

For another baseline method, TextRank is an unsupervised graph-based ranking algorithm for text summarization, inspired by Google's PageRank algorithm. It treats sentences in the text as nodes and the relationships between sentences as edges, calculating the weight matrix based on sentence similarity. Then, it selects sentences with high importance scores as the key sentences of the text. Finally, the key sentences are combined to form the summary. In this project, I first preprocessed the data by splitting the complete text into sentences, tokenizing them, and removing punctuation, stopwords, etc. Then, I converted all words to lowercase. Using GloVe, I obtained word embeddings and calculated sentence vectors based on these word embeddings. I computed the similarity matrix using cosine similarity and then applied the PageRank[8] algorithm to obtain sentence scores. Finally, I selected the top 3 sentences with the highest scores to form the summary.

5.3 Evaluation Results

Using the baseline methods, the original T5 model and my fine-tuned model, I generated the summary for the text in the testing dataset. Then I calculated the recall, precision, and F1 score using $Rouge_1$, $Rouge_2$, and $Rouge_L$ metrics by comparing the generated summaries with the reference summaries provided in the dataset.

The specific evaluation results are shown in Table 2, where I converted the recall, precision, and F1 score into percentage form, making it easier to compare the performance of the methods. Based on these results, I can validate the effectiveness of my proposed method.

6 Discussion

Based on the evaluation results in Table 2, I would discuss and analyze the overall performance of the fine-tuned model and its comparison with other methods.

6.1 Overall Performance Analysis

Based on the evaluation results, the fine-tuned text summarization model achieves a significant improvement in the precision and F1 score of ROUGE-1, ROUGE-2, and ROUGE-L. Besides, the generated summaries have better readability compared to the baseline methods. But the recall performance is relatively poor, indicating that the generated summaries might miss some key information that should be included in the reference

[7] https://pypi.org/project/rouge/

[8] https://networkx.org/documentation/stable/reference/algorithms/generated/networkx.algorithms.link_analysis.pagerank_alg.pagerank.html

summaries. I would specifically analyze the reasons for the model's performance during the comparison with other methods.

6.2 Compared to Baseline Methods

Compared to the TextRank and Lead-N baseline methods, the fine-tuned T5 model has a significant improvement in the precision and F1 scores of ROUGE-1, ROUGE-2, and ROUGE-L. This demonstrates that the T5 pre-trained model has a stronger ability to accurately extract key information, recognize article structure, and generate summaries. However, the fine-tuned T5 model's recall performance is relatively lower than that of TextRank and Lead-N, which means some key information is left out. I believe this might be due to the truncation of the original input text during the fine-tuning and evaluation processes when it exceeds the model's maximum sequential length, resulting in the omission of some key information. Additionally, since the WikiHow dataset size is quite large, to improve the speed of training and evaluation, I set the maximum length of the generated summaries to 150. As a result, this may lead to high precision and F1 scores, but low recall because the generated summaries are much shorter than the reference summaries.

6.3 Compared to the Original T5 Model

Compared to the original T5 model, the fine-tuned T5 model demonstrates a significant improvement in recall, precision, and F1 scores. Furthermore, I also found that the performance of the model after two epochs is better than after one epoch. This confirms that fine-tuning the pre-trained model on a specific dataset can improve its performance in the text summarization task, validating the effectiveness of the project's methodology.

7 Conclusion

In this project, I fine-tuned a pre-trained model (T5) on the wikiHow dataset to create a text summarization model. Through data preprocessing, selection of the pre-trained model, and fine-tuning of the wikiHow dataset, I successfully implemented this text summarization model. To validate the effectiveness of the model, I chose Lead-N and TextRank as the baseline methods. The evaluation results based on ROUGE metrics show that my fine-tuned model performs better than the baseline method. Based on the analysis of the evaluation results, the following conclusions can be drawn. Fine-tuning the pre-trained model on a specific dataset can lead to better accuracy and F1 scores for the text summarization model. This may be due to the fact that the T5 pre-trained model already has a large amount of text knowledge, and after fine-tuning, it is more capable of extracting key information, identifying article structure, and generating text summaries on the WikiHow dataset. However, due to the maximum length limitation of the T5 pre-training model, the generated summaries might miss some key information in the original text, resulting in a low recall on the ROUGE metric. Therefore, we need to consider generating more comprehensive and accurate summaries that include more relevant key information to improve recall and summary quality.

