Text Summarization through Transformers

Dhanny Indrakasuma
School of Information
The University of Texas at Austin
Austin, TX
dhannywi@utexas.edu

Manasvini Karthikeyan
School of Information
The University of Texas at Austin
Austin, TX
manasvinik@utexas.edu

Shashwat Jyotishi
School of Information
The University of Texas at Austin
Austin, TX
shashwatj9914@utexas.edu

***Abstract*—**

Text summarization is the process of creating a short, accurate, and fluent summary of a longer text document to distill the most important information from a source text. [9] Automatic text summarization is a common problem in machine learning and natural language processing (NLP). [9] With an ever-growing amount of text data generated daily, automatic text summarization methods are greatly needed to help users consume and discover relevant information more quickly. In this project, we intend to develop a text summarization model that would be trained on a Kaggle News Dataset containing 4515 news articles along with their summaries [2].

I. Introduction

In the last decade, there have been significant advancements in the field of NLP. This made way for amazing technologies including but not limited to language translation, text summarization, voice recognition, etc. Developments in language translation and text summarization hold the potential to overcome barriers between the interaction of machine language and natural language. [15]

In this project, we intend to develop a model which, when given a piece of text in English, would generate summaries of the text in the same language. We started by gathering the desired dataset from Kaggle. Then, we preprocessed the data, performed exploratory data analysis to gain insights, produced embeddings, and used various architectures like spaCy, gensim, Text-to-Text Transfer Transformer (T5) Transformer, and T5 Transformer implementation through Spark NLP to perform the summarization task.

The Transformer architecture was created with translation in mind. According to the Hugging Face explanation, the encoder receives inputs (sentences) in one language during training, while the decoder receives the same sentences in the target language. [14] The attention layers in the encoder can employ all of the words in a sentence (since, as we just saw, the translation of a given word can be dependent on what is after as well as before it in the sentence).[14] The decoder, on the other hand, operates in order and can only focus on the words in the phrase that it has previously translated (so, only the words before the word is currently being generated). [14]

For example, once the first three words of the translated target have been anticipated, we provide them to the decoder, which then uses all of the encoder's inputs to try to predict the fourth word.

The decoder is fed the entire target during training (when the model has access to target phrases), but it is not allowed to use future words (the problem would be much easier if it had access to the word at position 2 when trying to forecast the word at position 2!). When attempting to anticipate the fourth word, for example, the attention layer will only be able to see the words in positions 1 to 3.

II. Dataset

The Kaggle News dataset contains 4515 examples of news articles along with their summaries [2]. The dataset comprises the following features: Author_name, Headlines, Url_of_Article, Summary, Complete_Article [2]. In order to train the text summarization model, we have used only two features - Summary and Complete_Article.

III. Exploratory Data Analysis (EDA)

The first step that we took to explore our dataset was to look at the statistics of the data that was pre-processed using spaCy. In the first document we experimented with, we looked into the different part-of-speech tags that appeared. We also visualized the entities present in the first document and used the dependency visualizer to show part-of-speech tags and syntactic dependencies. [7] From the results, we decided to use part-of-speech tags “PROPN”, “ADJ”, “NOUN” and “VERB” to filter our keywords for the spaCy model.

Next, we looked into the statistics of randomly selected articles on different topics. For this exploration, we used five articles each on the three most prominent topics in the dataset, which are politics, Bollywood entertainment, and sports. Overall, sports articles have the lowest sentence count at 12 sentences on average, followed by Bollywood news at an average of 20 sentences, and followed by Political news with the highest count at 63 sentences on average. Political news also has the highest word count at an average of 937 words per article, followed by Sports news at 338 words per article, and Bollywood news at 318 words per article. This statistic suggests that on average, the sports articles selected have a higher word count per sentence than the political or Bollywood articles at 28 words per sentence.

However, the original summaries for all three topics are fairly similar in terms of word count and sentence count at between 59-57 words and 4-3 sentences on average. When

looking at the percentage of sentences and words used between the articles and the summary, we find that the percentage varied greatly depending on the topic. This was calculated by looking at the sentences or word count of the summary and dividing it by the sentences or word count of the article.

