UNIVERSITY OF HERTFORDSHIRE

SCHOOL OF PHYSICS, ENGINEERING AND COMPUTER SCIENCE

MSc in Data Science and Analytics

7COM1039-0109-2022

Advanced Computer Science Master's Project

Effect of Pre-Trained Models in Text Summarization for proceedings

Name : Aishwarya John Pole Madhu

Student ID : 19059835

Supervisor : Christoph Salge

MSc Final Project Declaration

This report is submitted in partial fulfilment of the requirement for the degree of Master of Science in Data Science and Analytics at the University of Hertfordshire (UH).

It is my own work except were indicated in the report. I did not use human participants in my MSc Project.

I hereby *give* permission for the report to be made available on the university website provided the source is acknowledged.

ABSTRACT

Text summarization is a technique that creates a shorter variation of a given text which includes only the essential data. This could be useful for various uses, such as giving a brief synopsis of a lengthy document or making a legal proceeding more accessible to the general public. Traditional techniques for text summarization typically depend on manual techniques, like identifying essential sentences and paragraphs. Nevertheless, these techniques can be time-consuming and prone to error. Pre-trained language models have emerged as a promising new text-summarization technique. The methods were trained on an enormous collection of text and code, allowing them to discover the statistical relationships between terms. Pretrained approaches could be utilized to generate more accurate as ell as informative summaries of proceedings than traditional methods. This research focuses on evaluating the Models that are pretrained for proceedings text summarization. The dataset was obtained from the Old Bailey website, which contains archival records of criminal trial proceedings, for this purpose. The data is pre-processed and utilising trained approaches such as BART, BERT, T5 and GPT-2, and evaluated utilising Rouge – 1, Rouge – 2 and Rouge – L evaluation metrics for a single sentence and also for all sentences. Through evaluation, it can be inferred that the pre-trained models BERT and GPT-2 have nearly identical maximum precision of 100% and 99%, recall of 18%, and f1 score of 31%. This suggests that the BERT as well as GPT-2 models are effective at generating summarised text that is similar to the text in the Old Bailey court proceedings in terms of the words used for one sentence and for all sentences. However, both models are ineffective at generating summarised text that closely resembles the text in the Old Bailey court proceedings in terms of word order.

Keywords: Text Summarization, Pre-trained models, Rouge Score, BART, BERT, T5 and GPT-2.

ACKNOWLEDGEMENT

I want to take this chance to thank Christoph Salge, who is my supervisor. He gave me a lot of motivation to finish my dissertation. And I owe him a huge debt of gratitude for his insightful counsel and unwavering support throughout the writing of my dissertation. Given that the topic of study is new to me, I also want to thank him for his exceptional patience and understanding.

I also want to express my gratitude to my family for always being there for me and for having the utmost faith in me. I want to thank myself for persevering in the end.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

1.1 Text Summarization and its Applications

Summarizing a text is the concept of taking the unique parts of a source and putting them into a shorter version for a specific person and job. Text summarization turns the original text into a shorter version that still has the same information and meaning. Text summary has grown into an important way to understand text information because it contains so much information (El-Kassas et al, 2021). Text summarization could be done in two methods: extractive summarization as well as abstractive summarization. As part of an extractive summarizing method, high-ranking sentences from the text are determined by their individual word along with phrase features, and the sentences were combined to make a summary. Statistical and linguistic traits of words are used to figure out how important they are. An abstractive summary is used to figure out what the main ideas are in a text and then to explain them in simple, natural language. It is a useful technique for various applications, such as news aggregation, document analysis, content generation, and information retrieval. In NLP, Text Summarization models automatically shorten documents, papers, podcasts, videos, and more into their most important soundbites (Allahyari, et al, 2017).

1.2 Pre-trained models for Text Summarization

In recent years, pretrained language models have become an important tool for obtaining remarkable advancements of natural language activities. These algorithms surpass the capacity of basic word embedding by acquiring contextual descriptions using large-scale corpora using a language simulation aim. (Liu and Lapata, 2019).

ELMo, GPT, and BERT are just a few examples of recent context encoders that have found widespread use in various NLP applications. Because these models are trained on a large unlabeled corpus, they are able to produce more accurate contextualised token embeddings, allowing methods that leverage them to perform more effectively (Duan et al, 2020).

BERT is an extremely big database that represents both words and sentences together. Adjustments can be made. After experienced trained on large amounts of material for the unsupervised objective of masked language model with next-sentence estimation, the transformed model is then used for a variety of task-specific purposes. Pretrained models of language were frequently employed as encoders for difficulties in natural language comprehension at both the paragraph and sentence levels, where a wide range of task classification is often involved (Zhang et al, 2019). Word embeddings that take into account the context in which they are used are what ELMo produces. Text categorization, sentiment analysis, and named entity recognition are just some of the downstream activities that frequently make use of this feature extraction tool. OpenAI's GPT models are a family of generating speech and text systems. Models like GPT-2 and GPT-3 aim to produce answers to questions using text that is both logical and appropriate to the topic at hand. Text generation, completion, and even summarization are all areas where GPT-2 and GPT-3 have shown promise (McGuffie, and Newhouse, 2020).

To improve results on language understanding tests, pretrained language models are sometimes utilised. In an effort to solve a number of generation difficulties, pretrained models have lately been applied. Further task-specific variables are adjusted in tandem with BERT's default values, but in ELMo they are often fixed (Khandelwal et al, 2019).

1.3Various Pre-trained models for Text Summarization

The field of language modelling has seen a plethora of transform-based models, such as BERT, BART, GPT, T5.

BERT: To create deep bidirectional representations using unlabeled text, BERT was designed, in contrast to current language representation models. To achieve this, the entire model layers are conditioned simultaneously on both sides of the contexts (Devlin, et al, 2018). This means that the already trained BERT model can be fine-tuned with just one more output layer to build state-of-the-art models to a wide range of tasks, including questions response and a language deduction, with no requiring major modifications to the structure needed for the specific task. The BERT model is

not only easy to understand theoretically but also powerful empirically (Dor, et al, 2020).

BART: The focus of BART is to promote the growth of more natural writing, making it a variant of BERT. This apparatus is composed of a bidirectional encoder with an auto-regressive processor (Lewis, et al, 2019).

GPT-2: The GPT-2 transformers model is a pre-trained model that is automatically trained on raw texts without any human labelling and that inputs and labels are also generated automatically (Kumar, et al, 2020).

T-5: T5 stands for Text-To-Text Transfer Transformer (5 Ts). The T5 transformer model is functional because it employs the tried-and-true encoder-decoder framework of previous transformer models. The encoding and decoding are done in blocks of 12 pairs. Self-attention, a feedforward network, and encoder-decoder attention are all built into each individual building block (Liu, et al, 2021). As a result of the large amount of data it is pre-trained on, the enhanced scalability of model parameter values, and the model's simple applicability to a wide variety of tasks thanks to its generative nature, T5 has grown more popular than previous architectures like BERT (Yang, et al, 2021).

1.4Problem statement

The purpose of text summarizing is to mechanically produce a brief abstract of a given text material. Without reading the whole thing, this may assist readers get the gist of what the material is about (Kumar, et al, 2021). The use of pre-trained algorithms for text summarizing has grown in popularity in recent years due to the fact that they may increase the precision and efficiency of summarization tools. Pre-trained models are models that are developed using large datasets and are trained to produce a certain output (Zhang, et al, 2019). The current limitations of text summarization in proceedings are a major challenge to effectively understanding the main points of a text document. However, pre-trained models are often limited to large documents and struggle to accurately summarize new proceedings with both small and large documents (Fabbri, et al, 2019). Therefore, it is important to study how well pre-trained models perform when applied to Old Bailey proceedings, as well as to assess their performance when applied to proceedings of different sizes. The purpose of this

research is to examine the usefulness of pre-trained models in this context in text summarization for new proceedings, and evaluating how well different pre-trained algorithms can summarize the content of Old Bailey hearings. Additionally, the study aims to analyze the effects of the size of the article on the accuracy of the model, as well as to determine the effectiveness of pre-trained models for summarizing new proceedings. The results will be evaluated using evaluation metrics such as accuracy, precision, recall, F1 score and Rouge score.

1.5Aim

The purpose of this research is to examine how well-known pre-trained models (such as BERT, BART, GPT2, and T5) perform when applied to Old Bailey proceedings and to study the performance of the models when applied to smaller and bigger proceedings.

1.6Research question

RQ 1: Does the BERT pre-trained models more effective than other text summarization models such as BART, T5 and GPT for Old Bailey court proceedings?

1.7Research hypothesis

Hypothesis 1: Pre-trained models are more effective for summarizing long proceedings than short proceedings than long proceedings.

1.8Objectives

The research objectives are,

- Evaluate how well various pre-trained models do at summarizing texts of Old Bailey proceedings and other proceedings.
- To analyze the effects of the size of the article on the accuracy of the model.
- To determine how well-suited pre-trained models are for summarizing new proceedings.

1.9Current Issues

“Text summarization has been a crucial problem in natural language processing (NLP) for several decades. It aims to condense lengthy documents into shorter versions while retaining the most critical information.” (Yang et al, 2023)

“Recent summarization methods based on sequence networks fail to capture the long-range semantics of the document which are encapsulated in the topic vectors of the document.” (Joshi et al, 2023)

“The volume of information is increasing at an incredible rate with the rapid development of the Internet and electronic information services. Due to time constraints, we don't have the opportunity to read all this information. Even the task of analyzing textual data related to one field requires a lot of work. The text summarization task helps to solve these problems.” (Madatov et al, 2023)

1.10 Plan to conduct the research

The plan for this study will involve a quantitative approach to assess pre-trained models' usefulness in practice in text summarization for new proceedings. Specifically, the research methodology will involve the following steps:

Data Collection: The proceedings will be of varying sizes, ranging from short proceedings (less than 500 words) to long proceedings (over 2000 words).

Pre-trained Model Selection: Pre-trained models such as BERT, BART, GPT2 and T5 algorithms will be selected for this study.

Model Training: The gathered data will be used to further hone the pre-trained models.

Evaluation and Analysis: The accuracy, loss and performance of the pre-trained models when applied to Old Bailey proceedings and other proceedings of varying sizes will be evaluated and analyzed.

Results: The results of the study will be reported and discussed.

This study will provide valuable insights into the efficiency of text summarization using pre-trained models of new proceedings.

1.11 Feasibility

Research on text summarization using pre-trained models for Old Bailey hearings in Python is feasible. The goals of the study and Python's capacity for natural language processing tasks are a good fit. The plan for data gathering can be implemented using Python's data manipulation and parsing modules, which entails manually collecting Old Bailey proceedings of varied sizes. Python's strong support for pre-trained models means that a number of widely used models, including BERT, BART, GPT2, and T5, can be easily implemented. Python's machine learning libraries make light work of the training process, allowing for precise adjustments to be made to the acquired data using the selected pre-trained models. Python's packages for metric computation and visualisation make it simple to implement evaluation and analysis, including accuracy, loss, and performance metrics. Using Python's data visualisation tools, the final results and comments may be displayed clearly. The research's implementation in Python seems realistic and beneficial, given that language's rich environment for NLP and ML.

1.12 Project Plan

Task ID	Effect of Pre-Trained Models in Text Summarization for New Articles	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9	Week 10	Week 11	Week 12	Week 13	Week 14	Week 15
T01	Backround Reseacrh															
T02	Choosing the Research Title & Dataset															
T03	Framing RQ, Aim, Objectives and Methodologies															
T04	Literature review															
T05	Dataset processing															
T06	Importing Dataset and Packages															
T07	Handling null values															
T08	Handling duplicate values															
T09	Label Encoding															
T10	Implementing BERT algorithm															
T11	Implementing BART Algorithm															
T12	Implementing GPT 2 Algorithm															
T13	Implementing T5 algorithm															
T14	Conclusion Drawn															
T15	Conclusion															
T16	Addressing Research Questions															
T17	Documentation															

1.13 Outline

The first chapter provides a comprehensive introduction to text summarizing and its applications; it also introduces a variety of pre-trained models for text summarization and outlines the study's goals, hypotheses, and challenges. The second part of the approach is devoted to a literature review on the subject of text summarizing. Chapter 3 discusses how to take into account social, ethical, legal, and professional concerns, as well as providing a comprehensive overview of methods and how they are evaluated. In the fourth chapter, the implementation process and its outcomes are analyzed in depth. The final chapter provides a synopsis of the study's analysis, findings, conclusion, limitations, and suggestions for further research.

2 BACKGROUND RESEARCH

Summarizing is shortening a longer document while keeping its main ideas and structure intact. This technique can be useful in a variety of contexts, including studying, research, and communication. Summarizing a text can help break down complex ideas and make them easier to comprehend. It may help people save time since they don't have to read the complete text to acquire the important elements (Fabbri, et al, 2019). The main advantage of text summarization is that it enables readers to quickly get the main points of anything without reading the whole thing. This is especially useful when dealing with lengthy texts such as research papers, articles, and books. Summarization can make it easier to understand a text's main points and to identify important details. In addition, the time spent reading may be cut in half with a good summary (Liu, et al, 2019).

2.1 Need for text summarization in court proceedings

Text summarization is becoming increasingly important in court proceedings. It is a process of automatically reducing a document to its most important points. To do this in just a small percentage of the time it takes to read and absorb the complete document, a succinct and precise summary of the content is required. Text summarization is especially useful in courts because it can help reduce the amount of time spent on tedious document review (Kanapala, et al, 2019). Lawyers and judges need to review vast amounts of material in a short amount of time, and text summarization can help them quickly identify the most important points in a document. This can help them to focus on the most relevant facts and arguments, and quickly identify any inconsistencies or inaccuracies in the document.

