# A Comprehensive Evaluation of LLM and Pre-trained Models Across Diverse Datasets

*Internship report submitted by*

**Ankush Sil Sarma (002011001042)**

**Kuntal Das (002011001032)**

**Souvik Bhattacharjee (302111001006)**

*Under the guidance of*

**Ms. Tohida Rehman**

**Department of Information Technology,**

**Faculty of Engineering and Technology,**

**Jadavpur University, Salt Lake Campus**

**Kolkata, India**

**2020-2024**

**Department of Information Technology**
**Faculty of Engineering and Technology**
**Jadavpur University**

# BONAFIDE CERTIFICATE

This is to certify that this internship report entitled **“A Comprehensive Evaluation of LLM and Pretrained Models Across Diverse Datasets”** submitted to **Department of Information Technology, Jadavpur University, Salt Lake Campus, Kolkata**, is a bonafide record of work done by **“Ankush Sil Sarma (Regn no:153773 of 2020-21), Kuntal Das (Regn no:153764 of 2020-21), Souvik Bhattacharjee (Regn no:159992 of 2021-22) ”** under my supervision from **15.11.2023** to **20.05.2024**

**Ms. Tohida Rehman**
Assistant Professor
Department of Information Technology
Jadavpur University, Kolkata

# Declaration

We, Ankush Sil Sarma, Kuntal Das, Souvik Bhattacharjee hereby declare that this internship report titled **“A Comprehensive Evaluation of LLM and Pre-trained Models Across Diverse Datasets”** contains only the work completed by us under the supervision of Ms. Tohida Rehman, Department of Information Technology, Jadavpur University,Kolkata, India. All information, materials and methods that are not original to this work have been properly referenced and cited. All information in this document has been obtained and presented in accordance with academic rules and ethical conduct.

**Anksuh Sil Sarma**
Roll No.- 002011001042

**Kuntal Das**
Roll No.- 002011001032

**Souvik Bhattacharjee**
Roll No.- 302111001006

Place: Kolkata
Date: 21/05/2024

# Acknowledgments

We want to seize this moment to convey our sincerest thanks and profound appreciation to everyone who has provided us with direct or indirect assistance, contributing to the triumphant accomplishment of this internship.

First and foremost, we extend our heartfelt appreciation to our esteemed mentor and teacher, Ms. Tohida Rehman (Assistant Professor, Department of Information Technology, Jadavpur University), for her exceptional guidance, unwavering supervision, and constant motivation throughout the duration of this.

Furthermore, we would like to extend our gratitude to Prof. Bibhas Chandra Dhara, (the Head of the Department of Information Technology at Jadavpur University), for generously providing invaluable support for the successful execution of this internship.

Finally, we would like to express our heartfelt appreciation to our parents, fellow classmates from the B.E. Information Technology batch of 2020-24, and all the dedicated faculty members in the Department of IT for their unwavering cooperation and support. Their vast reservoir of experience has served as a constant source of inspiration.

Together, the collective efforts of all these individuals have paved the way for the accomplishment of this internship, and for that, we are immensely grateful.

**Department of Information Technology**
**Jadavpur University**
**Salt Lake Campus, Kolkata**

| | | |
|---|---|---|
| **Ankush Sil Sarma** | **Kuntal Das** | **Souvik Bhattacharjee** |
| Roll No-002011001042 | Roll No-002011001032 | Roll No-302111001006 |

# Abstract

Text summarization is a fundamental task in natural language processing that aims to condense large amounts of textual information into concise and coherent summaries. With the exponential growth of content and the need to extract key information efficiently, text summarization has gained significant attention in recent years. In this study, we present a comprehensive evaluation of four prominent pre-trained language models and open version large language models, BART, FLAN-T5, Llama-3-8B, and Gemma-7B across five diverse text summarization datasets, CNN/DM, Gigaword, News Summary, XSum, and BBC News. Our objective is to systematically compare these models using a set of established automatic evaluation metrics such as ROUGE-1, ROUGE-2, and ROUGE-L, BERTScore, and METEOR. Each model is evaluated for its ability to generate coherent and informative summaries, with results highlighting the relative strengths and weaknesses of each model in handling different types of text data. The findings from this evaluation provide valuable insights into the performance and applicability of these models for summarization tasks, informing best practices

and guiding future research in the field of natural language processing.