Political news has the lowest summary to article percentage at 4.73% for sentence count and 6.23% for word count. Bollywood comes next at 14.85% for sentence count and 17.86% for word count. Sports news has the highest summary to article percentage at 28.57% for sentence count and 17.58% for word count. These results are expected since Political news has the highest words and sentence count, yet its summaries are similar in length to other topics. The table below highlights the results of our exploratory data analysis in greater detail.

Averages	Politics	Bollywood	Sports
Sentences (article)	63	20	12
Sentences (summary)	3	3	4
Words (article)	937	318	338
Words (summary)	58	57	59
Word Length	5	4	5

Fig 1. Results from Exploratory Data Analysis

IV. Data Preprocessing

IV.I Dealing with Null Values

We had around 4515 news articles along with their summaries. While exploring the dataset, we found that it had 119 null values. We dropped these samples from the dataset so that our results will not be affected.

IV.II Dealing with Stop Words

In NLP, stop words are a set of most commonly used words. [1] They contain very little semantic/useful information. Thus, it is imperative to identify and remove these stop words from your dataset. We used the lang module by spaCy to perform stop words removal. [7]

IV.III Dealing with punctuation marks

Just like stop words, even punctuation marks in the news articles needed to be dealt with since they are not required to understand the semantic information between different words. In order to remove these punctuation marks, the punctuation module in the string package came in very handy.

V. Model Implementation

V.I Implementation with spaCy

After we pre-processed the data, we used the python library spaCy to run the text summarization algorithm. We used en_core_web_sm, an English pipeline by spaCy, that is particularly optimized for CPU. [7]

1. Algorithm

The algorithm begins with tokenization and goes on to compute the word frequencies for all the words in a given news article. The word frequencies are computed by dividing the total occurrences of a single word in a given article by the largest word occurrence value. Then, we computed scores for all the sentence tokens. It was done by adding the word frequencies for all the words present in the sentence. Finally, a summary was derived for every news article by passing a target summary length (which in our case was 30% of the total article length) and sentence scores to the nlargest method of the heapq package in python. For illustration, a news article that was 800 words long, will generate a summary of approximately 370 words.

2. Evaluation Metrics

In our initial experiment, we pulled out a random article sample from the dataset to perform the text summarization. The generated summary from the model was then compared to the article's original summary to obtain the ROUGE score and BLUE score of the result.

- *ROGUE Score*

We have used ROGUE Score as an evaluation metric. ROGUE, or Recall-Oriented Understudy for Gisting Evaluation Metric measures the number of matching 'n-grams' between our model-generated text and a reference. [5] The reference, in our case, was the original article summary that we already had in our dataset.

Following are the Rogue Scores that we got on a specific summary of the testing dataset:

	f1 Score	Precision	Recall
Rogue-1	0.50	0.37	0.79
Rogue-2	0.29	0.20	0.51
Rogue-L	0.48	0.35	0.75

Fig 2. Rogue scores with spaCy

- *BLEU Score*

BLEU, or BiLingual Evaluation Understudy Score is a metric for automatically evaluating machine-translated text. [6] It is represented as a number between zero and one that measures the similarity of the machine-translated text to a reference text. [6] This is another evaluation metric that we have used to assess the performance of the spaCy-based

implementation. Following is the BLEU score that we received on the same testing set summary:

BLEU Score	0.40

Fig 3. BLEU score with spaCy

3. Inference

We can see that with the spaCy implemented model, our metric values varied greatly. This led us to make an attempt on generating summarization through a T5 Transformer, a stronger and more robust model.

V.II Implementation with T5 Transformer

T5 is an encoder-decoder model pre-trained on a multi-task mixture of unsupervised and supervised tasks and for which each task is converted into a text-to-text format. [12] T5 works well on a variety of tasks out-of-the-box by prepending a different prefix to the input corresponding to each task. [12]

T5, or Text-to-Text Transfer Transformer, is a text-to-text method of Transformer-based architecture. [12] Every task is framed as feeding the model text as input and training it to output some goal text, including translation, question answering, and classification. [12] This allows us to use the same model, loss function, hyperparameters, and other parameters across a diverse set of tasks.