Text summarization is also beneficial in court proceedings because it can help the parties involved in a case better understand the opposing side's arguments. By summarizing the opposing party's arguments into concise points, both sides can gain a better understanding of the other's position. This can lead to more efficient and effective negotiations between the parties, and ultimately result in a better outcome (Keneshloo,

et al, 2019). Text summarization can also be used to help identify key evidence in a case. By summarizing a document, lawyers can quickly identify any relevant evidence that could be used to support their argument. This can be especially helpful in cases involving large volumes of evidence. Overall, text summarization is an invaluable tool for court proceedings. It can help reduce the time spent on document review, provide a better understanding of the opposing party's arguments, and identify key evidence in a case. As text summarization technology continues to improve, it is likely to become even more important in court proceedings in the years to come (Kouris, et al, 2022).

2.2 Pretrained models

Pre-trained models for text summarization are models that had been trained on a huge body of data before being applied to a particular summarising job. Pre-trained models are particularly useful for text summarization because to summarise well, one must be fluent in the language and culture of the original material (Zhang, et al, 2019). Pre-trained models are able to capture the underlying semantics of the text and are more accurate than models that are trained from scratch. There are normally two parts to every pre-trained model for summarizing text: an encoder and a decoder. The encoder works to understand the structural and semantic features of the text, such as word order and relationships between words, while the decoder works to generate a summary of the text (Miller, et al, 2019).

2.3 ML and DL in text summarization

The purpose of the article by Alami et al. (2019) is improve automated text classification (ATS) by combining unconstrained deep neural network methods with a word embedding approach. Word2Vec format is shown to be superior to the more popular BOW model, and therefore the authors begin by developing a text summary approach dependent on word embeddings. Second, the authors provide supplementary models that include data from several sources by using a mixture of word2vec and unsupervised feature learning. In this work, the authors show that Word2Vec-trained unsupervised NN models outperform BOW-trained models. Finally, authors suggest three different ensemble methods. The first ensemble uses a majority vote approach to

combining BOW and word2vec. Information from the BOW method and unsupervised NN are combined in the second ensemble. Thirdly, authors have an ensemble that takes the data authors have gotten from Word2Vec and unsupervised neural networks and combines them. Authors demonstrate that ensemble approaches enhance ATS quality, and in particular, that an ensemble based on the word2vec approach produces superior results. Lastly, authors assess the models' effectiveness using a variety of experiments.

Goularte et al (2019) approach to using a minimal set of fuzzy rules, we can summarise texts has the potential to be used in the creation of expert systems that can autonomously grade written work in the future. The suggested technique of summarizing has been taught and evaluated in trials utilising a dataset of texts written in Brazilian Portuguese that was submitted by students as a response to assignments that were assigned to them in a VLE. The suggested technique was evaluated with a number of different approaches, such as a naïve baseline, Score, Model, and Sentence, with the assistance of ROUGE measurements. A summarization system's effectiveness may be measured with the help of the Rouge metric. A system's ability to properly extract important information from a text is evaluated using precision and recall. Metric Rouge calculates an overall score based on a number of parameters, including the summary's length, the degree to which it overlaps with the reference summary, and other considerations. The findings indicate that the proposed technique yields a more accurate f-measure (with a confidence interval of 95%), in comparison to the methods described before.

In Song, et al, (2019) study, the authors provide an LSTM-CNN based ATSDL that may generate novel sentences by delving into smaller units than sentences, such as semantic phrases. This allows for the construction of new sentences. In contrast to other methods that are based on abstraction, ATSDL is made up of two primary phases: the process begins with the identification of target phrases in the input sentences, and the latter of these stages provides text summaries via the use of deep learning. The ATSDL framework beats the frameworks within the context of meaning and syntax, and it obtains successful performance on manual linguistic quality assessment, as shown by experimental findings on the CNN and DailyMail datasets. This project used ROUGE score to evaluate the algorithms.

Table 1 Gist of Literature survey conducted

Author	Objective	Parameter used	Result Achieved
Alami et al.	Improving text summarization using unsupervised neural networks by including word embeddings and ensemble learning	Word Embedding and Ensemble Learning	Improved precision and recall by 8.72% and 27.08%, respectively.
Zhang et al.	Text summary using MLP and pre-training	Pretraining-based natural language generation	Improved ROUGE scores of up to 4.57%.
Goularte et al.	An approach to text summarising using fuzzy rules for use in machine evaluation	Fuzzy Rules	High accuracy of 98.50%.
Song et al.	Deep learning-based LSTM-CNN abstract text summarization	LSTM-CNN based DL	ROUGE-1, ROUGE-2, and ROUGE-L scores of 0.44, 0.21, and 0.41, respectively.
Ramesh et al.	Abstractive Text Summarization Using T5 Architecture	T5 Architecture	ROUGE-1 and ROUGE-2 scores of 0.72 and 0.51, respectively.

2.4Pre-trained models for text summarization

In Zhang, et al, (2019) research, authors offer a new encoder-decoder architecture that is based on pretraining and features a two-stage process that produces an output sequence based on an input sequence. In order to transform the input sequence into representations of context, the authors of this model employ the encoding method BERT. There are two stages of the design that focus on the decoder. To begin, an initial output sequence draught is generated using a Transformer-based decoder. Step two involves writers hiding individual words in the draught sequence before sending it to BERT. The authors then use the BERT-provided draught representation in conjunction with an input Transformer-based decoder to make predictions about the improved word

at each masked point. The methodology is, as far as authors are aware, the first way that integrates the BERT into the process of text creation jobs. In order to take the initial step in this direction, authors will assess the suggested approach by applying it to the text summarizing problem. The results of our experiments indicate that our model reaches new levels of excellence using information from both the CNN/Daily Mail and New York Times datasets. The algorithms were graded using the ROUGE score in this assignment.

2.5 Summarization using T5 technique

Today, especially on the internet, there is a plethora of text due to the proliferation of information and communication technologies. The text must be summarized so that it is easier to read and understand without losing the essential meaning or context. In order to quickly and easily find the most relevant and important information in a large body of text, automatic text summarization is a useful tool. In this paper, authors propose a T5 and NLP-based text summarization model that can grasp the big picture, extract the most vital information, and produce coherent summaries. The algorithms were graded using the ROUGE score in this assignment (Ramesh, et al, 2022).

Ghadimi et al. (2022) provide HMSumm, a method for abstractively summarizing the contents of several documents. The proposed method combines extractive summarizing with abstractive summarization. The abstractive summary is created by first creating an extractive summary from the source materials. As a first step, writers deal with duplicate information, a common challenge in summarizing many documents. The DPP is used to avoid repetition. At this point, we may also choose a maximum length for the input sequence that will be used in the abstractive summarization procedure. This may go one of two ways. The major motivation is to save time throughout the computation process. Second, an abstractive summarizer must retain the most important parts of the texts it is given to summarize. The authors assess the extracted summary phrases using a deep submodular network (DSN), and they determine redundancy using BERT-based similarities. By feeding the collected extractive summary into pre-trained models, two abstract summaries are created (BART and T5). The number of words in an abstract summary helps authors choose the best one. The performance of HMSumm is

compared to that of several other methods, including evaluations from both humans and ROUGE. The algorithms are evaluated using the DUC 2002, DUC 2004, Multi-News, and CNN/DailyMail datasets. Experiments show that HMSumm outperforms competing algorithms.

2.6 GPT2 for summarization

Kieuvongngam, et al, (2020) states that in light of the current COVID-19 pandemic, it is crucial for the medical community to keep up with the ever-increasing volume of published research on coronaviruses. Because of this disconnect between researchers and the ever-increasing volume of publications, the COVID-19 Open Research Dataset Challenge has released a dataset of scholarly articles and is calling for the use of machine learning techniques to help bridge the existing knowledge gap. Authors address this difficulty by conducting text summarization using two pre-trained NLP models, BERT and OpenAI GPT-2. ROUGE scores and visual examination are used in our analysis. The algorithm, using keywords pulled from the source articles, gives abstracted and detailed data. To aid the medical community, the effort will provide concise summaries of works for which an abstract is not yet accessible.

Rinse, et al, (2019) research paper takes an extractive and an abstractive approach to automating text summarizing. The former strategy makes use of submodular components and the BERT model of linguistic representation, whereas the GPT-2 model is used in the latter. Authors use two distinct kinds of datasets in this work: the CNN/DailyMail dataset, which is a standard for news article datasets, and the Podcast dataset, which is made up of transcripts of podcast episodes. On the CNN/DailyMail dataset, the GPT-2 achieves results that are on par with methods. A qualitative study, in the shape of a human evaluation, is also conducted by the authors, and they evaluate the trained model to show that it learns plausible abstractions, in addition to the quantitative evaluation.

Reading the news in little doses, understanding it "from long to short," and quickly, properly, and thoroughly getting important information are all aided by employing an automatic text summary, as stated by Yang et al (2021). The GPT2 model is used to power the authors' automatic summarizing technique and modify the loss calculation

section of the GPT2LMHeadModel included in the transformers package. The authors then conduct tests on both the original news information as well as the split-word data, collecting data for both 5 and 10 epochs, respectively, and evaluate the results with the ROUGE evaluation value. The measurements for both Rouge-1 and Rouge-2 have significantly increased when compared to previous studies. This model is an improvement on the BERT benchmark model's representation of the summary text, which, in the context of automated extraction of news headlines, is insufficient and inaccessible.

2.7 Research gap

Despite the recent advancements in the discipline of summarizing text, there is still a lack of research focused on the effects that pre-trained models have on the summarization of real time court proceedings. However, few studies have explored the use of pre-trained models such as BERT, GPT-2, and T5 in text summarization in various fields, there is a lack of implementation of the techniques in court proceedings. These models have already proven useful for a variety of NLP applications, but their effectiveness for text summarization has yet to be adequately assessed. Thus, in this research, the gap of using pre-trained models on the summarization of real time court proceedings is carried out in detail. Another aspect of identifying if the size of the article changes the accuracy obtained by the models is also studied in detailed which was not explored in previous researches. The results will be evaluated using evaluation metrics such as accuracy, precision, recall, F1 score and Gouge score.

2.8Linkage to Aims

The findings of this study will provide an understanding of the effects of pre-trained models on the summarization of proceedings. This understanding can be used to develop more effective summarization methods that can be used to quickly and accurately summarize proceedings. Furthermore, the results of this study can be used to develop new techniques and strategies for summarizing proceedings that can be used in a variety of applications. Additionally, the findings of this study can be used to

develop more accurate algorithms for summarizing proceedings that can be used in a variety of domains.

3 METHODOLOGY

3.1 Choice of Methods

The below advantages of these pretrained models that includes BERT, BART, GPT2, and T5 make them suitable for text summarization task.

BERT: It considers phrase from both the previous and subsequent words. This is useful for figuring out how various words relate to one another semantically (Clark, et al, 2019).

BART: - Encoder-Decoder Architecture BART is well-suited for sequence-to-sequence tasks like text review since it is built on an encoder-decoder architecture. This framework allows it to provide well-organized summaries from the input (Soper, et al, 2021).

GPT- 2: The context of an input text is efficiently captured by GPT2's Transformer design, which is the third major component of GPT2. As a result, readers gain a deeper comprehension of the material and better summaries are produced (Liu, et al, 2021).

T5: T5's ability to produce high-quality summaries with consistent output format stems from its treatment of summarising as a text-to-text problem. T5 has been pretrained on a large variety of activities, which greatly increases its flexibility. T5's extensive linguistic knowledge enables it to provide clear and useful summaries from a wide range of input texts (Singh, 2020).

3.2 Research Methodology

In this research method, evaluates pre-trained models for text summarization of Old Bailey hearings. Methodology involves a series of distinct stages:

1. Data Collection: The Old Bailey website provided the data for this set, which contains 100 records detailing court proceedings from 1900 to 1910.

2. Pre-trained Model Selection: Text summarization models with a track record of success are selected, and these include the BERT, BART, GPT2, and T5 models.
3. Model Training: To improve the algorithms' ability to summarise Old Bailey proceedings, they are fine-tuned on the dataset.
4. Evaluation and Analysis: Metrics like as accuracy, loss, precision, recall, F1 score, and Gouge score are utilized to assess efficiency in numerous contexts.
5. Results and Discussion: Pre-trained models' strengths, shortcomings, and potential for development in summarising Old Bailey hearings of varying lengths are shown and discussed.

3.3 Tool Required

Hardware Tools required are computer with enough disk space.

Software Tools required are Anaconda Navigator with Jupyter Notebook Platform.

Programming Language required are Python programming version greater than 3.7.

3.4 Bailey Court Proceedings Dataset

The dataset utilized for the research is downloaded from the Old Bailey website. It comes as an XLSX file, with 100 rows and 4 columns. This dataset includes the following columns: -

- Date: This column includes the date when the bail procedures took place.

- Link: The URL or link to the relevant case on the Old Bailey website can be found in this column.

- Category: The category or classification of the case is listed in this column, which provides information on the nature of the proceeding being tried.

- Proceedings: This column contains the text of the proceedings, which has data on the trial like the names of the parties involved, the charges against them, the jury's verdict, and the judge's punishment.

It is a rich source of knowledge that may be utilised for researching and comprehending the social, cultural, and legal history of the time period from 1900 to 1910.