***Keywords***: *Text Summarization, Evolution metrics, Pre-trained Models, Natural language generation*

# Table of Contents

# List of Figures

# List of Tables

# 1 Introduction

## 1.1 Motivation

The field of natural language processing (NLP) has witnessed remarkable advancements in recent years, revolutionizing various domains such as information retrieval, sentiment analysis, and machine translation. Among the key areas within NLP, text summarization plays a crucial role in condensing documents into concise summaries while preserving the essential meaning. Abstractive text summarization, in particular, aims to generate summaries that not only extract crucial information but also generate new sentences that capture the overall essence of the original text.

## 1.2 Research Goal and Contribution

### 1.2.1 Research Goal

The primary goal of this research is to systematically evaluate and compare the performance of four advanced pre-trained language models in the context of abstractive text summarization across a range of diverse datasets. By employing a comprehensive set of evaluation metrics, we aim to elucidate the relative strengths and weaknesses of each model. This research seeks to provide a deeper understanding of model capabilities and limitations, guiding the selection and development of more effective summarization techniques for various applications.

### 1.2.2 Research Contribution

We conduct extensive experiments and evaluations to ascertain the effectiveness of various pretrained large language models (LLMs) in the domain of abstractive text summarization. We juxtapose the performance of these advanced models against state-of-the-art methodologies, undertaking meticulous analyses of summary coherence and the retention of pivotal information. Through these evaluations, we endeavor to substantiate the efficacy and practicality of our selected LLMs, while elucidating their respective strengths and limitations. This exhaustive assessment enhances our comprehension of model behavior across diverse summarization datasets, thereby informing the development and deployment of more resilient summarization techniques.

## 1.3 Organization of the project

The project is organized into several distinct phases to achieve our research goal.

In the initial phase, we conduct an extensive literature review to understand the existing methodologies and challenges in abstractive text summarization. This provides us with the necessary back-
ground knowledge to identify the gaps and limitations that we aim to address.

Next, we concentrate on dataset preparation and preprocessing. This involves curating the CNN/DM, Gigaword, News Summary, XSum, and BBC News datasets, performing data cleaning and normalization, and applying appropriate techniques for text representation and feature extraction.

Following that, we focus on the application and evaluation of existing advanced models in abstractive text summarization, specifically BART, FLAN-T5, Llama-3-8B, and Gemma-7B. Through comprehensive experimentation, we aim to enhance the fidelity and quality of the generated summaries,
determining the most effective approaches for improving summary coherence and informativeness.

In the final phase, we evaluate and compare our proposed models against existing state-of-the-art approaches using appropriate evaluation metrics. We analyze the results, draw meaningful conclu-sions, and discuss the implications of our findings.

By organizing the project in this manner, we aim to address the research goal of abstractive text summarization and make a significant contribution to the field.

# 2 Literature Survey

## 2.1 Modeling Based Extractive Text Summarization

### 2.1.1 Single Document Summarization

During the initial research in text summarization,summarizing single documents was the primary focus of scholars. The technique of identifying salient information from a single text document, thus compressing and summarizing it, is called single-document summarization. A lot of the state-of-theart models for single document summarization limit their functioning to short length text and hence do not produce encouraging results when applied over longer length text.Romain P. et al. [1] propose a neural network model having intra-attention to generate abstractive summaries for input documents. They have used reinforcement learning and a supervised algorithm to predict words for creating novel phrases for their abstractive summary. With this combination, they claim to generate more readable summaries. However, they have trained and evaluated their model on different datasets than ours, using the standard CNN/DM and New York Times datasets.

An extractive summarization model on a single text document as a tree induction scenario is formulated by Yang L. et al. [2]. Here, the root nodes in the tree represent summary sentences. Each root node has attached sub-trees that contain phrases or words to elaborate on the root node summary sentence. Their model is iterative in nature which helps to generate better summaries in each iteration, but again, they limit their evaluation to the standard datasets used by Romain P. et al. in [1], i.e. CNN/DM and New York Times datasets.