T5 differs from BERT due to two major reasons. The two reasons why we chose T5 to do the summarization over BERT:

- A causal decoder has been added to the bidirectional architecture.
- A variety of other pre-training assignments to replace the fill-in-the-blank cloze task

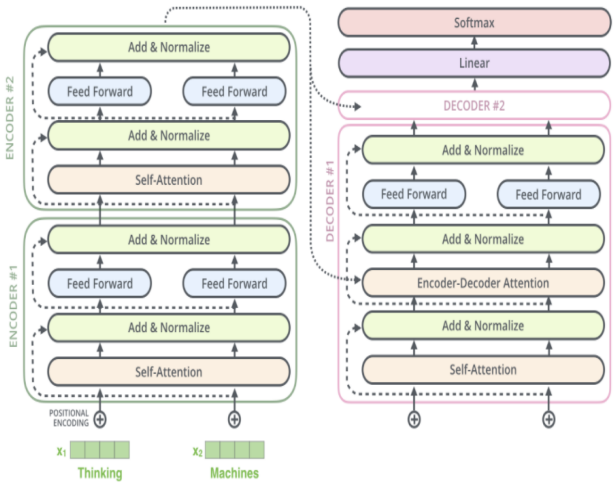

Fig 4. T5 Transformer Model Architecture [13]

1. Algorithm

T5 is an abstractive summarization algorithm. [12] It means that it will rewrite sentences when necessary rather than just picking up sentences directly from the original text. The algorithm is split into three parts. Import and Initialization, Data and Tokenization, and Summary Generation. PyTorch-Transformers (formerly known as pytorch pretrained bert) is a library of state-of-the-art pre-trained models for Natural Language Processing (NLP).

After importing AdamW (For optimization), T5ForConditionalGeneration, T5tokenizer, we can perform standard preprocessing, splitting the data, and dropping the nulls and the tokenizer takes every word or punctuation and converts them to numeric IDs. The T5 model will read and map that to a pre-trained word embedding. We can then visualize the number of tokens in the original text and the summary to gather how truncated the summary actually is.

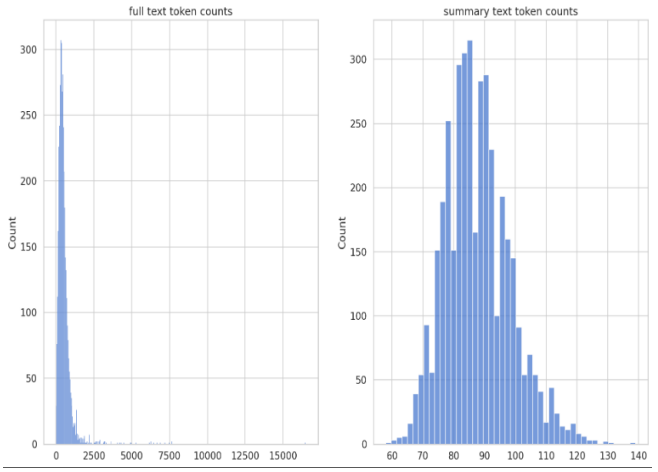

Fig 5. Token counts for Text and Summary

After training and fitting the model, we can test it on the testing data and calculate the ROUGE and BLEU score to evaluate the results.

2. Evaluation Metrics

- *ROGUE Score*

With the T5 Transformer, following are the Rogue Scores that we got on on a specific summary in the testing dataset:

	f1 Score	Precision	Recall
Rogue-1	0.70	0.66	0.74
Rogue-2	0.44	0.43	0.45
Rogue-L	0.57	0.54	0.60

Fig 6. ROGUE scores with T5 Transformer

- *BLEU Score*

With the T5 Transformer, the following is the BLEU Score that we got on a specific summary in the testing dataset:

BLEU Score	0.56

Fig 7. BLEU score with T5 Transformer

3. Inference

With the T5 Transformer, we could see a significant improvement in the performance of our text summarization model. With the T5 Transformer, recall of Rogue-1 came out to be 0.74, which is a considerable improvement from the previous Rogue-1 recall of 0.14.

V.III Implementation with Gensim

1. Algorithm

Gensim is a free open-source Python library for representing documents as semantic vectors, as efficiently (computer-wise) and painlessly (human-wise) as possible. [8] The gensim library utilizes TextRank algorithm to generate extractive summarization. [8] Upon importing the summarization package from gensim, we tested it on the same article we used on the other algorithms in order to create a fair comparison.