3.5 Data Pre-Processing

The purpose of data preprocessing is to prepare the data so that ML algorithms can more effectively learn from it (Yang, 2018). Preprocessing used in this research consists of following steps:

Handling null values: In order to prevent machine learning models from being corrupted, null values should be removed. Handling null values can be done in a few different methods, including:

- Imputing: It entails substituting guessed values for missing information in place of nulls.

- Deleting: When performing a delete, any rows or columns that do not contain data will be discarded (Li, et al, 2020).

Removing unwanted attributes: Some features may not be important to the machine learning task at hand, and removing them can boost the model's performance (Maharana, et al, 2022).

3.6 Pre-Trained Models

A pre-trained model (PTM) is an approach that has already been trained to solve a similar problem using a big standard dataset. The primary goal is to acquire a general, latent model of a language from a typical task only once and then use it in different NLP tasks. Language modelling is the general goal, and there is a lot of self-supervised text for training (Min, et al, 2021).

3.6.1 BERT: Bidirectional Encoder Representations

A deep bidirectional transformer, BERT learns language from text. Pre-trained models have improved greatly. The Transformer-based bidirectional encoder is described in this module. It conditions the right and left contexts of all layers using two-way representations of unmarked text (Qu, et al, 2020). Some NLP tasks can be BERT-compatible with pre-training and fine-tuning. It performs two self-training tasks to improve computer language comprehension. MLM, SOP, and sentence prediction are pre-training tests. Binary categorization is used to determine if two words make sense for this job. Three embedding types are supported by BERT. Token, positional, and segment embeddings (Han, et al, 2021). The encoder layers focus on the scaled dot-product with numerous heads after the input layer. Fully linked layers are added to stackable encoders' final layers. To fine-tune NLP tasks like answering questions, BERT replaces the fully-connected layer with the correct layer (Kim, et al, 2022). The unique token can be used for sequence labelling or query response, and a further layer for categorization (Muller, et al, 2023). The BERT architecture is given below.

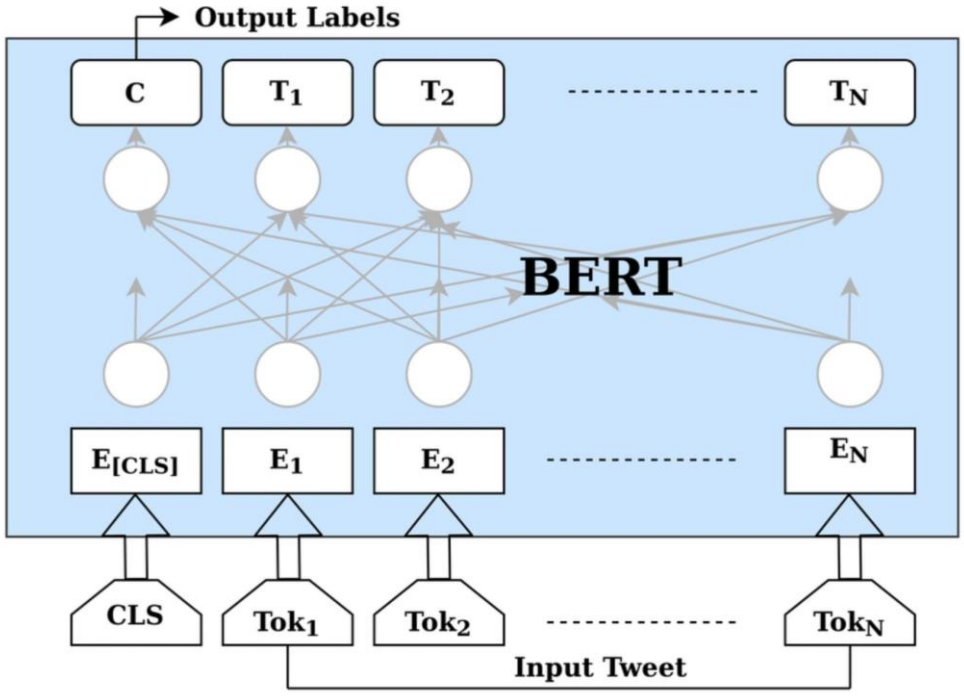

Figure 1 BERT model (Gundapu, et al, 2021)

The BERT model allows for the usage of greater quantity of unannotated data, that significantly elevates the accuracy of the model. It produces word representations that

are dynamically informed by the words around them and is designed to generate high-quality representations of text which could be utilized for a wide range of NLP tasks. The accuracy of the model is effective as it is regularly updated and metrics are fine-tuned and used immediately (Al-Ghamdi, et al, 2023).

3.6.2 BART: Bidirectional Auto-Regressive Transformers

BART is a change to BERT that focuses on making smooth writing. It has an encoder that works in both directions and a decoder that works backwards. Also, the training process starts by making the input text noisy with functions like erasing and blocking before trying to piece together the original text in order (Lewis, et al, 2019). Since it has a decoder, it can be taught to do many things, such as token masking, content infilling, token recognition, phrase arrangement, and document rotation. BART works best when it is fine-tuned for text generation, but it also does well when it is used for reading tasks (Muster, et al, 2020). The BART Model is shown below.

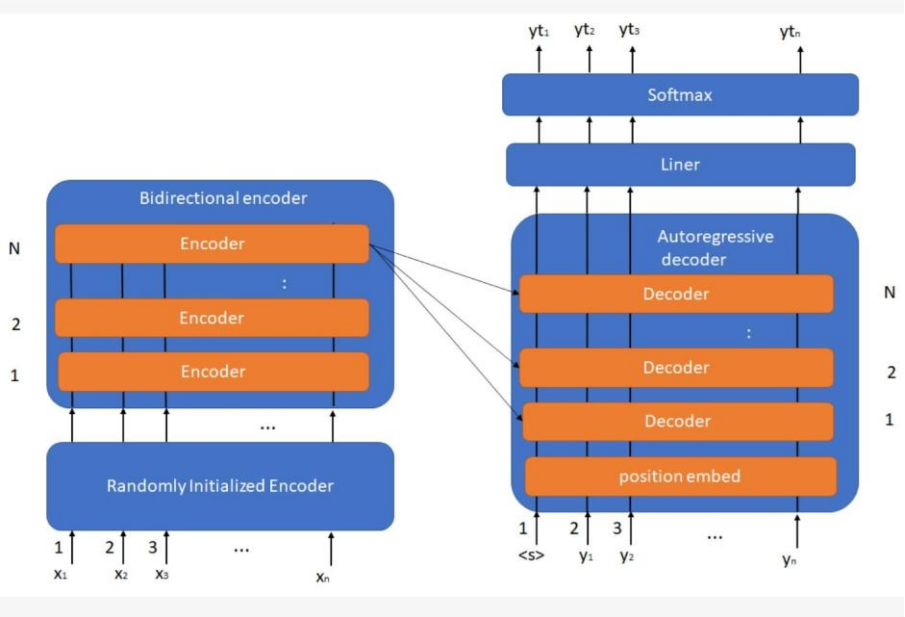

Figure 2 BART Model (Alokla, et al, 2022)

BART is better for text generation. It is a noise reduction auto encoder for initial training sequence-to-sequence models. It can do any NLP job easily, like summarizing, machine translation, putting words from raw text into groups, or answering questions

in the real world. It makes subtitles that are more accurate and relevant to the situation. This makes predictions more accurate (Ou, et al, 2022).

3.6.3 GPT2: Generative Pretrained Transformer 2

GPT2 is the first Transformer model with self-supervised pre-training. The decoder-only transformer component makes GPT-2 (Lee, et al, 2020). Only one token is sent out by the model using the set starting token. Every token is added to the end of the previous tokens. This sequence becomes the model's next phase's data. After setting the length or flag, the model sends out a generation sequence and stops creating text. The simple diagram of GPT-2 structure is shown below.

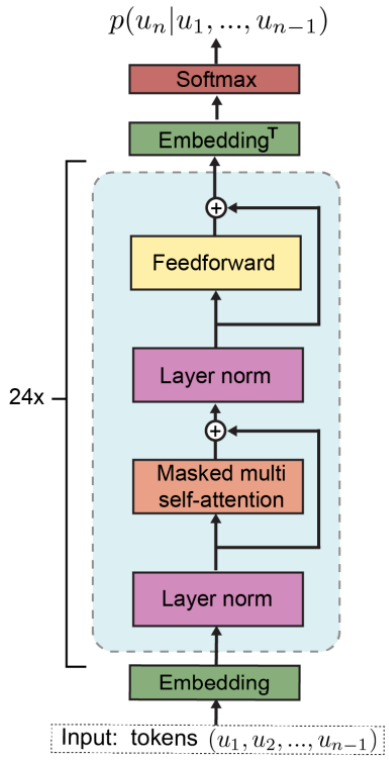

Figure 3 GPT2 architecture (Heilbron, et al, 2019)

The major job when employing this model is to make sentences determined by the first words. The important goal of the method is to make the subsequent phrase in a loop that depends on a series of available text inputs until it hits a length limit or an end

marker. The GPT2 is a one-way model that uses training to ddetect the left-to-right context of the future (Liu, et al, 2021).

3.6.4 T5: Text-to-Text Transfer Transformer Evaluation

The fundamental concept for T5 is that it initially pre-trains the model on a big, general dataset employing a self-supervised condition, such as putting in missing words in the particular sentences. When it has been pre-trained, it can be fine-tuned on more specialized datasets, each of which is linked to a specific job like translating languages or classifying sentences (Mastropaolo, et al, 2021). The structure of the T5 model is depicted below.

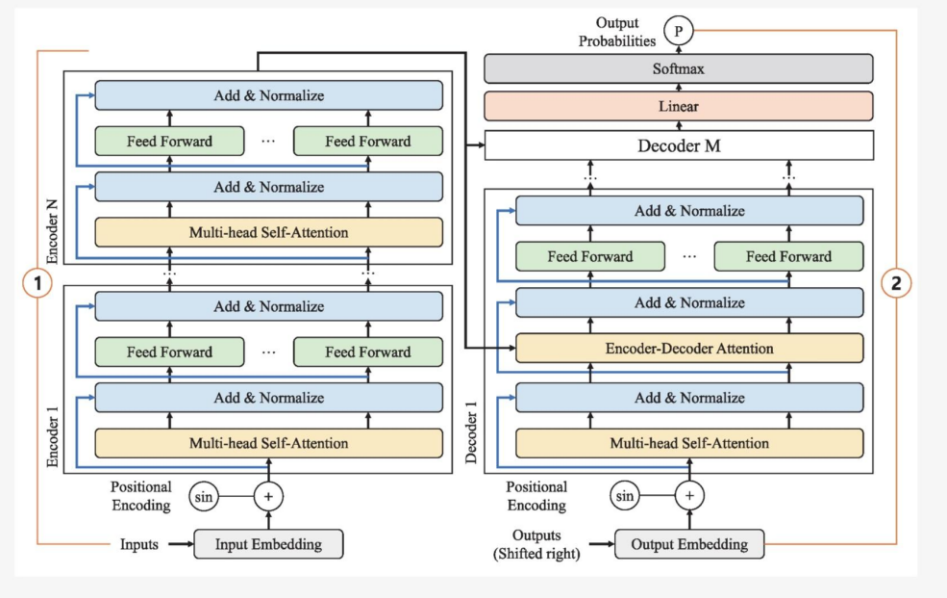

Figure 4 T5 architecture (Hwang, et al, 2023)

The transformer model allows the T5 model handle different-sized inputs with stacks of self-attention levels. A chain of inputs is converted into embeddings and passed to the encoder. The self-attention layer and negligible feed-forward network of each encoder are the same. Each subcomponent's input is layer normalised and appended to its output using a residual skip link. Dropout regulates the skip link, stack input, and output in the feed-forward network (Nagoudi, et al, 2021). The encoders and decoders work similarly. After every self-attention layer, another attention device examines

encoder output. The last encoder block's result is introduced through a layer that employs softmax output to get language results (Itsnaini, et al, 2023).

3.7Validation

ROUGE Score

Machine-generated text summaries are commonly compared to reference or human-generated summaries using the ROUGE metrics (Recall-Oriented Understudy for Gisting Evaluation). ROUGE analyses how many n-grams (sequences of n words) are present in both the generated summary and the original text (Grusky, et al, 2023).

ROUGE Precision: It evaluates how well the produced review matches the review of literature in terms of include key information. It determines how similar the review of literature is to the reference summary by determining how many n-grams (words or phrases) appear in both (Zopf et al, 2018).

ROUGE Recall: ROUGE Recall evaluates how well the generated summary replicates the review of literature's coverage of the source material. It determines how similar the produced summary is to the review of literature by dividing the number of overlapping n-grams between the two sets of data by the total number of n-grams in the review of literature (Liu, 2019).

ROUGE F1 Score: The ROUGE F1 Score is a comprehensive evaluation of the generated summary's quality that considers both its accuracy and its recall. It determines a single value that considers both the ability to contain relevant content and to cover essential details by calculating the harmony between accuracy and memory recall (Ganesan, 2018).

3.8Considering issues

3.8.1 Social issues

The social value of the study demonstrates its commitment to fair and just results. In order to mitigate potential negative effects, the study explicitly takes into account societal concerns. Care is taken to prevent bias and discrimination, leading to a more welcoming environment. Participants' rights to confidentiality are protected, and their

freedom and permission are emphasised. The research lives up to its social and ethical obligations by strictly following data protection practises and privacy norms.

3.8.2 Ethical issues

In the context of "Effect of Pre-Trained Models in Text Summarization for Proceedings," moral concerns play a central role. The steps used in the research to prevent the misuse of data show a dedication to ethical data usage. Precautions are taken to eliminate the possibility of discriminating or biased outcomes. Strict confidentiality procedures are in place to prevent any outside parties from gaining access to participant information. The study incorporates access restrictions, data minimization, retention policies, and robust security measures to protect against unauthorised access, usage, disclosure, or destruction of data in accordance with stringent data privacy, security, and integrity standards.