A single document extractive summarization is conceptualized by Sanchit et al. [3]. It uses sentence embeddings and K-Means clustering. Using K-Means clustering, they form clusters of text for each document, such that the intra-cluster similarity is higher than the inter-cluster similarity. Following this, the most informative and important sentence from each cluster is chosen to generate an

extractive summary for a given source document.

Naveen S. et al. [4] present an interesting model for generating single document extractive summaries by considering the text summarization problem to be one of the binary optimization problems, which uses MOBDE (Multi-Objective Binary Differential Evolution) to generate text summaries. They have used various statistical features such as coverage, sentence length, the position of a sentence in the document, cohesion, similarity between title and sentence, and readability score, to evaluate summary sentences. Moreover, when evaluating their model on DUC2001 and DUC2002 standard

text summarization datasets, they obtain comparative results to other state-of-the-art techniques.

### 2.1.2 Multi-Document Summarization

The text summarization approach which generates a summary for a topic after considering multiple documents about the topic is called multi-document summarization. In this summarization technique,
relevantsentencesforthetopicareeitherpickedup(forextractivetextsummarization)orarerephrased (for abstractive text summarization) after performing multi-document analysis. The sentences are then combined to form a representative summary for the set of documents relevant to a topic.

In Jian-Ping M. et al [5] developed a multi-document summarization framework to generate extractive summaries. They introduced two new features to calculate the scores of sentences in the summaries. The first being 'Exemplar', a feature to balance the relevance and coverage in their summariesandthesecondbeing'Position', tomeasuretherelativepositionofeachsentenceinadocument. However, along with evaluating their model on the standard DUC datasets (DUC2004, DUC2005, DUC2006), they have used the ROUGE-2 and ROUGE-SU4 scores to compare their summaries with those generated by the other state-of-the-art techniques. Moreover, this comparison shows a very marginal increase in the performance of their approach.

An evolutionary algorithm is proposed by Rasim M. A. et al. [6] to perform summarization on multiple documents.They claim their model to have a superior correlation between sentences with a low rate of redundancy.Nevertheless, their model requires a large computational overhead and is experimented over the standard DUC2002 and DUC2004 datasets to get viable results. Aiming to have good topic coverage, this approach cannot be directly applied to WikiHow dataset for generating topic-based extractive summaries.

A multi-document summarization technique to summarize documents containing events that have occurred in the past at different times was introduced in [7] by Giang T. et al. This simple timeline summarization approach aims to summarize prolonged events such as economic crises and war. Though it achieves good performance on timeline data, this method will not present equivalent results on the WikiHow dataset. This is because the task of forming text clusters on a well-defined timeline dataset and generating summaries is different and less challenging than performing a similar task on a dataset with a high diversity of topics and novel writing style.

### 2.1.3 Topic Modeling Summarization

The objective of this summarization approach is to form clusters of text from the content present in a source document and summarize the clusters to generate the document summary. In order to achieve good results in this summarization approach, the source document must have a high topic-diversity. Various topic modeling techniques exist that help to enhance the topic selection quality.

A novel topic augmented abstractive summarization method that identifies the topics in the source document using LDA has been introduced by Melissa A. et al. [8]. Moreover, they have used CNN/DM and WikiHow datasets for evaluation. They have performed five different implementations, i.e., (i) Pointer-Generator, (ii) TAG, (iii) Pointer-Generator + Coverage, (iv) TAG + Coverage and (v) Lead-3, on the WikiHow dataset. Our model outperforms (i),(ii), (v) and gets an equivalent ROUGE for (iii).

Liu N. et al. [9] propose an algorithm to summarize multi-document text using LDA topic modeling. Among the topics identified by LDA, their model selects essential topics based on weight criteria. They have also used statistical features such as sentence length, frequency of terms and sentence position to train their models. Evaluating their model on the standard DUC2002 corpus, they claim to
achieve stronger results than other state-of-the-art algorithms.

An unsupervised approach is presented by Lu W. et al. [10], to summarize spoken meetings using topic modeling.Their method exceeds the summarization results of existing models and eliminates redundancies. Their token-level summarization framework uses utterance-level topic modeling and is claimed to perform better than other document-level models.