2. Evaluation Metrics

- *ROGUE Score*

With gensim, the following are the Rogue Scores that w*UE Scoree got on a specific summary in the testing dataset:*

	f1 Score	Precision	Recall
Rogue-1	0.71	0.66	0.77
Rogue-2	0.41	0.38	0.45
Rogue-L	0.67	0.62	0.73

Fig 8. ROGUE scores with gensim

- *BLEU Score*

With gensim, the following is the BLEU Score that we got on a specific summary in the testing dataset:

BLEU Score	0.41

Fig 9. BLEU score with gensim

V.IV Implementation with SparkNLP

1. Algorithm

Spark NLP is an open-source natural language processing library, built on top of Apache Spark and Spark ML. It provides an easy API to integrate with ML Pipelines and it is commercially supported by John Snow Labs. [10]

The library covers many common NLP tasks, including tokenization, stemming, lemmatization, part of speech tagging, sentiment analysis, spell checking, named entity recognition, and more. [10] Spark NLP library is written in Scala and it includes Scala and Python APIs for use from Spark. It has no dependency on any other NLP or ML library. [10]

While using the Google T5 model for text summarization, we faced the challenge of heavy resource requirements. The full model is more than thirty times the size of established general-purpose NLP models like BERT, and these earlier models were already big enough to be difficult and expensive to use on commodity GPU hardware.

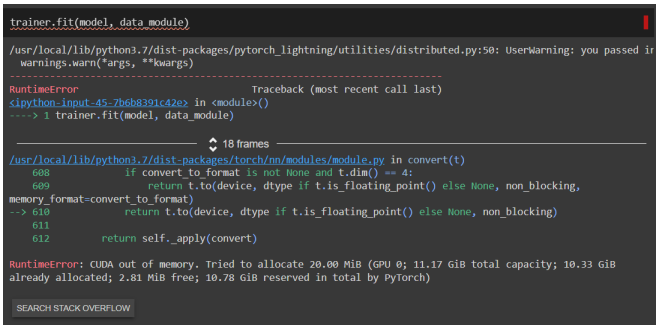

Fig 10. CUDA out of memory

SparkNLP enables extensive functionality for Text-to-Text Transfer Transformer (T5) and the ability to process it efficiently, and fast. It borrows parallelism, concurrency, and distributed computing from Apache Spark.

2. Implementation

In Spark NLP, all Annotators are either Estimators or Transformers as we see in Spark ML. [10] There are two types of annotators: the Annotator Approach and the Annotator-Model. [10] Annotator-Approach extends Estimators from Spark ML, which are meant to be trained through fit(), and Annotator-Model extends Transformers which are meant to transform data frames through transform().

When we fit() on the pipeline with Spark data frame (df), its text column is fed into DocumentAssembler() transformer at first and then a new column “document” is created in Document type (AnnotatorType). To be able to perform any NLP task on a text document, we need to get raw data transformed into Document type at first.

DocumentAssembler() is a special transformer that creates the first annotation of type Document which may be used by annotators down the road. DocumentAssembler() comes from sparknlp.base class and has the following settable parameters:

- setInputCol() -> the name of the column that will be converted. We can specify only one column here. It can read either a String column or an Array[String]
- setOutputCol() -> optional : the name of the column in Document type that is generated. We can specify only one column here. Default is ‘document’

- setIdCol() -> optional: String type column with id information
- setMetadataCol() -> optional: Map type column with metadata information

After the data frame is transformed using Document-Assembler, we integrated the annotator with SparkML Pipelines. A pipeline chains multiple Transformers and Estimators together to specify an ML workflow, in this case, t5-small. Fitting the model took considerably less time, due to the parallelism offered by SparkNLP, and to test the model, we took one reference and a candidate from the output.

- *ROUGE Score*

  We got some tremendous ROUGE score results with the sparkNLP implementation.

	f1 Score	Precision	Recall
Rogue-1	0.74	1	0.6
Rogue-2	0.68	0.91	0.54
Rogue-L	0.74	1	0.6

Fig 11. ROGUE scores with SparkNLP

- *BLEU Score*

  The BLEU score with the sparkNLP implementation was fairly similar to what we got to see before through previous models.

BLEU Score	0.43

Fig 12. BLEU score with sparkNLP

3. *Inference*

With gensim, our scores are better than the spaCy implemented model but not than the T5 Transformer one. Until now, the T5 Transformer with SparkNLP has remained the best performer.