3.8.3 Legal issues

The "Effect of Pre-Trained Models in Text Summarization for Proceedings" study focuses heavily on the topic of legal compliance. All data collected and analysed in the study will be done so in accordance with the relevant privacy laws and standards. All the necessary steps are taken to ensure compliance with the regulations, so there will be no legal issues. The study is based on a firm dedication to ethical research practises, and its operations are backed by an ironclad guarantee of legality.

3.8.4 Professional issues

The "Effect of Pre-Trained Models in Text Summarization for Proceedings" study places a premium on honesty and reliability in the field of research. The study's dedication to objectivity and accuracy is demonstrated. Careful work is done to avoid mistakes, track down inaccuracies, and maintain a high level of professionalism. Correct citations and credits are used, protecting against plagiarism and preserving the study's credibility among academics. All of the data and programmes used in the study are properly cited, demonstrating the highest standards of professionalism. The

research is conducted openly and to the greatest standards of quality throughout its entirety, from planning to presentation of findings.

3.9Consideration of Commercial and Economic Context

There are significant commercial and financial implications of the proposed study plan, which will assess the efficacy of summarization models that have already been trained in court processes. The legal industry can improve service quality, streamline processes, and cut costs by implementing cutting-edge strategies like pre-trained models. This uptake helps give businesses an edge, encourages innovation, and opens up new opportunities. Professional development and effective regulation management are two more benefits. Adopting pre-trained models not only improves interactions with clients, but also puts businesses at the cutting edge of technical development, which boosts their profitability, longevity, and standing in the marketplace. Therefore, this research has the potential to reform the legal system by making better use of pre-trained models, thereby supporting economic development and growth.

4 EXPERIMENTS & RESULTS

This research goal to examine the efficacy of pre-trained methods like BERT, BART, GPT2 and T5 approaches in text summarization while applying to Old Bailey proceedings, as well as the models' efficiency when applied to smaller and larger proceedings. In this session, collection of data and its preparation, how the data is applied to the pre-trained methods, and the results of its evaluation using the rouge score are detailed.

4.1 Data Collection

The information for the study comes from the Old Bailey's online presence. Data from historic criminal trials are archived in The Old Bailey Proceedings xlsx at the Old Bailey Courthouse in London, England between 1674 and 1913.

https://www.oldbaileyonline.org/static/Data.jsp

The identities of those involved, along with charges, judgements, and punishments, are all included in the data set. It also includes extra data including defendants' employment, trial dates, and the courthouse's physical address. Researchers interested in the social, cultural, and legal history of the time will find this important dataset beneficial. it provides a rare glimpse into the criminal justice system of the time. There are 100 rows and 4 columns in the data set.

4.2 Data Description

The dataset has been obtained from Old Bailey website, which is in the xlsx format and it contains 100 rows and 4 columns. The data contains date of the bails, link, category and proceedings contents from year 1900 to 1910, totally 100 cases are available. This dataset has been imported using the code as shown below.

```
baile_Proced = baile_ProcedPs.read_excel('Old Bailey Proceedings.xlsx')
baile_Proced.shape
```

```
(100, 4)
```

Figure 5 Identifying shape of the dataset

The dataset contains 9 unique elements in the attribute of ‘Category’. They are 'Breaking Peace', 'Theft', 'Killing', 'Deception', 'Royal Offences', 'Violent Theft', 'Sexual Offences', 'Miscellaneous', 'Damage to Property' and the value counts in the output attribute ‘Category’ described in the following.

```
Theft                36
Royal Offences       18
Breaking Peace       15
Deception            15
Killing               7
Violent Theft         4
Sexual Offences       2
Damage to Property    2
Miscellaneous         1
Name: Category, dtype: int64
```

Figure 6 Identifying output value counts

The most information is found in the 'Theft' category, while the 'Miscellaneous' category holds the least.

4.3Data Cleaning

The data which have been imported has to be checked for finding any null values present in the dataset and if any null values present, the same has to be removed. Finding the null values in the dataset is as described below.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 4 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   Date                100 non-null    object
 1   link                100 non-null    object
 2   Category            100 non-null    object
 3   Proceedings Content  100 non-null    object
dtypes: object(4)
memory usage: 3.2+ KB
```

Figure 7 Description of the dataset and identifying null values

Pre-trained models for text summarization do not include any null data and also inferred that the output column contains object type of data.

4.3.1 Dropping Unwanted Attributes

As a part of preprocessing, the unwanted attributes have to be deleted which do not give any value to the predictions. These proceedings are summarised in text without the 'Date' and 'link' elements. The coding for dropping unwanted attributes are as shown below.

```
##Drop date and link attributes...
del baile_Proced['Date']
del baile_Proced['link']
```

Figure 8 Coding for dropping unwanted attributes

4.4EDA of Pre-trained models

The ‘Category’ column is the output column and before implementing the pre-trained models which category has the highest count and lowest count can be visualized using count plot. The visualization of the count plot is as shown below.

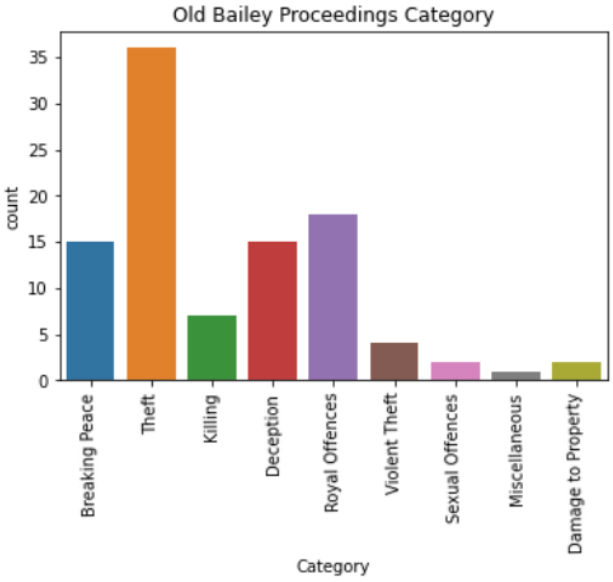

Figure 9 Visualizing output attribute ‘Category’ of proceedings

The most information is concentrated in the 'Theft' category of the above count plot variable. More than 15 integers are stored in the variables "Breaking Peace," "Deception," and "Royal Offences."

4.5Evaluation and Results of Pre-trained Models

For implementing the pre-trained models, the necessary packages such as transformers and rouge-score have to be implemented. Transformers package is used categorization, extraction, question answering, summarization, translation, and synthesis of text are all examples of text-related activities that may be performed. Rouge-score package is an evaluation metric for automatic text summarization. The coding for importing the packages is as shown below.

```
!pip install transformers
!pip install rouge-score
```

Figure 10 Importing transformers and rouge-score package

4.5.1 BART Model in Text Summarization for Proceedings

The dataset now contains 100 rows and 2 columns and the columns are ‘Category’ and ‘Proceedings Content’. In this case, the pre-trained BART algorithm was used to summarise both individual sentences and the entire dataset.

In the column ‘Proceedings Content’, the first entry [0] contains 3114 text data and the pre-trained model is implemented by defining the model's title, with the limit on the abridged version of the content set to 70, tokenizer is utilized for tokenize the input text & encode the data and further Conditional Generation is used for generating the summarization of the text and finally the data is decoded into text format. The following is the code used to implement the BART model for text summarizing.

```
def bart_txt_summarizer(txt_sum_inpt_txt):
  txt_sum_bart_mdl_nm ="facebook/bart-large-cnn"
  txt_sum_bart_mx_lth=70
  txt_sum_toknzr = txt_sum_bart_tknzr.from_pretrained(txt_sum_bart_mdl_nm)
  txt_sum_bart_mdl = txt_sum_bart_cg.from_pretrained(txt_sum_bart_mdl_nm)
  txt_sum_bart_inputs = txt_sum_toknzr.encode("summarize: " + txt_sum_inpt_txt, return_tensors="pt", max_length=1024, truncation=True)
  txt_sum_summary_ids = txt_sum_bart_mdl.generate(txt_sum_bart_inputs, max_length=txt_sum_bart_mx_lth, num_beams=4, early_stopping=True)
  txt_sum_output = txt_sum_toknzr.decode(txt_sum_summary_ids[0], skip_special_tokens=True)
  return txt_sum_output
```

Figure 11 Coding for implementing BART model for text summarization

After reviewing the text's overview, the text data is summarized from 3114 count to 284 count. The BART is then analyzed by utilizing Rouge-score evaluation metric i.e., Rouge 1, Rouge 2 and Rouge L. Again, the same procedure is carried out for all sentence i.e., 100 sentence has been summarized using BART pre-trained model and then evaluated using Rouge-score evaluation metric i.e., Rouge 1, Rouge 2 and Rouge L.

Table 2 Results after implementing BART model - rouge score

Rouge Score - BART Model		Precision	Recall	F1 Score
Testing With All Sentences	Rouge - 1	1.00	0.88	0.16
	Rouge - 2	0.81	0.07	0.13
	Rouge - L	0.96	0.08	0.15

The above table shows the Rouge scores for the BART model in Text summarization. The Rouge scores measure the quality of text summarization by comparing the generated summaries with Old Bailey court proceedings text data. In this, ROUGE-1 is a measure that compares the generated text to the actual text based on the number of redundant unigram tokens. ROUGE-2 is a measure that compares a generated text to the actual text based on the number of redundant bigram (consecutive-word) tokens. ROUGE-L is a metric that compares the generated text to the reference text using the longest shared subsequence as the basis. It helps evaluate the naturalness and consistency by noting the longest sequence of source-text- and generated-text-words and generated texts.

When the BART method is utilized for text summarization and evaluated using the ROUGE-1, ROUGE-2, and ROUGE-L metrics, the precision scores are 1.00, 0.81, and 0.96, accordingly. This indicates that the produced outcomes are same as the original

text, with a precision of 100%, 81%, and 96%. However, the recall scores are 0.88, 0.07, and 0.08, respectively. This indicates that only 88% of the words in the Old Bailey court proceedings text data are present in the generated summaries and for recall of 7% and 8% i.e., Rouge 2 and Rouge L, the words in the Old Bailey court proceedings are not present in the generated summaries as per order. The F1 scores of the model are 0.16, 0.13, and 0.15, accordingly, which indicates that the BART model is not very good at generating summaries that are both similar to the original text and that capture most of the information in the original text at the phrase level.

4.5.2 BERT Model in Text Summarization for Proceedings

In the session, the pre-trained BERT model has been implemented first for summarizing a single sentence from the data, and then for summarizing all sentences from the data.

In the column 'Proceedings Content', the first entry [0] contains 3114 text data, and the pre-trained model is implemented by defining the model's name, setting the maximum length of the summarised text to 70, using a tokenizer to tokenize the input text and encode the data, Conditional Generation to produce the summarization of the text, and decoding the data into text format. Code to automate the BERT text-summarization methodology is provided below.

```
# defining the bert model to summarize the text data

def bert_txt_summarizer(txt_sum_inpt_txt):
  txt_sum_bert_mdl_nm ="bert-base-uncased"
  txt_sum_bert_mx_lth=150
  txt_sum_toknzr = txt_sum_bert_tknzr.from_pretrained(txt_sum_bert_mdl_nm)
  txt_sum_bert_mdl = txt_sum_bert_mskd_lm.from_pretrained(txt_sum_bert_mdl_nm)
  txt_sum_bert_inputs = txt_sum_toknzr(txt_sum_inpt_txt, return_tensors='pt', truncation=True, max_length=txt_sum_bert_mx_lth, padding=True)
  with txt_sum_trch.no_grad():
      txt_sum_outputs = txt_sum_bert_mdl.generate(txt_sum_bert_inputs.input_ids, attention_mask=txt_sum_bert_inputs.attention_mask, max_length=txt_sum_bert_mx_lth)

  txt_sum_output = txt_sum_toknzr.decode(txt_sum_outputs[0], skip_special_tokens=True)
  return txt_sum_output
```

Figure 12 Coding for implementing BERT model for text summarization

After reviewing the text's overview, the total of the text data is reduced from 3114 to 570. The BERT model is then evaluated utilizing the Rouge-score evaluation metric, that has Rouge 1, Rouge 2, and Rouge L. Again, the same procedure is followed for every sentence, i.e., 100 sentences have been summarised using a BERT methods that

are pre-trained and then evaluated using the Rouge-score evaluation metric, i.e., Rouge 1, Rouge 2, and Rouge L.

Table 3 Results after implementing BERT model - rouge score

Rouge Score - BERT Model		Precision	Recall	F1 Score
Testing With All Sentences	Rouge - 1	1.00	0.18	0.31
	Rouge - 2	1.00	0.18	0.31
	Rouge - L	1.00	0.18	0.31

The above table gives the scores for the BERT model in Text summarization. The Rouge scores the quality of text summarization in comparison with the generated summaries with Old Bailey court proceedings text data.

When the BERT method is utilized to summarize text and assessed using the ROUGE-1, ROUGE-2, and ROUGE-L metrics, the accuracy, recall, and F1 scores are 1.00, 0.18, and 0.31, respectively. The BERT model performs well at summarizing text that is comparable to the text in the Old Bailey court proceedings in terms of the words used, as seen by its high precision scores for both one sentence and all sentences. The reason behind this is that BERT just extracts the text from the source text, while other models like BART, T5, and GPT2 generate summarized text from the text in the Old Bailey court proceedings. However, the recall score of the BERT model is poor for both single sentences and all sentences. This demonstrates that the model struggles to produce content that, in terms of overall meaning, is comparable to the text in the Old Bailey court proceedings.

4.5.3 T5 Model in Text Summarization for Proceedings

The dataset now consists of 100 rows and 2 columns titled 'Category' and 'Proceedings Content'. In the session, the pre-trained T5 model has been implemented first for

summarizing a single sentence from the data, and then for summarizing all sentences from the data.

In the column 'Proceedings Content', the first entry [0] contains 3114 text data, and the pre-trained model is implemented by defining the model's name, setting the maximum length of the summarised text to 70, using a tokenizer to tokenize the input text and encode the data, Conditional Generation to produce the summarization of the text, and decoding the data into text format. Below is the code for implementing the T5 method.