Zongda W. et al. [11] introduces a robust framework to summarize novel documents. This well explained research work uses topic modeling to create text clusters from which salient candidate sentences are selected based upon importance scores, to generate text summaries. Trained directly on 63 narrative novels, this model is suitable mainly for summarizing long-form novel datasets and is not appropriate for summarizing documents present in the WikiHow dataset. Post generating their initial
summaries,they also smooth their summaries to make it more readable

An approach to extractive text summarization using binary classification modeling and topic-based sentence extraction using LDA is proposed in the research work by Nikolaos G. et al. [12].

Their evaluation of the MultiLing 2015 MSS dataset proves that topic modeling helps generate better extractive text summaries. However, they use supervised learning to perform binary classification on sentences, where one class holds sentences to be included in the summary and the other class includes the sentencesthatshouldnotbepresentinthesummary. Contrarytoit, weproposeacompletelyunsupervised approach to generated extractive summaries.

## 2.2 Abstractive text summarization

Text summarization is a fundamental task in Natural Language Processing (NLP) that aims to condense lengthy textual documents into concise and coherent summaries. Over the years, researchers have made significant progress in developing techniques and models to tackle this challenging problem. In this literature survey, we provide an overview of the key approaches, methodologies, and trends in the field of text summarization.

Early research efforts in text summarization predominantly focused on extractive methods, which involve selecting the most important sentences or phrases from the source document to form the summary. Abstractive text summarization gained attraction as early as the 1950s.

Luhn et al., 1958 [13] introduced one of the earliest extractive approaches based on the frequency of occurrence of [14] important words or phrases. They proposed to weight the sentences of a document as a function of high frequency words, ignoring very high frequency common words. Edmundson et al., 1969 [15] described a paradigm based on key phrases which in addition to standard frequency depending weights, used three methods to determine the sentence weight. Since then, many works have been published to address the problem of automatic text summarization (see [16], [17] for more information about more advanced techniques until 2000s).

Following this, numerous statistical and graph-based algorithms, such as LexRank proposed by Radev et val. [17]) and TextRank (Mihalcea and Tarau, 2004 [18]), were proposed to identify salient sentences through graph centrality measures.

With the advent of machine learning techniques, researchers began exploring supervised methods for extractive summarization. Supervised approaches involve training models on labeled data, where each sentence is annotated with its importance for summarization. Filatova and Hatzivassiloglou, 2004 [19],utilized Support Vector Machines (SVM) for extractive summarization, achieving promising results in identifying informative sentences based on various features such as sentence length, position, and content-specific features. Radev et al., 2004 [20] proposed Maximum Entropy (MaxEnt) models for sentence extraction, further improving the effectiveness of supervised approaches.

While extractive summarization approaches achieved reasonable success, they often struggled to handle information that required rephrasing or merging content from multiple sentences. This limitation led to the emergence of abstractive summarization methods, which aim to generate summaries by understanding the source text and generating new sentences that capture the essential information. Notable landmark moments in abstractive summarization include the introduction of sequence-to-
sequence models with attention mechanism.

Due to the difficulty of abstractive summarization, the great majority of past work has been extractive (Kupiec et al., 1995 [21]; Paice, Saggion and Poibeau, 2013 [22]). However, the recent success of sequence-to-sequence models (Sutskeveret al., 2014 [23]), in which recurrent neural networks (RNNs) both read and freely generate text, has made abstractive summarization viable (Chopra et al., 2016 [24]; Nallapati et al., 2016 [25] ; Rush et al., 2015 [26]; Zeng et al., 2016). Though these systems are promising, they exhibit undesirable behavior such as inaccurately reproducing factual
details, an inability to deal with out-of-vocabulary (OOV) words, and repeating themselves.

Paulus et al., 2017 [27] presented an influential work on abstractive summarization using the sequence-to-sequence framework. Their model incorporated attention mechanisms to generate informative and coherent summaries. See et al., 2017 [28] proposed Pointer-Generator Networks, which combined extractive and abstractive approaches by allowing the model to copy words directly from
the source text when necessary.

These advancements in abstractive summarization have been driven by deep learning and neural network-based models. Researchers have explored various architectures, such as Transformer models [29], to capture the contextual information and improve the coherence of generated summaries. Recent works, such as Pegasus by Zhang et al., 2020 [30] and BART by Lewis et al., 2020 [31], have further advanced abstractive summarization by leveraging pre-training techniques and large-scale language models.