VI. Analysis of Performance on different categories of Articles

After we analyzed each and every model individually, we wanted to dive deep into their performance and see how well they perform against different categories of articles. Looking at the articles present in our dataset, we discovered that our Kaggle News Dataset was predominantly rich in three distinct categories of news articles - Sports, Politics, and Entertainment (Bollywood). For that reason, we randomly picked five articles from each category and generated summaries for each category. Then, we compared it against the original summaries to evaluate their mean ROUGE F1 and BLEU scores. This allows us to see how our models were performing in different categories of news articles.

VI.I Performance of spaCy over the three categories

With spaCy, we got the following Mean Rogue-1 F1 scores over the three categories:

	Politics	Entertainment	Sports
Mean Rogue-1 F1 Score	0.28	0.27	0.33
Mean BLEU Score	0.30	0.28	0.26

Fig 13. Mean Metric Values with spaCy

While we were getting poor mean scores, doing this analysis with spaCy gave us an intuition that our model was indeed a bit consistent when it came to summarizing articles over different genres.

VI.II Performance of gensim over the three categories

With gensim, results in some categories seemed to improve on some topics. On the other hand, some metric values worsened. A lot of variations could be seen with the gensim model. While we were getting a significant improvement in the Mean ROUGE F1 Score over sports topics and BLEU score on entertainment topics, BLEU scores of political articles decreased drastically.

	Politics	Entertainment	Sports
Mean Rogue-1 F1 Score	0.24	0.22	0.44
Mean BLEU Score	0.27	0.36	0.23

Fig 14. Mean Metric Values with gensim

VI.III Performance of T5 Transformer (through spark NLP) over the three categories

Initially, when we tested T5 Transformer to summarize articles individually, we were getting ROUGE Scores up to 0.7 and BLEU scores up to 0.6. Similarly, with spark NLP implementation, we were getting amazing results - ROUGE scores up to 0.8 and BLEU scores up to 0.65. Now, when we grouped articles into three categories and computed the mean ROUGE and BLEU scores, we noticed that our model was performing consistently well over all the categories of the articles. Thus, two good takeaways from this analysis were the consistency of the sparkNLP summarization model and its amazing performance, especially in terms of how similar our generated summaries were to the references (high BLEU scores). Irrespective of the category of the article being fed to it, it did a good job across all the news categories.

	Politics	Entertainment	Sports
Mean Rogue-1 F1 Score	0.21	0.34	0.40
Mean BLEU Score	0.96	0.85	0.92

Fig 15. Mean Metric Values with sparkNLP

VI.IV Analysis of the Performances of all the Models

Finally, after having conducted exhaustive research on all the models and after comparing their performances, we concluded that the T5 Transformer, when implemented through sparkNLP, was giving us the most optimal performance. Be it consistency or highly similar generated summaries, T5 with sparkNLP turned out to be at the top, followed by manual T5 Transformer implementation, gensim, and spaCy. Looking at the two evaluation metrics, ROUGE and BLEU scores, let's see how we analyzed our performance through these evaluation metrics.

- Analysis through ROUGE Score

  The F1 Score and Precision kept increasing as we tried more complex models and eventually peaked when it came to transformers.

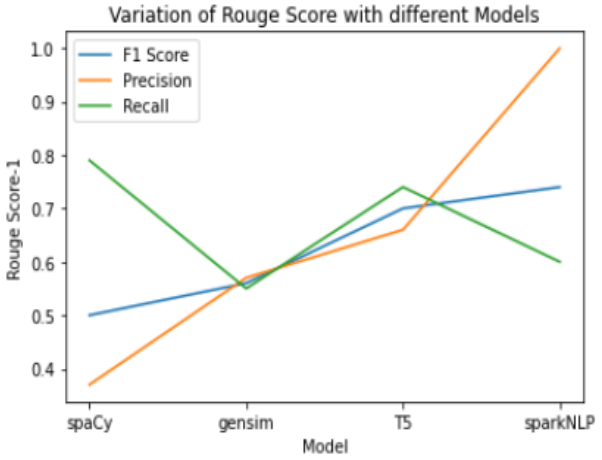

Fig 16. Variation of Rouge-1 with various models

- Analysis through BLEU Score

  BLEU Score, being a measure of the similarity between the machine summarized text and the reference, increased as we tried more robust models and ultimately excelled in the performance when it came to T5 Transformer and sparkNLP. We categorized articles into three groups - politics, sports, and entertainment and saw how our models were performing against all of the categories. T5 and sparkNLP consistently achieved a high BLEU score over all of the categories.