```
# defining the model to summarize the text data

def t5_txt_summarizer(txt_sum_inpt_txt):
  txt_sum_t5_mdl_nm = "t5-small"
  txt_sum_t5_mx_lth=200
  txt_sum_toknzr = txt_sum_t5_tknzr.from_pretrained(txt_sum_t5_mdl_nm)
  txt_sum_t5_mdl = txt_sum_t5_cg.from_pretrained(txt_sum_t5_mdl_nm)
  txt_sum_t5_inputs = txt_sum_toknzr.encode("summarize: " + txt_sum_inpt_txt, return_tensors="pt", max_length=1024, truncation=True)
  txt_sum_summary_ids = txt_sum_t5_mdl.generate(txt_sum_t5_inputs, max_length=txt_sum_t5_mx_lth, num_beams=4, early_stopping=True)
  txt_sum_output = txt_sum_toknzr.decode(txt_sum_summary_ids[0], skip_special_tokens=True)
  return txt_sum_output
```

Figure 13 Coding for implementing T5 model for text summarization

The text data is summarised from 3114 count to 218 count after the content has been summarised. Then, the T5 model is assessed using the Rouge-score evaluation method, namely the Rouge 1, Rouge 2, and Rouge L. The process is repeated for every sentence; in this case, 100 sentences were analyzed using the T5 pre-trained model and scored using the Rouge-score assessment metric (Rouge 1, Rouge 2, and Rouge L).

Table 4 Results after implementing T5 model - rouge score

Rouge Score - T5 Model		Precision	Recall	F1 Score
Testing With All Sentences	Rouge - 1	0.90	0.05	0.10
	Rouge - 2	0.34	0.02	0.04
	Rouge - L	0.67	0.04	0.08

The above table shows the Rouge scores for the T5 model in Text summarization. The Rouge scores gives the quality of text summarization by comparing the generated summaries with Old Bailey court proceedings text data.

The precision, recall, and scores for F1 were 0.90, 0.34, and 0.67, respectively, while the pre-trained T5 algorithm is employed for analysing text and assessed employing the ROUGE-1, ROUGE-2, along with ROUGE-L metrics. The T5 model has a high precision score for both one sentence and all sentences and it is good at generating text that is similar to the text in the Old Bailey court proceedings in terms of the words that

are used. However, the T5 model has a very low recall and f1 score for both one sentence and all sentences and it is not good at generating text that is similar to the reference in terms of the overall meaning of the text.

4.5.4 GPT2 Model in Text Summarization for Proceedings

Starting with summarizing a single sentence from the data, the pre-trained GPT2 system was deployed in this session. In the column 'Proceedings Content', the first entry [0] contains 3114 text data and the pre-trained model is implemented by specifying the model's name and maximum summary length as 70 tokens, tokenizer are utilized for tokenize the input text & encode the data and further Conditional Generation is used for generating the summarization of the text and finally the data is decoded into text format. Below is the code used to put into action the GPT2 model for text summarising.

```
# defining the GPT2 model to summarize the text data

def gpt2_txt_summarizer(txt_sum_inpt_txt):
  txt_sum_gpt2_mdl_nm ="gpt2"
  txt_sum_gpt2_mx_lth=150
  txt_sum_toknzr = txt_sum_gpt2_tknzr.from_pretrained(txt_sum_gpt2_mdl_nm)
  txt_sum_gpt2_mdl = txt_sum_gpt2_lm_hd.from_pretrained(txt_sum_gpt2_mdl_nm)
  txt_sum_gpt2_inputs = txt_sum_toknzr(txt_sum_inpt_txt, return_tensors='pt', truncation=True, max_length=txt_sum_gpt2_mx_lth)
  with txt_sum_trch.no_grad():
      txt_sum_outputs = txt_sum_gpt2_mdl.generate(txt_sum_gpt2_inputs.input_ids, attention_mask=txt_sum_gpt2_inputs.attention_mask, max_length=txt_sum_gpt2_mx_lth)