## 2.3 Hybrid Approach for Single Text Document Summarization

Text document summarization playing an important role in IR (Information Retrieval) because, it condense a large pool of information into a concise form, through selecting the salient sentences and discards redundant sentences (or information) and we termed it as summarization process.

Radev et al. in [32] has defined a text summary as “a text that is produced from one or more texts that convey important information in the original texts, and that is no longer than half of the original text and usually significant less than that”. As explained in [33] Automatic text document summarization is an interdisciplinary research area of computer science that includes AI (artificial intelligence), DataMining, StatisticsaswellasPsychology. Wecanclassifytextdocsummarizationintwoways(by techniques) Abstractive summarization and Extractive summarization. Abstractive summarization is more human like a summary, which is the actual goal of Text document summarization. As defined by [34] abstractive summarization needs three things as Information Fusion.Sentences Compression and Reformation. Abstractive summarization may contain new sentences, phrases, words even which are not present in the source document. Although till now a lot of research in happened in the last decades in the area of NLP (Natural language processing), NLG (Natural Language Generation), so much computing power increased, but still we are not near for abstractive summarization. The actual challenge is a generation of new sentences, new phases, along with produced summary must retain the same meaning as the same source document has. Extractive summarization based on extractive entities, entities may be sentence, sub part of sentence, phrase or a word. Our work is focused on
extractive based technique.

According to M. Ramiz summarization in [33] is defined as a three steps process (1) Analysis of text. (2) Transformation- as summary representation, and (3) Synthesis- produce an appropriate summary. Eduard Hovy and Chin-Yew Lin [35] introduced SUMMARIST system to create a robust text summarization system, system that works on three phases which can describe in form of an equation like “Summarization = Topic Identification + Interpretation + Generation”.

The early work in Document Summarization started on ”Single Document Summarization”, by H. P. Luhn [36], he proposed a frequency based model, in which frequency of word play crucial role, to decide importance of any sentence in the given document. Another work of P. Baxendale [37], had been introduced a statistical model based on sentence position. In his research, he found that, starting and ending sentences became salient sentences for summarization, but this is not better for every document like scientific research paper but good for newspapers summarization. H.P. Edmondson [38] also proposed an effective technique for document summarization. At first,Edmonson designed some rules for manual extraction, then rules were applied to 400 technical documents.Edmondson considers four features (1) sentences position, (2) frequency of word, (3) presence of cue words, and (4) the skeleton of the document. The work was done almost manual. After these early work lot of work done in this discipline some are available in , here in the next section we are presenting only work done in recent years.

Query focused summarization is a special case of document summarization, in which summary purposely demands, to be biased according to the user query. You Ouyang et.al [39] used SVR (Support Vector Regression) to calculate the importance of the sentences in a given document. Another

query focused summarization, multi-document summarization proposed by Carbonell et al. [40].

Researcher done a lot of work in the multi document summarization field and many more like in considering this is as a global/multi optimization problem which requires simultaneous optimization of more than one objective function. Rasim et al. [6], in which the objective function is a weighted combinationof(1)contentcoverage, and(2)forredundancyobjectives. Inanotherworktheyproposed CDDS based summarization with two objectives diversity and coverage. In summarization, similarity evaluation among of sentences is a laborious task because of (1) complex sentence structure, and (2) lack of extra information so, Ming Che Lee [41] had been proposed “Transformed Vector Space” model based on Word-Net. Due to improper ordering of information there is possibility that it can confuse the readers as well as can degrade the readability of the summary. To maintain the association and order of sentences, Danushka Bollegala et.al [42] defined four criteria (1) chronology, (2) Topicalcloseness (3) Precedence (4) Succession. These all four criteria are combined into a single criteria by using a supervised learning approach.