8 Other Things We Tried

During this project, after obtaining a fine-tuned model based on the "t5-small" pre-trained model, I obtained good results in the evaluation metrics. Therefore, I hope to fine-tune with "t5-large" because it has more parameters and can better understand the text, thereby might achieve better results. However, due to the large size of the pre-trained "t5-large" model, even on Great Lakes, the "CUDA out of memory" error still occurred. I tried reducing the batch size and setting a larger gradient accumulation step, but still couldn't solve the problem. So I ultimately chose to use "t5-small" as the pre-trained model.

In addition, I also tried to combine abstractive and extractive text summarization because the maximum length of the "t5-small" pre-trained model is only 512, which may not be enough for some longer texts. Therefore, I wanted to use TextRank to extract key sentences from the original text and then fine-tune the T5 pre-trained model using these key sentences as input text. However, the results obtained from this approach were worse than using TextRank and T5 separately. I think this may be because simply extracting key sentences makes it difficult for the pre-trained model to understand the entire article and much significant information might be lost. Besides, abstractive text summarization has the disadvantage of generating uncertain content, which leads to poor summary generation results. And it might also be because this method is not reasonable. For example, changing the maximum sequence length of a T5 model or choosing variants of the T5 model that allow for larger in-

put lengths might be useful for long input text, although these methods require more computational and memory resources. A further literature review is needed to find a more suitable approach.

9 What You Would Have Done Differently or Next

Through this project, I learned how to fine-tune a pre-trained model for a specific dataset. If I could start some parts of the project over, I would make the following improvements:

9.1 Choose Domain-specific Dataset

In this project, I used the WikiHow Dataset for text summarization. The advantage of the WikiHow dataset has a clear structure and provides reference summaries (headlines) for model training and evaluation. But it covers a wide range of topics, including health, entertainment, technology, etc., with relatively low expertise. If I could start again, I would select a more specialized dataset in a specific domain for fine-tuning. This could result in a text summarization model tailored to a specific domain, and the fine-tuned model might have a better performance.

9.2 Experiment with Different Pre-trained Models

In this project, I only chose the T5 pre-trained model for text summarization. However, there are many other models available for text summarization, such as BERT and PEGASUS. If I had more time, I would fine-tune multiple pre-trained models, compare their results, and analyze the advantages and disadvantages of different models to achieve better performance.

9.3 Improve Text Summarization Methods

As mentioned earlier, input texts exceeding the model's maximum sequence length are truncated, which can restrict the understanding of the text content and structure and negatively impact the performance of the text summarization model. Using models with longer maximum sequence lengths requires more computational resources. In this project, using TextRank to extract key sentences for summarization didn't get better results. Next, I would like to explore different text summarization methods, such as not only extracting key sentences but also important entities, vocabulary, or phrases as input. This additional information could help the model better understand the article's content and improve the model performance.

References

Mahnaz Koupaee and William Yang Wang. 2018. Wikihow: A large scale text summarization dataset. *ArXiv*, abs/1810.09305.

Logan Lebanoff, Kaiqiang Song, Franck Dernoncourt, Doo Soon Kim, Seokhwan Kim, W. Chang, and Fei Liu. 2019. Scoring sentence singletons and pairs for abstractive summarization. In *Annual Meeting of the Association for Computational Linguistics*.

Yang Liu and Mirella Lapata. 2019. Text summarization with pretrained encoders. *ArXiv*, abs/1908.08345.

Rada Mihalcea and Paul Tarau. 2004. TextRank: Bringing order into text. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 404–411, Barcelona, Spain. Association for Computational Linguistics.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *CoRR*, abs/1910.10683.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. *CoRR*, abs/1704.04368.

Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter J. Liu. 2019. Pegasus: Pre-training with extracted gap-sentences for abstractive summarization. *ArXiv*, abs/1912.08777.