  txt_sum_output = txt_sum_toknzr.decode(txt_sum_outputs[0], skip_special_tokens=True)
  return txt_sum_output
```

Figure 14 Coding for implementing GPT2 model for text summarization

After reviewing the text's overview, the text data is summarized from 3114 count to 544 count. The GPT2 method is then analyzed by utilizing Rouge-score evaluation metric i.e., Rouge 1, Rouge 2 and Rouge L. Again, the same procedure is carried out for all sentence i.e., 100 sentence has been summarized using GPT2 pre-trained model and then evaluated using Rouge-score evaluation metric i.e., Rouge 1, Rouge 2 and Rouge L.

Table 5 Results after implementing GPT2 model - rouge score

Rouge Score - GPT2 Model		Precision	Recall	F1 Score
Testing With All Sentences	Rouge - 1	0.99	0.18	0.31
	Rouge - 2	0.99	0.18	0.31
	Rouge - L	0.99	0.18	0.31

The above table gives Rouge scores for the GPT2 model in Text summarization. The Rouge scores measure the quality of text summarization by comparison the generated summaries with Old Bailey court proceedings text data.

Utilizing a ROUGE-1, ROUGE-2, along with ROUGE-L metrics, the pre-trained GPT2 model achieves precision, recall, and F1 scores of 0.99, 0.18, and 0.31 while utilized for summarizing text. The GPT2 model has a high precision score for both one sentence and all sentences, which means that the model is good at generating text that is similar to the text in the Old Bailey court proceedings in terms of the words that are used. However, the GPT2 model has a low recall score for both one sentence and all sentences. This means that the model is not good at generating text that is similar to the text in the Old Bailey court proceedings in terms of the overall meaning of the text.

4.6Novelty

This study is novel in its approach since it compares popular trained methods such as BERT, BART, GPT2, and T5 in the context of summarising Old Bailey events. It fills a knowledge vacuum by examining how these models function with content that varies in length from short to long in the legal domain.

5 RESULTS & CONCLUSION

5.1 Comparison of all pre-trained models

They utilized and analyzed the pre-trained BART, BERT, T5, and GPT2 models with the help of the Rouge score metrics (Rouge 1, Rouge 2, and Rouge L). The following list provides a visual overview of the assessed findings.

Table 6 Table showing comparison of all pre-trained models

Rouge Score		Precision	Recall	F1 Score
BART	Rouge - 1	1.00	0.88	0.16
	Rouge - 2	0.81	0.07	0.13
	Rouge - L	0.96	0.08	0.15
BERT	Rouge - 1	1.00	0.18	0.31
	Rouge - 2	1.00	0.18	0.31
	Rouge - L	1.00	0.18	0.31
T5	Rouge - 1	0.90	0.05	0.10
	Rouge - 2	0.34	0.02	0.04
	Rouge - L	0.67	0.04	0.08
GPT2	Rouge - 1	0.99	0.18	0.31
	Rouge - 2	0.99	0.18	0.31
	Rouge - L	0.99	0.18	0.31

BERT has the highest ROUGE scores for all three ROUGE metrics, which suggests that BERT is the best at generating summaries that are similar to the original text. However, BERT also has the lowest recall scores, which suggests that BERT does not capture as much information from the original text as the other models.

Followed by BERT, GPT2 has the second highest ROUGE scores for all three ROUGE metrics. This suggests that GPT2 is somewhat good at generating summaries that are similar to the original text. BART has ROUGE scores that are similar to GPT2. However, GPT2's and BERT has the same recall and F1 scores of 0.18 and 0.31, which suggests that GPT-2 does not capture as much information from the original text as BERT.

T5: T5 has the lowest ROUGE scores for all three ROUGE metrics. This suggests that T5 is the worst at generating summaries that are similar to the original text. However, T5's recall scores are higher than BART's, which suggests that T5 captures more information from the original text than BART.

Overall, BERT is the best model at producing synopses of are similar to the original text. BART and GPT-2 are also good at producing synopses of are similar to the original text. T5 is the worst model at producing synopses of are similar to the original text.

5.2Critical Analysis

The research critically examines the strengths and weaknesses of pre-trained algorithms used to summarise court cases. Models that are pretrained are shown to be effective in abstractive summarization, and this study looks into whether or not they can be applied to legal content. This study employs a dataset found on the Old Bailey website to do a ROUGE-metrics-based evaluation of method like as BART, BERT, T5, and GPT-2. According to the findings, BERT and GPT-2 are effective at producing information that is consistent with references, as seen by their high precision and recall scores. But problems with preserving correct word order exist. This research provides useful information about using pre-trained models for legal summarization and makes suggestions for future research and development in this field.

5.3Research Findings

Why does BART have a high Rouge-1 score but a low Rouge-2 and Rouge-L score?

The Rouge-1 score calculates the overlap between the generated text and the text in the Old Bailey court proceedings, while the Rouge-2 and Rouge-L scores measure the overlap among the produced text and the text in the Old Bailey court proceedings, taking into consideration the order of the words. BART has a high Rouge-1 score

because it is good at generating text that is similar to the text in the Old Bailey court proceedings in terms of the words that are used. However, BART has a low Rouge-2 and Rouge-L score because it is not as good at generating text that is similar to the text in the Old Bailey court proceedings in terms of the order of the words.

Why does GPT-2 have a similar Rouge-1, Rouge-2, and Rouge-L score to BERT?

GPT-2 and BERT have similar Rouge-1, Rouge-2, and Rouge-L scores because they are both trained on similar datasets of text. This allows both models to learn how to produce copy that closely resembles the copy in the Old Bailey court proceedings in terms of the words that are used, the order of the words, and the overall meaning.

As a result, GPT-2 pre-trained model is more likely to produce a copy that closely resembles the text in the Old Bailey court proceedings in terms of the words that are used, but not necessarily the order of the words or the overall meaning. BERT pre-trained model is more likely to produce copy that closely resembles the copy in the Old Bailey court proceedings in terms of the order of the words and the overall meaning, but not necessarily the specific words that are used.

Why does T5 have a low Rouge-1, Rouge-2, and Rouge-L score?

T5 has a low Rouge-1, Rouge-2, and Rouge-L score because it is trained on a dataset of text that is very different from the type of text that is typically used to evaluate language models. The proceedings articles dataset is often very long and complex, while the summaries in the evaluation dataset are typically much shorter and simpler. This can make it difficult for T5 to produce summaries that are conceptually close to the Old Bailey court proceedings text data in terms of the words that are used, the context and the meaning of the words as they are arranged.

Why all the pre-trained models give good precision but low recall and F1 score?

The models such as BART, BERT, T5 and GPT2 give good precision but low recall and F1 score because they are trained on a massive dataset of text that is very diverse. This diversity can be a weakness, as it can make it difficult for the language models to

generate text that is specific and accurate. This can lead to a situation where the language model has a high precision, and generate text that is same as the text in the Old Bailey court proceedings. However, it may also have a low recall and F1 score, as it may not be able to generate text that is specific and accurate.

5.3.1 Comparing with other research work

The comparison of various research work with this research work are detailed below.

Table 7 Comparing of other research work

Author	**Objective**	**Parameter used**	**Result Achieved**
Alami et al.	Integrating word embeddings with ensemble learning into unsupervised neural networks for enhanced text summarising	Ensemble learning and word embedding	Improved precision and recall by 8.72% and 27.08%, respectively.
Zhang et al.	Condensing texts with MLP and pretrained models	Pretraining-based natural language generation	Improved ROUGE scores of up to 4.57%.
Goularte et al.	Fuzzy rule-based text summarization for automated assessment	Fuzzy Rules	High accuracy of 98.50%.
Song et al.	Summarization of abstract texts using a deep learning LSTM-CNN model	LSTM-CNN based DL	ROUGE-1, ROUGE-2, and ROUGE-L scores of 0.44, 0.21, and 0.41, respectively.
Ramesh et al.	Summarising Abstract Text using the T5 Framework	T5 Architecture	ROUGE-1 and ROUGE-2 scores of 0.72 and 0.51, respectively.

This research work focus on evaluating the Pre-Trained Models in text summarization for proceedings. The data is preprocessed and implemented using pre-trained methods such as BART, BERT, T5 and GPT-2 and evaluated using Rouge – 1, Rouge – 2 and Rouge – L evaluation metrics. Through evaluation and comparison, the pre-trained

model BERT and GPT-2 gives almost similar maximum precision of 100% and 99% and recall of 18% and f1 score of 31%, which infers that the BERT and GPT-2 models are good at generating summarized text that is similar to the text in the Old Bailey court proceedings in terms of the words that are used. However, both the models are not good at generating summarized text that is similar to the text in the Old Bailey court proceedings in terms of the order of the words.

5.4Conclusion

Text summarization is a challenging task that involves extraction of most important data from a text file and generates a concise summary that retains the original meaning. The language models that are pretrained have recently given promising outcomes for text summarization, especially for abstractive summarization tasks. However, it is not clear how well these models perform for summarizing legal proceedings using pre-trained models. Thus, this research work focus on evaluating the Pre-Trained Models in text summarization for proceedings. For this purpose, dataset has been obtained from Old Bailey website, that dates from 1674 to 1913 and is housed at London, England's Old Bailey Courthouse, contains archive records of criminal trial processes. The data is preprocessed and implemented using pre-trained models such as BART, BERT, T5 and GPT-2 and evaluated using Rouge – 1, Rouge – 2 and Rouge – L evaluation metrics for one sentence and also for all sentence. Through evaluation and comparison, the pre-trained model BERT and GPT-2 gives almost similar maximum precision of 100% and 99% and recall of 18% and f1 score of 31%, which infers that the BERT and GPT-2 models are good at generating summarized text that is similar to the text in the Old Bailey court proceedings in terms of the words that are used for one sentence and also for all sentences. However, both the models are not good at generating summarized text that is similar to the text in the Old Bailey court proceedings in terms of the order of the words.

Addressing Research Question,

RQ 1: Does the BERT pre-trained models more effective than other text summarization models such as BART, T5 and GPT for Old Bailey court proceedings?

Yes, in both lengthy and short sessions, the pre-trained models perform better. The pre-trained model BERT gives maximum precision of 100% and 99% and recall of 18% and f1 score of 31%. Meanwhile, GPT-2 model also good at generating summarized long and short proceedings that is similar to the text in the Old Bailey court proceedings in terms of the words.

5.5Summary of Achievements

The following are the summary of the achievements,

- This research has found that BART, BERT, and GPT-2 all performed well at summarizing these texts, but that T5 performed poorly.
- This research work has evaluated how well various pre-trained models perform in summarizing texts of Old Bailey proceedings and other legal documents.
- This research has concluded that pre-trained models are well-suited for summarizing legal documents, but that they may not be able to summarize very large or complex documents accurately.

5.6Reflection

- The study showed that Old Bailey court events could be adequately summarised by BERT, BART, and GPT-2, with BERT attaining the highest levels of precision (100% and 99%) and recall (18%, leading to an F1 score of 31%).
- As evidence of the efficacy of numerous pre-trained models in text summarising, GPT-2 showed proficiency in producing summaries that resembled the court events.
- Notably, T5's poor performance highlights the diversity in effectiveness between pre-trained models for this job.

- ❖ According to the findings, the models that are pretrained such as BERT, BART, and GPT-2 perform admirably when it comes to summarising legal papers; yet, difficulties persist when it comes to accurately summarising excessively vast or complex texts.

5.7Future Work

This research work has evaluated the efficacy of the models using the rouge metric. Rouge is an excellent metric to evaluate the overlap among the generated text along with the text in the Old Bailey court proceedings, but it does not measure the fluency or precision of the generated text. Using a more advanced evaluation metric, including METEOR or CIDEr, would aid in obtaining a more accurate depiction of the efficacy of the models, which can be taken into account for future improvement.

It is recommended that, for future work, a larger dataset be used, as this would enhance the accuracy of the models and enable the team to make more generalizable claims about the performance of pre-trained methodss for summarizing proceedings. It is also suggested that new techniques for summarizing proceedings, such as reinforcement learning or neural machine translation, could enhance the process' accuracy and efficacy.

6 REFERENCES

Akhmetov, I., Mussabayev, R. and Gelbukh, A., 2022. Reaching for upper bound ROUGE score of extractive summarization methods. PeerJ Computer Science, 8, p.e1103.

Alami, N., Meknassi, M. and En-nahnahi, N., 2019. Enhancing unsupervised neural networks based text summarization with word embedding and ensemble learning. Expert systems with applications, 123, pp.195-211.

Al-Ghamdi, S., Al-Khalifa, H. and Al-Salman, A., 2023. Fine-Tuning BERT-Based Pre-Trained Models for Arabic Dependency Parsing. Applied Sciences, 13(7), p.4225.

Allahyari, M., Pouriyeh, S., Assefi, M., Safaei, S., Trippe, E.D., Gutierrez, J.B. and Kochut, K., 2017. Text summarization techniques: a brief survey. arXiv preprint arXiv:1707.02268.

Alokla, A., Gad, W., Nazih, W., Aref, M. and Salem, A.B., 2022. Pseudocode Generation from Source Code Using the BART Model. Mathematics, 10(21), p.3967.

Clark, K., Khandelwal, U., Levy, O. and Manning, C.D., 2019. What does bert look at? an analysis of bert's attention. *arXiv preprint arXiv:1906.04341*.

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

Dor, L.E., Halfon, A., Gera, A., Shnarch, E., Dankin, L., Choshen, L., Danilevsky, M., Aharonov, R., Katz, Y. and Slonim, N., 2020, November. Active learning for BERT: an empirical study. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP) (pp. 7949-7962).

Duan, J., Zhao, H., Zhou, Q., Qiu, M. and Liu, M., 2020, November. A study of pre-trained language models in natural language processing. In 2020 IEEE International Conference on Smart Cloud (SmartCloud) (pp. 116-121). IEEE.

El-Kassas, W.S., Salama, C.R., Rafea, A.A. and Mohamed, H.K., 2021. Automatic text summarization: A comprehensive survey. Expert systems with applications, 165, p.113679.

Fabbri, A.R., Li, I., She, T., Li, S. and Radev, D.R., 2019. Multi-news: A large-scale multi-document summarization dataset and abstractive hierarchical model. arXiv preprint arXiv:1906.01749.

Ganesan, K., 2018. Rouge 2.0: Updated and improved measures for evaluation of summarization tasks. arXiv preprint arXiv:1803.01937.

Ghadimi, A. and Beigy, H., 2022. Hybrid multi-document summarization using pre-trained language models. Expert Systems with Applications, 192, p.116292.

Goularte, F.B., Nassar, S.M., Fileto, R. and Saggion, H., 2019. A text summarization method based on fuzzy rules and applicable to automated assessment. Expert Systems with Applications, 115, pp.264-275.

Grusky, M., 2023, July. Rogue Scores. In Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) (pp. 1914-1934).

Gundapu, S. and Mamidi, R., 2021. Transformer based automatic COVID-19 fake news detection system. arXiv preprint arXiv:2101.00180.

Han, X., Zhang, Z., Ding, N., Gu, Y., Liu, X., Huo, Y., Qiu, J., Yao, Y., Zhang, A., Zhang, L. and Han, W., 2021. Pre-trained models: Past, present and future. AI Open, 2, pp.225-250.

Heilbron, M., Ehinger, B., Hagoort, P. and De Lange, F.P., 2019. Tracking naturalistic linguistic predictions with deep neural language models. arXiv preprint arXiv:1909.04400.

Hwang, M.H., Shin, J., Seo, H., Im, J.S., Cho, H. and Lee, C.K., 2023. Ensemble-NQG-T5: Ensemble Neural Question Generation Model Based on Text-to-Text Transfer Transformer. Applied Sciences, 13(2), p.903.

Itsnaini, Q.A.Y., Hayaty, M., Putra, A.D. and Jabari, N.A., 2023. Abstractive text summarization using Pre-Trained Language Model" Text-to-Text Transfer Transformer (T5)". ILKOM Jurnal Ilmiah, 15(1), pp.124-131.

Joshi, A., Fidalgo, E., Alegre, E. and Fernández-Robles, L., 2023. DeepSumm: Exploiting topic models and sequence to sequence networks for extractive text summarization. Expert Systems with Applications, 211, p.118442.

Joshi, A., Fidalgo, E., Alegre, E. and Fernández-Robles, L., 2023. DeepSumm: Exploiting topic models and sequence to sequence networks for extractive text summarization. Expert Systems with Applications, 211, p.118442.

Kanapala, A., Pal, S. and Pamula, R., 2019. Text summarization from legal documents: a survey. Artificial Intelligence Review, 51, pp.371-402.

Keneshloo, Y., Ramakrishnan, N. and Reddy, C.K., 2019, May. Deep transfer reinforcement learning for text summarization. In Proceedings of the 2019 SIAM International Conference on Data Mining (pp. 675-683). Society for Industrial and Applied Mathematics. Liu, Y. and Lapata, M., 2019. Text summarization with pretrained encoders. arXiv preprint arXiv:1908.08345.

Khandelwal, U., Clark, K., Jurafsky, D. and Kaiser, L., 2019. Sample efficient text summarization using a single pre-trained transformer. arXiv preprint arXiv:1905.08836.

Kieuvongngam, V., Tan, B. and Niu, Y., 2020. Automatic text summarization of covid-19 medical research articles using bert and gpt-2. arXiv preprint arXiv:2006.01997.

Kim, Y., Kim, J.H., Lee, J.M., Jang, M.J., Yum, Y.J., Kim, S., Shin, U., Kim, Y.M., Joo, H.J. and Song, S., 2022. A pre-trained BERT for Korean medical natural language processing. Scientific Reports, 12(1), p.13847.

Kouris, P., Alexandridis, G. and Stafylopatis, A., 2022. Abstractive text summarization based on deep learning and semantic content generalization.

Kumar, V., Choudhary, A. and Cho, E., 2020. Data augmentation using pre-trained transformer models. arXiv preprint arXiv:2003.02245.

Lee, J.S. and Hsiang, J., 2020. Patent claim generation by fine-tuning OpenAI GPT-2. World Patent Information, 62, p.101983.

Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V. and Zettlemoyer, L., 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. arXiv preprint arXiv:1910.13461.

Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V. and Zettlemoyer, L., 2019. Bart: Denoising sequence-to-sequence pre-training for

natural language generation, translation, and comprehension. arXiv preprint arXiv:1910.13461.

Li, J., Sun, A., Han, J. and Li, C., 2020. A survey on deep learning for named entity recognition. IEEE Transactions on Knowledge and Data Engineering, 34(1), pp.50-70.

Li, L., Du, B., Wang, Y., Qin, L. and Tan, H., 2020. Estimation of missing values in heterogeneous traffic data: Application of multimodal deep learning model. *Knowledge-Based Systems*, *194*, p.105592.

Liu, F., Shakeri, S., Yu, H. and Li, J., 2021. Enct5: Fine-tuning t5 encoder for non-autoregressive tasks. arXiv e-prints, pp.arXiv-2110.

Liu, J., Wu, J. and Luo, X., 2021. Chinese judicial summarising based on short sentence extraction and GPT-2. In Knowledge Science, Engineering and Management: 14th International Conference, KSEM 2021, Tokyo, Japan, August 14–16, 2021, Proceedings, Part II 14 (pp. 376-393). Springer International Publishing.

Liu, X., Zheng, Y., Du, Z., Ding, M., Qian, Y., Yang, Z. and Tang, J., 2021. GPT understands, too. *arXiv preprint arXiv:2103.10385*.

Liu, Y., 2019. Fine-tune BERT for extractive summarization. arXiv preprint arXiv:1903.10318.

Luo, Z., Xie, Q. and Ananiadou, S., 2023. Chatgpt as a factual inconsistency evaluator for abstractive text summarization. arXiv preprint arXiv:2303.15621.

Madatov, K., Bekchanov, S. and Vičič, J., 2023. Uzbek text summarization based on TF-IDF. arXiv preprint arXiv:2303.00461.

Madatov, K., Bekchanov, S. and Vičič, J., 2023. Uzbek text summarization based on TF-IDF. arXiv preprint arXiv:2303.00461.

Maharana, K., Mondal, S. and Nemade, B., 2022. A review: Data pre-processing and data augmentation techniques. *Global Transitions Proceedings*, *3*(1), pp.91-99.

Mastropaolo, A., Scalabrino, S., Cooper, N., Palacio, D.N., Poshyvanyk, D., Oliveto, R. and Bavota, G., 2021, May. Studying the usage of text-to-text transfer transformer to support code-related tasks. In 2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE) (pp. 336-347). IEEE.

McGuffie, K. and Newhouse, A., 2020. The radicalization risks of GPT-3 and advanced neural language models. arXiv preprint arXiv:2009.06807.

Min, B., Ross, H., Sulem, E., Veyseh, A.P.B., Nguyen, T.H., Sainz, O., Agirre, E., Heintz, I. and Roth, D., 2021. Recent advances in natural language processing via large pre-trained language models: A survey. ACM Computing Surveys.

Müller, M., Salathé, M. and Kummervold, P.E., 2023. Covid-twitter-bert: A natural language processing model to analyse covid-19 content on twitter. Frontiers in Artificial Intelligence, 6, p.1023281.

Mustar, A., Lamprier, S. and Piwowarski, B., 2020, July. Using BERT and BART for query suggestion. In Joint Conference of the Information Retrieval Communities in Europe (Vol. 2621). CEUR-WS. org.

Nagoudi, E.M.B., Elmadany, A. and Abdul-Mageed, M., 2021. AraT5: Text-to-text transformers for Arabic language generation. arXiv preprint arXiv:2109.12068.

Ou, Z., Zhang, M. and Zhang, Y., 2022. On the Role of Pre-trained Language Models in Word Ordering: A Case Study with BART. arXiv preprint arXiv:2204.07367.

Qu, Y., Liu, P., Song, W., Liu, L. and Cheng, M., 2020, July. A text generation and prediction system: pre-training on new corpora using BERT and GPT-2. In 2020 IEEE 10th international conference on electronics information and emergency communication (ICEIEC) (pp. 323-326). IEEE.

Ramesh, G.S., Manyam, V., Mandula, V., Myana, P., Macha, S. and Reddy, S., 2022. Abstractive Text Summarization Using T5 Architecture. In Proceedings of Second International Conference on Advances in Computer Engineering and Communication Systems (pp. 535-543). Springer, Singapore.

Rinse, V. and Siitova, A., 2019. Text summarization using transfer learnin: Extractive and abstractive summarization using bert and gpt-2 on news and podcast data.

Singh, A., 2020. Point-5: Pointer network and t-5 based financial narrativesummarisation. *arXiv preprint arXiv:2010.04191*.

Song, S., Huang, H. and Ruan, T., 2019. Abstractive text summarization using LSTM-CNN based deep learning. Multimedia Tools and Applications, 78, pp.857-875.

Soper, E., Fujimoto, S. and Yu, Y.Y., 2021, November. BART for post-correction of OCR newspaper text. In *Proceedings of the Seventh Workshop on Noisy User-generated Text (W-NUT 2021)* (pp. 284-290).

Wang, D., Liu, P., Zheng, Y., Qiu, X. and Huang, X., 2020. Heterogeneous graph neural networks for extractive document summarization. arXiv preprint arXiv:2004.12393.

Yang, H., 2018. Data preprocessing. *Pennsylvania State University: Citeseer*.

Yang, P.J., Chen, Y.T., Chen, Y. and Cer, D., 2021. Nt5?! training t5 to perform numerical reasoning. arXiv preprint arXiv:2104.07307.

Yang, X., Li, Y., Zhang, X., Chen, H. and Cheng, W., 2023. Exploring the limits of chatgpt for query or aspect-based text summarization. arXiv preprint arXiv:2302.08081.

Yang, X., Li, Y., Zhang, X., Chen, H. and Cheng, W., 2023. Exploring the limits of chatgpt for query or aspect-based text summarization. arXiv preprint arXiv:2302.08081.

Yang, Z., Dong, Y., Deng, J., Sha, B. and Xu, T., 2021, October. Research on Automatic News Text Summarization Technology Based on GPT2 Model. In 2021 3rd International Conference on Artificial Intelligence and Advanced Manufacture (pp. 418-423). Zopf, M., Mencía, E.L. and Fürnkranz, J., 2018, June. Which scores to predict in sentence regression for text summarization? In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers) (pp. 1782-1791).

Zhang, H., Xu, J. and Wang, J., 2019. Pretraining-based natural language generation for text summarization. arXiv preprint arXiv:1902.09243.

Zhang, X., Wei, F. and Zhou, M., 2019. HIBERT: Document level pre-training of hierarchical bidirectional transformers for document summarization. arXiv preprint arXiv:1905.06566.

Zhao, S., You, F. and Liu, Z.Y., 2020. Leveraging Pre-Trained language model for summary generation on short text. IEEE Access, 8, pp.228798-228803.

APPENDIX

1- Old Bailey Proceedings Files collection