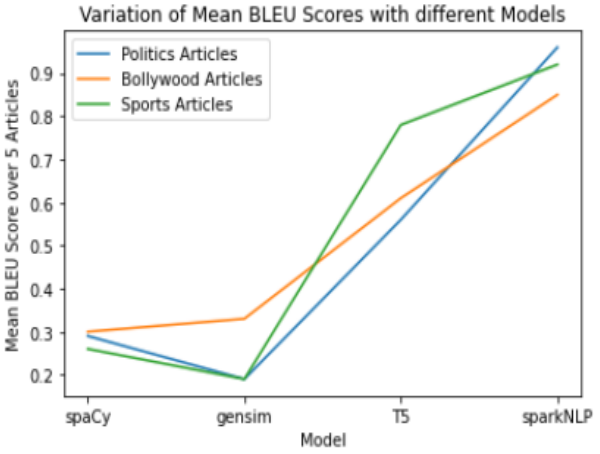

Fig 17. Variation of mean BLEU scores with various models

VII. Fine-tuning models

After reviewing the performances of all the models, we wanted to see if we are able to improve the performance of our simpler models, spaCy and gensim. We decided to fine-tune the models by inputting the original summaries' sentence length and word count, depending on the length argument that the model required.

VII.I Fine-tuning spaCy model

We wanted to see if we can improve the performance of the spaCy model by varying the length of summary generated by the model. The previous spaCy model generated summaries at 30% of the text, which corresponds to an input of 19 for political articles, 6 for Bollywood entertainment articles, and 3 for sports articles for the sentence length argument in the nlargest function. In this experiment, we used the average sentence length of the original summaries, which is at 3 sentences, to see if it affects the performance of the model.

	Politics	Entertainment	Sports
Mean Rogue-1 F1 Score	0.32	0.29	0.34
Mean BLEU Score	0.29	0.30	0.26

Fig 13. Mean Metric Values for fine-tuned spaCy

VII.II Fine-tuning gensim model

We also performed additional experimentation by varying the length of summary generated by the gensim model. Since gensim summarize function does not take in sentence length, we inputted 30% of the original article's word count in the word_count argument on the previous gensim model. These values were 281 words for political articles, 95 words for Bollywood entertainment articles, and 101 for sports articles. Our gensim fine-tuned model follows the average word count of the original summaries. This number

corresponds to 58, 56, and 59 words respectively for political, entertainment, and sports articles.

	Politics	Entertainment	Sports
Mean Rogue-1 F1 Score	0.34	0.27	0.46
Mean BLEU Score	0.19	0.33	0.19

Fig 14. Mean Metric Values of fine-tuned gensim

- Analysis through ROUGE Score

  The F1 Score increased across all topics for both spaCy and gensim fine-tuned models. This may be due to how ROUGE F-1 is calculated since the metric relies not only on the model capturing as many words as possible (recall) but doing so without outputting several words (precision). This showed that comparing a longer summary with a shorter original summary will not work well for this metric.

- Analysis through BLEU Score

  On the other hand, the BLUE scores on both fine-tuned models did not have a significant improvement. It actually performed worse on certain topics. This is especially prominent for fine-tuned gensim models generating summaries on political and sports articles. This might be caused by a greater variation in word count for both articles which may correlate to a higher degree of similarity between the machine summarized text and the reference as there were more word combinations for the metric to compare.

Future Scope

There are several things that could be done to ameliorate and expand our text summarization model. The first thing that we realized after having evaluated our results was that we could work on achieving better accuracy with hyperparameter tuning. Also, several other transformer architectures could be applied to the same problem to see if any architecture excels our T5 sparkNLP implementation in terms of performance. Additionally, we can integrate our text summarization model with a translation model to develop a translation bot.

Appendix

Below, you will find the contribution of each and every member of this group towards the completion of this project.

- Dhanny Indrakusuma - Exploratory data analysis, data preprocessing, implementation of text summarization through spaCy and gensim, experimentation and comparative analysis of spaCy as well as gensim model performance on articles of different topics, fine-tuning spaCy and gensim models, analysis of fine-tuned models, report writing, and presentation.