Redundancy can be defined as a multiplicity of sentences, sub sentences or information. Coverage and redundancy are reciprocal to each other. Summarization objective is maximum coverage and minimum redundancy. Sarkar.K et al [43] gave a simple approach to include a sentence in summary one by one based on modified cosine similarity threshold. To include sentences in the summary, system first selects the most top rank sentence, include it in the summary and this process is repeated for remaining sentences. Next sentence is included in the summary if similarity between sentences and summary is less than some threshold, otherwise the sentence is not included in the summary and algorithm stops when required summary length is reached. We are using this model in our implementation to handle redundancy. Another model MMR [21] is popularly used (especially in with a given query) to reduce redundancy. The MMR (Maximal Marginal Relevance) criteria, ”strives to reduce redundancy while maintaining query relevance in re-ranking retrieved documents and in selecting appropriate passages for text summarization”. This technique gives better result for Multi Document Summarization. Rasim M. Alguliev et.al [33] proposed an unsupervised text summarization model which can be used for Single as well as Multi Document Summarization. In simple terms, the problem is treated as a Multi objective problem, where the objective is to optimize three objectives (1) Relevance, (2) Redundancy and (3) Length.

## 2.4 Summarization with LLMs

It is widely acknowledged that LLMs have propelled forward abstractive summarization research [44], with their summaries being highly rated by human annotators [45]; [46]. Liu et al. [47] proposes to train smaller models like BART [48] or BRIO [49] with contrastive learning using LLMs like ChatGPT as evaluator providing signal on which generated summary candidate is better.Summary chain-of-thought designs a custom chainof-thought method which first prompts the LLM to list important facts, then integrates these facts into a coherent summary [50]. SummIt utilizes ChatGPT to iteratively write then refine summaries given feedback from an evaluator LLM [46]. Chain-ofdensity gradually makes GPT-4 generated summaries contain more and more entities while keeping length budget constant, creating more informative albeit a bit less readable summaries [51]. Ravaut et al. [52] noticed that data points with higher compression are generally harder to summarize with pre-trained models.

Sun et al. [53] showed that for Transformer-based models, most recent tokens play a greater role compared to older tokens for next-token prediction. It was later found that for in-context learning, the orderofexampleswithinthepromptimpactsGPT-3'sperformance.Relianceonpositionalinformation affects LLMs capabilities in arithmetic [54], in multiplechoice question answering [55], and as text generation evaluators [50] ; making it hard to rank LLMs [56]. Liu et al. [47] was the first to show that LLMs' performance weakens in the middle of the prompt, yet,how LLMs make use of their full context window remains poorly understood. The passkey retrieval evaluation, which consists in prompting the LLM to recall a complex string or long number inserted in its prompt, is becoming popular recently as a way of verifying LLM's processing capability at each position [57] However, this task does not measure position bias on complex, abstract reasoning tasks like summarization. A line of work attempts to solve the middle curse through compressing the prompt , with very promising results albeit at the cost of prompt fluency. Another approach marginalizes results over different permutations of the input to suppress dependency on input order [58]. Concurrent work to ours also finds that in zero-shot summarization, LLMs tend to prefer lead content [59].

In the realm of abstractive long text summarization, several prominent transformer-based models have played a pivotal role in advancing the field. These models, including BERT, GPT-2, and XLNet, each bring their distinctive architectural strengths to the table. Notably, Google's BERT (Bidirectional En-coder Representation from Transformers) has proven to be a versatile choice for various NLP tasks. BERT operates on the foundation of the Masked Language Model (MLM) technique, transforming
text into word embeddings based on statistically derived similarity measures.

To assess the quality of summaries generated by the BERT model, researchers often employ a set of standard evaluation metrics known as ROGUE, comparing the output with human-generated summaries.In contrast,BART (Bidirectional Auto-Regressive Transformer) has gained recognition for surpassing BERT in performance.BART introduces a novel pre-training method and architecture that enables it to function as a sequence-to-sequence model (seq2seq model) for diverse NLP tasks.

The Generative Pre-trained Transformer(GPT) family, initiated by OpenAI in 2018, has made remarkable strides in generating realistic text. Among its various iterations, including GPT-3 and GPT-3.5,the original GPT model was trained with 117 million parameters , aiming to comprehend language in a manner akin to human understanding. GPT-2, equipped with even more training parameters , was introduced to address the limitations of its predecessor. In 2023,the latest iteration, GPT-4, emerged, further building upon the strengths of GPT-3 and delivering improved performance.