```
import pandas as baile_ProcedPs
##import Old Bailey Proceedings Files
baile_Proced = baile_ProcedPs.read_excel('Old Bailey Proceedings.xlsx')
baile_Proced.shape

baile_Proced[:5]# 5 top data

baile_Proced[-5:]# 5 tail data

*** the data contains date of the bails, link, category and proceedings contents.
*** from year 1900 to 1910, totally 100 cases are available.

baile_Proced['Category'].nunique()## number of case category

baile_Proced['Category'].unique()

** cases are under 9 different categories.

baile_Proced['Category'].value_counts()

baile_Proced['link'][:10] #some link

baile_Proced['Proceedings Content'][3]# 3rd case proceedings.

baile_Proced.info()

** infor: All the attributes are object type data.
** And non_ull.

## EDA

import seaborn as baile_ProcedBSr
## Plot to check the category
baile_ProcedBSr.countplot(x ='Category', data = baile_Proced).set(title='Old Bailey
Proceedings Category')

import matplotlib.pyplot as baile_ProcedYP
baile_ProcedYP.xticks(rotation=91)

** 'Theft' kind of cases are more.
```

```
##Drop date and link attributes...
del baile_Proced['Date']
del baile_Proced['link']

baile_Proced.to_csv('Data.csv', index=False)

!pip install rouge_score

from rouge_score import rouge_scorer

# a list of the hypothesis documents
hyp = ['hi.. this is manisha here']
# a list of the references documents
ref = ['hi.. manisha here']

# make a RougeScorer object with rouge_types=['rouge1']
scorer = rouge_scorer.RougeScorer(['rouge1'])

# a dictionary that will contain the results
results = {'precision': [], 'recall': [], 'fmeasure': []}

# for each of the hypothesis and reference documents pair
for (h, r) in zip(hyp, ref):
    # computing the ROUGE
    score = scorer.score(h, r)
    # separating the measurements
    precision, recall, fmeasure = score['rouge1']
    # add them to the proper list in the dictionary
    results['precision'].append(precision)
    results['recall'].append(recall)
    results['fmeasure'].append(fmeasure)

results
```

2- Bart

```
!pip install transformers
!pip install rouge-score

import pandas as txt_sum_pd
import torch as txt_sum_trch
from transformers import BartTokenizer as txt_sum_bart_tknzr
from transformers import BartForConditionalGeneration as txt_sum_bart_cg
from rouge_score import rouge_scorer as txt_sum_rg_scr
from transformers import pipeline as txt_sum_rg_ppln
```

```
txt_sum_data_frm = txt_sum_pd.read_csv("Data.csv")
txt_sum_data_frm.shape

txt_sum_data_frm

txt_sum_data_frm['Proceedings Content'].str.len()

#### Summarizing one sentence from the data using BART as an example to
understand the before and after results of summarization

txt_sum_ex_text = txt_sum_data_frm["Proceedings Content"][0]
print(len(txt_sum_ex_text))

def bart_txt_summarizer(txt_sum_inpt_txt):
  txt_sum_bart_mdl_nm ="facebook/bart-large-cnn"
  txt_sum_bart_mx_lth=70
  txt_sum_toknzr = txt_sum_bart_tknzr.from_pretrained(txt_sum_bart_mdl_nm)
  txt_sum_bart_mdl = txt_sum_bart_cg.from_pretrained(txt_sum_bart_mdl_nm)
  txt_sum_bart_inputs = txt_sum_toknzr.encode("summarize: " + txt_sum_inpt_txt,
return_tensors="pt", max_length=1024, truncation=True)
  txt_sum_summary_ids = txt_sum_bart_mdl.generate(txt_sum_bart_inputs,
max_length=txt_sum_bart_mx_lth, num_beams=4, early_stopping=True)
  txt_sum_output = txt_sum_toknzr.decode(txt_sum_summary_ids[0],
skip_special_tokens=True)
  return txt_sum_output


txt_sum_ex_output = bart_txt_summarizer(txt_sum_ex_text)

print(len(txt_sum_ex_output))

txt_sum_mdl_nm = "facebook/bart-large-cnn"
txt_sum_smrzr = txt_sum_rg_ppln("summarization", model=txt_sum_mdl_nm)

txt_sum_opt_smry = txt_sum_ex_output
txt_sum_inpt_ex = txt_sum_ex_text

txt_sum_scr = txt_sum_rg_scr.RougeScorer(['rouge1', 'rouge2', 'rougeL'],
use_stemmer=True)
txt_sum_scrs = txt_sum_scr.score(txt_sum_inpt_ex, txt_sum_opt_smry)

for txt_sum_mtrc, txt_sum_scr in txt_sum_scrs.items():
    print(f'{txt_sum_mtrc}:')
    print(f'  Precision: {txt_sum_scr.precision:.4f}')
    print(f'  Recall: {txt_sum_scr.recall:.4f}')
    print(f'  F1 Score: {txt_sum_scr.fmeasure:.4f}')
```

Summarizing all the sentences in the data

defining the model to summarize the text data

```
def bart_txt_summarizer(txt_sum_inpt_txt):
  txt_sum_bart_mdl_nm ="facebook/bart-large-cnn"
  txt_sum_bart_mx_lth=70
  txt_sum_toknzr = txt_sum_bart_tknzr.from_pretrained(txt_sum_bart_mdl_nm)
  txt_sum_bart_mdl = txt_sum_bart_cg.from_pretrained(txt_sum_bart_mdl_nm)
  txt_sum_bart_inputs = txt_sum_toknzr.encode("summarize: " + txt_sum_inpt_txt,
return_tensors="pt", max_length=1024, truncation=True)
  txt_sum_summary_ids = txt_sum_bart_mdl.generate(txt_sum_bart_inputs,
max_length=txt_sum_bart_mx_lth, num_beams=4, early_stopping=True)
  txt_sum_output = txt_sum_toknzr.decode(txt_sum_summary_ids[0],
skip_special_tokens=True)
  return txt_sum_output

txt_sum_output_lis = []
for txt in txt_sum_data_frm['Proceedings Content']:
  txt_sum_smrzd_opt = bart_txt_summarizer(txt)
  txt_sum_output_lis.append(txt_sum_smrzd_opt)

txt_sum_opt_dta_frm = txt_sum_pd.DataFrame()
for opt in range(len(txt_sum_output_lis)):
  txt_sum_pre =
txt_sum_pd.Series({'Proceeding_content':txt_sum_data_frm['Proceedings
Content'].unique()[opt],'Bart_summarized_output':txt_sum_output_lis[opt]})
  txt_sum_opt_dta_frm =
txt_sum_opt_dta_frm.append(txt_sum_pre,ignore_index=True)
txt_sum_opt_dta_frm['Category']=txt_sum_data_frm['Category']
txt_sum_opt_dta_frm

txt_sum_opt_dta_frm['Bart_summarized_output'].str.len()
```

Evaluation using Rougue Score

```
txt_sum_mdl_nm = "facebook/bart-large-cnn"

txt_sum_smrzr = txt_sum_rg_ppln("summarization", model=txt_sum_mdl_nm)

row_index = 0

txt_sum_opt_smry =
txt_sum_opt_dta_frm['Bart_summarized_output'].iloc[row_index]
txt_sum_inpt_ex = txt_sum_opt_dta_frm["Proceeding_content"].iloc[row_index]
```

```
txt_sum_scr = txt_sum_rg_scr.RougeScorer(['rouge1', 'rouge2', 'rougeL'],
use_stemmer=True)
txt_sum_scrs = txt_sum_scr.score(txt_sum_inpt_ex, txt_sum_opt_smry)

for txt_sum_mtrc, txt_sum_scr in txt_sum_scrs.items():
    print(f"{txt_sum_mtrc}:")
    print(f"  Precision: {txt_sum_scr.precision:.4f}")
    print(f"  Recall: {txt_sum_scr.recall:.4f}")
    print(f"  F1 Score: {txt_sum_scr.fmeasure:.4f}")
```

3- Bert

```
from transformers import BertTokenizer as txt_sum_bert_tknzr
from transformers import BertForMaskedLM as txt_sum_bert_mskd_lm
from transformers import BertModel as txt_sum_bert_mdl


#### Summarizing one sentence from the data using BERT as an example to
understand the before and after results of summarization

txt_sum_ex_text = txt_sum_data_frm["Proceedings Content"][0]
print(len(txt_sum_ex_text))

def bert_txt_summarizer(txt_sum_inpt_txt):
  txt_sum_bert_mdl_nm ="bert-base-uncased"
  txt_sum_bert_mx_lth=150
  txt_sum_toknzr = txt_sum_bert_tknzr.from_pretrained(txt_sum_bert_mdl_nm)
  txt_sum_bert_mdl =
txt_sum_bert_mskd_lm.from_pretrained(txt_sum_bert_mdl_nm)
  txt_sum_bert_inputs = txt_sum_toknzr(txt_sum_inpt_txt, return_tensors='pt',
truncation=True, max_length=txt_sum_bert_mx_lth, padding=True)
  with txt_sum_trch.no_grad():
      txt_sum_outputs = txt_sum_bert_mdl.generate(txt_sum_bert_inputs.input_ids,
attention_mask=txt_sum_bert_inputs.attention_mask,
max_length=txt_sum_bert_mx_lth)

  txt_sum_output = txt_sum_toknzr.decode(txt_sum_outputs[0],
skip_special_tokens=True)
  return txt_sum_output


txt_sum_ex_output = bert_txt_summarizer(txt_sum_ex_text)

print(len(txt_sum_ex_output))
```

```
txt_sum_mdl_nm = "bert-base-uncased"

txt_sum_smrzr = txt_sum_rg_ppln("summarization", model=txt_sum_mdl_nm)

txt_sum_opt_smry = txt_sum_ex_output
txt_sum_inpt_ex = txt_sum_ex_text

txt_sum_scr = txt_sum_rg_scr.RougeScorer(['rouge1', 'rouge2', 'rougeL'],
use_stemmer=True)
txt_sum_scrs = txt_sum_scr.score(txt_sum_inpt_ex, txt_sum_opt_smry)

for txt_sum_mtrc, txt_sum_scr in txt_sum_scrs.items():
    print(f"{txt_sum_mtrc}:")
    print(f"  Precision: {txt_sum_scr.precision:.4f}")
    print(f"  Recall: {txt_sum_scr.recall:.4f}")
    print(f"  F1 Score: {txt_sum_scr.fmeasure:.4f}")