- Manasvini Karthikeyan - Data preprocessing, implementing text Summarization through gensim, T5 and SparkNLP, comparative performance analysis of T5 and SparkNLP on articles of different genres, report writing, and presentation.

- Shashwat Jyotishi - comparative performance analysis of T5 and SparkNLP on articles of different genres, report writing, and presentation.

Acknowledgment

This paper and the research behind it would not have been possible without the support of our professor, Jyothi Vinjumur. Her enthusiasm, knowledge, and exacting attention to detail have been an inspiration and kept our work on track from the beginning of the project to the final draft of this paper.

We also would also like to thank Kolandarao Vonteru for making the valuable news summary dataset available on Kaggle. His dataset has made it possible for us to conduct our experiments in an efficient manner.

References

[1] C. Khanna, “Text pre-processing: Stop words removal using different libraries,” *Medium*, Feb. 10, 2021. https://towardsdatascience.com/text-pre-processing-stop-words-removal-using-different-libraries-f20bac19929a

[2] K. Vonteru, “News summary,” Kaggle, 13-Nov-2019. [Online]. Available: https://www.kaggle.com/datasets/sunnysai12345/news-summary. (Accessed: Apr 15, 2022).

[3] S. JUGRAN, A. KUMAR, B. S. TYAGI and V. ANAND, "Extractive Automatic Text Summarization using SpaCy in Python & NLP," 2021 International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE), 2021, pp. 582-585, doi: 10.1109/ICACITE51222.2021.9404712.

[4] M. Barbella, M. Risi, and G. Tortora, “A comparison of methods for the evaluation of text summarization techniques,” Proceedings of the 10th International Conference on Data Science, Technology and Applications, 2021.

[5] Chin-Yew Lin. “ROUGE: A Package for Automatic Evaluation of Summaries,” In Text Summarization Branches Out, pp. 74–81, Barcelona, Spain. Association for Computational Linguistics. 2004.

[6] Y. Graham, “Re-evaluating Automatic Summarization with BLEU and 192 Shades of ROUGE,” Association for Computational Linguistics, 2015. Accessed: May 12, 2022. [Online]. Available: https://aclanthology.org/D15-1013.pdf

[7] “spaCy · Industrial-strength Natural Language Processing in Python,” *spaCy*, 2015. https://spacy.io/ (accessed Apr 17, 2022).

[8] “gensim: topic modelling for humans,” *radimrehurek.com*. https://radimrehurek.com/gensim_3.8.3/summarization/summariser.html (accessed Apr 17, 2022).

[9] D. M. J. Garbade, “A Quick Introduction to Text Summarization in Machine Learning,” *Medium*, Sep. 19, 2018.

https://towardsdatascience.com/a-quick-introduction-to-text-summarization-in-machine-learning-3d27ccf18a9f#:~:text=Text%20summarization%20refers%20to%20the (accessed Apr 20, 2022).

[10] “John Snow Labs - Spark NLP,” nlp.johnsnowlabs.com. https://nlp.johnsnowlabs.com/ (accessed May 1, 2022).

[11] “Spark NLP: State of the Art Natural Language Processing,” GitHub, May 11, 2022. https://github.com/JohnSnowLabs/spark-nlp (accessed May 1, 2022).

[12] “T5,” huggingface.co. https://huggingface.co/docs/transformers/model_doc/t5 (accessed May 1, 2022).

[13] “T5: a detailed explanation,” Medium. https://medium.com/analytics-vidhya/t5-a-detailed-explanation-a0ac9bc53e51 (accessed May 1, 2022).

[14] “How do Transformers work? - Hugging Face Course,” huggingface.co. https://huggingface.co/course/chapter1/4 (accessed May 1, 2022).

[15] K. N. Dew, A. M. Turner, Y. K. Choi, A. Bosold, and K. Kirchhoff, “Development of machine translation technology for assisting health communication: A systematic review,” Journal of Biomedical Informatics, vol. 85, pp. 56–67, Sep. 2018, doi: 10.1016/j.jbi.2018.07.018.

[16] “spaCy Tutorial - Complete Guide - NLP FOR HACKERS,” NLP-FOR-HACKERS, Mar. 28, 2018. https://nlpforhackers.io/complete-guide-to-spacy/ (accessed May 1, 2022).