# 3 Methodology

In our methodology, we employed both large language models (LLMs) and pre-trained models, fine-tuning them to enhance their ability to generate efficient and concise summaries. This approach involved adjusting each model to optimize performance on diverse datasets, ensuring that the generated summaries were not only accurate but also coherent and informative. By leveraging the strengths of both LLMs and pre-trained models, we aimed to achieve high-quality summary generation across
various contexts.

## 3.1 Abstractive summarization with BART

First, we downloaded the KerasNLP GitHub repository to access the BART model, a highly advanced transformer-based architecture specifically designed for sequence-to-sequence tasks such as text summarization. BART, which boasts 139 million trainable parameters, utilizes a unique denoising autoencoder approach that enables it to generate high-quality, coherent summaries by reconstructing text with impressive accuracy. This model is pre-trained on large corpora, making it exceptionally proficient at understanding and generating natural language.

To tailor BART for our specific summarization tasks, we installed the necessary datasets for finetuning. These datasets were carefully selected to cover a diverse range of topics and text styles, ensuring that the model could generalize well across different types of content. Next, we meticulously preprocessed these datasets, involving steps such as tokenization, normalization, and removing extraneous elements to ensure compatibility with the model and enhance its performance.

The fine-tuning process was a critical phase where we trained the BART model for three epochs. During this process, we adjusted the model's parameters to optimize its summarization capabilities, allowing it to generate concise, informative, and contextually accurate summaries. This training phase was conducted in a high-performance computing environment to handle the substantial computational demands efficiently. By refining BART's parameters and leveraging its powerful architecture, we significantly enhanced its ability to produce efficient and compact summaries, making it a robust tool for various text summarization applications.

## 3.2 Abstractive summarization with FLAN-T5

First, we download the necessary libraries and packages required for our project. This includes setting up our environment with dependencies essential for natural language processing tasks. Next, we acquire the required summarization datasets from Hugging Face, a reliable repository for diverse and high-quality datasets. Once the datasets are downloaded, we split them into training and test sets to ensure proper evaluation of our model's performance.

Following this, we download the tokenizer from Google, specifically designed for the FLAN-T5 model, which helps in efficiently processing the text data. We then proceed to tokenize our training dataset, converting the raw text into a format suitable for the model to understand. This step is crucial for preparing the data for the subsequent training phase.

After tokenization, we download the FLAN-T5 model from the Google repository. This model is known for its advanced capabilities in generating high-quality text summaries. With the model and tokenized data ready, we initiate the training process. The model is trained for 3 epochs across all datasets, ensuring it learns effectively from the diverse range of texts provided.

Upon completing the training process, we push our trained model to the Hugging Face Hub. This step is essential for making the model accessible for future use. By hosting the pre-trained model on the Hugging Face Hub, we ensure that it can be easily downloaded and utilized for generating summaries without repeating the entire training process. This streamlined workflow not only optimizes our summarization tasks but also enhances the model's accessibility and usability for ongoing and
future projects.

## 3.3 AbstractivesummarizationwithLlama-3-8BandGemma-7B

First, we downloaded the necessary packages from the Unsloth AI GitHub repository to ensure we had all required dependencies and tools for our project. We then proceeded to load the Llama-38B model, developed by Meta AI, which is renowned for its extensive 8 billion trainable parameters, making it exceptionally powerful for various natural language processing tasks.

Following this, we sourced the necessary datasets from Hugging Face, a platform known for its extensive repository of high-quality datasets. These datasets were carefully chosen to cover a broad spectrum of text types and domains, ensuring the model could generalize well across different summarization tasks.

To enhance the model's efficiency and reduce memory usage, we integrated LoRA (Low-Rank Adaptation) and QLoRA (Quantized Low-Rank Adaptation) optimization techniques. These techniques are pivotal in minimizing computational resources and memory requirements without compromising the model's performance. LoRA focuses on optimizing the low-rank representation of the model parameters, while QLoRA introduces quantization to further reduce memory footprint, making
the training process more efficient.

With these optimizations in place, we trained the model for 3 epochs. This extended training period allowed the model to thoroughly learn and adapt to the nuances of the datasets, optimizing its loss function for better accuracy and performance. The training process involved rigorous fine-tuning, where the model's parameters were continuously adjusted to minimize the loss function and improve its summarization capabilities.