#### Summarizing all the sentences in the data


# defining the bert model to summarize the text data

def bert_txt_summarizer(txt_sum_inpt_txt):
  txt_sum_bert_mdl_nm ="bert-base-uncased"
  txt_sum_bert_mx_lth=150
  txt_sum_toknzr = txt_sum_bert_tknzr.from_pretrained(txt_sum_bert_mdl_nm)
  txt_sum_bert_mdl =
txt_sum_bert_mskd_lm.from_pretrained(txt_sum_bert_mdl_nm)
  txt_sum_bert_inputs = txt_sum_toknzr(txt_sum_inpt_txt, return_tensors='pt',
truncation=True, max_length=txt_sum_bert_mx_lth, padding=True)
  with txt_sum_trch.no_grad():
      txt_sum_outputs = txt_sum_bert_mdl.generate(txt_sum_bert_inputs.input_ids,
attention_mask=txt_sum_bert_inputs.attention_mask,
max_length=txt_sum_bert_mx_lth)

  txt_sum_output = txt_sum_toknzr.decode(txt_sum_outputs[0],
skip_special_tokens=True)
  return txt_sum_output


txt_sum_output_lis = []
for txt in txt_sum_data_frm['Proceedings Content']:
  txt_sum_smrzd_opt = bert_txt_summarizer(txt)
  txt_sum_output_lis.append(txt_sum_smrzd_opt)

txt_sum_opt_dta_frm = txt_sum_pd.DataFrame()
```

```
for opt in range(len(txt_sum_output_lis)):
  txt_sum_pre =
txt_sum_pd.Series({'Proceeding_content':txt_sum_data_frm['Proceedings
Content'].unique()[opt],'Bert_summarized_output':txt_sum_output_lis[opt]})
  txt_sum_opt_dta_frm =
txt_sum_opt_dta_frm.append(txt_sum_pre,ignore_index=True)
txt_sum_opt_dta_frm['Category']=txt_sum_data_frm['Category']
txt_sum_opt_dta_frm

txt_sum_opt_dta_frm['Bert_summarized_output'].str.len()

Evaluation using Rougue Score

txt_sum_mdl_nm = "bert-base-uncased"

txt_sum_smrzr = txt_sum_rg_ppln("summarization", model=txt_sum_mdl_nm)

row_index = 0

txt_sum_opt_smry =
txt_sum_opt_dta_frm['Bert_summarized_output'].iloc[row_index]
txt_sum_inpt_ex = txt_sum_opt_dta_frm["Proceeding_content"].iloc[row_index]

txt_sum_scr = txt_sum_rg_scr.RougeScorer(['rouge1', 'rouge2', 'rougeL'],
use_stemmer=True)
txt_sum_scrs = txt_sum_scr.score(txt_sum_inpt_ex, txt_sum_opt_smry)

for txt_sum_mtrc, txt_sum_scr in txt_sum_scrs.items():
   print(f"{txt_sum_mtrc}:")
   print(f"  Precision: {txt_sum_scr.precision:.4f}")
   print(f"  Recall: {txt_sum_scr.recall:.4f}")
   print(f"  F1 Score: {txt_sum_scr.fmeasure:.4f}")
```

4- T5

```
import sentencepiece
from transformers import T5Tokenizer as txt_sum_t5_tknzr
from transformers import T5ForConditionalGeneration as txt_sum_t5_cg

#### Summarizing one sentence from the data using T5 as an example to understand
the before and after results of summarization

txt_sum_ex_text = txt_sum_data_frm["Proceedings Content"][0]
print(len(txt_sum_ex_text))

def t5_txt_summarizer(txt_sum_inpt_txt):
```

```
  txt_sum_t5_mdl_nm = "t5-small"
  txt_sum_t5_mx_lth=200
  txt_sum_toknzr = txt_sum_t5_tknzr.from_pretrained(txt_sum_t5_mdl_nm)
  txt_sum_t5_mdl = txt_sum_t5_cg.from_pretrained(txt_sum_t5_mdl_nm)
  txt_sum_t5_inputs = txt_sum_toknzr.encode("summarize: " + txt_sum_inpt_txt,
return_tensors="pt", max_length=1024, truncation=True)
  txt_sum_summary_ids = txt_sum_t5_mdl.generate(txt_sum_t5_inputs,
max_length=txt_sum_t5_mx_lth, num_beams=4, early_stopping=True)
  txt_sum_output = txt_sum_toknzr.decode(txt_sum_summary_ids[0],
skip_special_tokens=True)
  return txt_sum_output


txt_sum_ex_output = t5_txt_summarizer(txt_sum_ex_text)

print(len(txt_sum_ex_output))

txt_sum_mdl_nm = "t5-small"
txt_sum_smrzr = txt_sum_rg_ppln("summarization", model=txt_sum_mdl_nm)

txt_sum_opt_smry = txt_sum_ex_output
txt_sum_inpt_ex = txt_sum_ex_text

txt_sum_scr = txt_sum_rg_scr.RougeScorer(['rouge1', 'rouge2', 'rougeL'],
use_stemmer=True)
txt_sum_scrs = txt_sum_scr.score(txt_sum_inpt_ex, txt_sum_opt_smry)

for txt_sum_mtrc, txt_sum_scr in txt_sum_scrs.items():
    print(f'{txt_sum_mtrc}:')
    print(f'  Precision: {txt_sum_scr.precision:.4f}')
    print(f'  Recall: {txt_sum_scr.recall:.4f}')
    print(f'  F1 Score: {txt_sum_scr.fmeasure:.4f}')

#### Summarizing all the sentences in the data


# defining the model to summarize the text data

def t5_txt_summarizer(txt_sum_inpt_txt):
  txt_sum_t5_mdl_nm = "t5-small"
  txt_sum_t5_mx_lth=200
  txt_sum_toknzr = txt_sum_t5_tknzr.from_pretrained(txt_sum_t5_mdl_nm)
  txt_sum_t5_mdl = txt_sum_t5_cg.from_pretrained(txt_sum_t5_mdl_nm)
  txt_sum_t5_inputs = txt_sum_toknzr.encode("summarize: " + txt_sum_inpt_txt,
return_tensors="pt", max_length=1024, truncation=True)
  txt_sum_summary_ids = txt_sum_t5_mdl.generate(txt_sum_t5_inputs,
max_length=txt_sum_t5_mx_lth, num_beams=4, early_stopping=True)
```

```
  txt_sum_output = txt_sum_toknzr.decode(txt_sum_summary_ids[0],
skip_special_tokens=True)
  return txt_sum_output

txt_sum_output_lis = []
for txt in txt_sum_data_frm['Proceedings Content']:
  txt_sum_smrzd_opt = t5_txt_summarizer(txt)
  txt_sum_output_lis.append(txt_sum_smrzd_opt)

txt_sum_opt_dta_frm = txt_sum_pd.DataFrame()
for opt in range(len(txt_sum_output_lis)):
  txt_sum_pre =
txt_sum_pd.Series({'Proceeding_content':txt_sum_data_frm['Proceedings
Content'].unique()[opt],'T5_summarized_output':txt_sum_output_lis[opt]})
  txt_sum_opt_dta_frm =
txt_sum_opt_dta_frm.append(txt_sum_pre,ignore_index=True)
txt_sum_opt_dta_frm['Category']=txt_sum_data_frm['Category']
txt_sum_opt_dta_frm

txt_sum_opt_dta_frm['T5_summarized_output'].str.len()

Evaluation using Rougue Score

txt_sum_mdl_nm = "t5-small"
txt_sum_smrzr = txt_sum_rg_ppln("summarization", model=txt_sum_mdl_nm)

row_index = 0

txt_sum_opt_smry =
txt_sum_opt_dta_frm['T5_summarized_output'].iloc[row_index]
txt_sum_inpt_ex = txt_sum_opt_dta_frm["Proceeding_content"].iloc[row_index]

txt_sum_scr = txt_sum_rg_scr.RougeScorer(['rouge1', 'rouge2', 'rougeL'],
use_stemmer=True)
txt_sum_scrs = txt_sum_scr.score(txt_sum_inpt_ex, txt_sum_opt_smry)

for txt_sum_mtrc, txt_sum_scr in txt_sum_scrs.items():
   print(f'{txt_sum_mtrc}:')
   print(f'  Precision: {txt_sum_scr.precision:.4f}')
   print(f'  Recall: {txt_sum_scr.recall:.4f}')
   print(f'  F1 Score: {txt_sum_scr.fmeasure:.4f}')
```

5- GPT2

```
from transformers import GPT2Tokenizer as txt_sum_gpt2_tknzr
from transformers import GPT2LMHeadModel as txt_sum_gpt2_lm_hd
```

```
#### Summarizing one sentence from the data using GPT2 as an example to
understand the before and after results of summarization

txt_sum_ex_text = txt_sum_data_frm["Proceedings Content"][0]
print(len(txt_sum_ex_text))

def gpt2_txt_summarizer(txt_sum_inpt_txt):
  txt_sum_gpt2_mdl_nm ="gpt2"
  txt_sum_gpt2_mx_lth=150
  txt_sum_toknzr = txt_sum_gpt2_tknzr.from_pretrained(txt_sum_gpt2_mdl_nm)
  txt_sum_gpt2_mdl = txt_sum_gpt2_lm_hd.from_pretrained(txt_sum_gpt2_mdl_nm)
  txt_sum_gpt2_inputs = txt_sum_toknzr(txt_sum_inpt_txt, return_tensors='pt',
truncation=True, max_length=txt_sum_gpt2_mx_lth)
  with txt_sum_trch.no_grad():
     txt_sum_outputs = txt_sum_gpt2_mdl.generate(txt_sum_gpt2_inputs.input_ids,
attention_mask=txt_sum_gpt2_inputs.attention_mask,
max_length=txt_sum_gpt2_mx_lth)

  txt_sum_output = txt_sum_toknzr.decode(txt_sum_outputs[0],
skip_special_tokens=True)
  return txt_sum_output


txt_sum_ex_output = gpt2_txt_summarizer(txt_sum_ex_text)

print(len(txt_sum_ex_output))

txt_sum_mdl_nm = "gpt2"
txt_sum_smrzr = txt_sum_rg_ppln("summarization", model=txt_sum_mdl_nm)

txt_sum_opt_smry = txt_sum_ex_output
txt_sum_inpt_ex = txt_sum_ex_text

txt_sum_scr = txt_sum_rg_scr.RougeScorer(['rouge1', 'rouge2', 'rougeL'],
use_stemmer=True)
txt_sum_scrs = txt_sum_scr.score(txt_sum_inpt_ex, txt_sum_opt_smry)

for txt_sum_mtrc, txt_sum_scr in txt_sum_scrs.items():
   print(f"{txt_sum_mtrc}:")
   print(f"  Precision: {txt_sum_scr.precision:.4f}")
   print(f"  Recall: {txt_sum_scr.recall:.4f}")
   print(f"  F1 Score: {txt_sum_scr.fmeasure:.4f}")


#### Summarizing all the sentences in the data
```

```
# defining the GPT2 model to summarize the text data

def gpt2_txt_summarizer(txt_sum_inpt_txt):
  txt_sum_gpt2_mdl_nm ="gpt2"
  txt_sum_gpt2_mx_lth=150
  txt_sum_toknzr = txt_sum_gpt2_tknzr.from_pretrained(txt_sum_gpt2_mdl_nm)
  txt_sum_gpt2_mdl = txt_sum_gpt2_lm_hd.from_pretrained(txt_sum_gpt2_mdl_nm)
  txt_sum_gpt2_inputs = txt_sum_toknzr(txt_sum_inpt_txt, return_tensors='pt',
truncation=True, max_length=txt_sum_gpt2_mx_lth)
  with txt_sum_trch.no_grad():
     txt_sum_outputs = txt_sum_gpt2_mdl.generate(txt_sum_gpt2_inputs.input_ids,
attention_mask=txt_sum_gpt2_inputs.attention_mask,
max_length=txt_sum_gpt2_mx_lth)

  txt_sum_output = txt_sum_toknzr.decode(txt_sum_outputs[0],
skip_special_tokens=True)
  return txt_sum_output

txt_sum_output_lis = []
for txt in txt_sum_data_frm['Proceedings Content']:
  txt_sum_smrzd_opt = gpt2_txt_summarizer(txt)
  txt_sum_output_lis.append(txt_sum_smrzd_opt)

txt_sum_opt_dta_frm = txt_sum_pd.DataFrame()
for opt in range(len(txt_sum_output_lis)):
  txt_sum_pre =
txt_sum_pd.Series({'Proceeding_content':txt_sum_data_frm['Proceedings
Content'].unique()[opt],'GPT2_summarized_output':txt_sum_output_lis[opt]})
  txt_sum_opt_dta_frm =
txt_sum_opt_dta_frm.append(txt_sum_pre,ignore_index=True)
txt_sum_opt_dta_frm['Category']=txt_sum_data_frm['Category']
txt_sum_opt_dta_frm

txt_sum_opt_dta_frm['GPT2_summarized_output'].str.len()

Evaluation using Rougue Score

txt_sum_mdl_nm = "gpt2"
txt_sum_smrzr = txt_sum_rg_ppln("summarization", model=txt_sum_mdl_nm)

row_index = 0

txt_sum_opt_smry =
txt_sum_opt_dta_frm['GPT2_summarized_output'].iloc[row_index]
txt_sum_inpt_ex = txt_sum_opt_dta_frm["Proceeding_content"].iloc[row_index]
```

```
txt_sum_scr = txt_sum_rg_scr.RougeScorer(['rouge1', 'rouge2', 'rougeL'],
use_stemmer=True)
txt_sum_scrs = txt_sum_scr.score(txt_sum_inpt_ex, txt_sum_opt_smry)

for txt_sum_mtrc, txt_sum_scr in txt_sum_scrs.items():
    print(f"{txt_sum_mtrc}:")
    print(f"  Precision: {txt_sum_scr.precision:.4f}")
    print(f"  Recall: {txt_sum_scr.recall:.4f}")
    print(f"  F1 Score: {txt_sum_scr.fmeasure:.4f}")
```