By leveraging the massive scale of the Llama-3-8B model, coupled with advanced optimization techniques like LoRA and QLoRA, we were able to significantly enhance the model's ability to generate high-quality, concise summaries. This comprehensive approach not only improved the model's efficiency and reduced its computational demands but also ensured that it remained robust and effective across diverse text summarization tasks.

# 4 Experimental Setup

## 4.1 Dataset

We employed five distinct datasets to fine-tune our models,partitioning each into 1000 training examples and 100 testing examples, ensuring the production of concise and precise summaries.

1. **CNN/DM**

   The CNN/DM Dataset is an English-language dataset containing just over 300k unique news articles as written by journalists at CNN and the Daily Mail. The current version supports both extractive and abstractive summarization, though the original version was created for machine reading and comprehension and abstractive question answering.

2. **Gigaword**

   It consists of over 6 million articles collected from various newswire sources, including agencies like the Associated Press (AP) and the New York Times. Each article in the Gigaword dataset
   is paired with a headline or a brief summary, making it suitable for supervised learning tasks.

3. **News Summary**

   The dataset consists of 4515 examples and contains Author_name, Headlines, Url of Article, Short text, Complete Article. I gathered the summarized news from Inshorts and only scraped the news articles from Hindu, Indian times and Guardian. Time period ranges from febrauary to august 2017.

4. **XSum**

   The dataset consists of 226,711 news articles accompanied with a one-sentence summary. The articlesarecollectedfromBBCarticles(2010to2017)andcoverawidevarietyofdomains(e.g., News, Politics, Sports, Weather, Business, Technology, Science, Health, Family, Education, Entertainment and Arts).

5. **BBC News**

This dataset for extractive text summarization has four hundred and seventeen political news articles of BBC from 2004 to 2005 in the News Articles folder. For each articles, five summaries are provided in the Summaries folder. The first clause of the text of articles is the respective title.

| **Datasets** | **Train(1000)** | | **Test(100)** | |
|---|---|---|---|---|
| | **Main Text** | **Summary** | **Main Text** | **Summary** |
| CNN/DM | 592 | 43 | 540 | 34 |
| Gigaword | 31 | 8 | 30 | 10 |
| XSum | 362 | 21 | 395 | 21 |
| BBC News | 387 | 170 | 330 | 143 |
| News Summary | 402 | 58 | 400 | 60 |

Table 1: Comparison of the average word counts for various datasets, after partitioning each into 1000 training examples and 100 testing examples

## 4.2 Data Preprocessing

To prepare the data for training the BART model, we begin by accessing the dataset from Hugging Face, a comprehensive repository of NLP datasets. Once loaded, the articles and their corresponding summaries are organized into separate arrays. This segregation allows for easier handling and manipulation of the data. Subsequently, these arrays are paired together to form tuples, each comprising an article and its associated summary. This step is crucial as it establishes the relationship between the input text and the target summary, facilitating supervised learning. Following this, the data is transformed into tensor slices, a format conducive to efficient processing by the model during training. Tensor slices enable batch processing, which accelerates training and enhances computational efficiency. Finally, with the preprocessed data ready, the training process commences, leveraging the power of the BART model to learn and generate concise and accurate summaries. This

meticulous data preprocessing ensures that the input data is appropriately formatted and optimized for effective
training of the BART model for text summarization tasks.

For the Llama3 model, our data preparation process initiates with the acquisition of datasets using the “load_dataset” library from Python’s Hugging Face ecosystem. Within this framework, we engineer a specialized function dubbed “formatting_prompt_func()” to meticulously structure our dataset. This function meticulously traverses through each item in the dataset, orchestrating a harmonious blend of essential components. In this orchestration, the instructions encapsulate the main article, while the output field encapsulates the corresponding summary. Subsequently, we amalgamate these components into a cohesive string, anchoring it with the key identifier “text” for streamlined processing. Notably, to ensure coherence and prevent infinite sentence generation, we append an EOS_TOKEN (End-of-Sequence token) as a terminal punctuation mark. This methodical approach not only ensures the seamless integration of data but also primes it for effective training of the Llama3 model, elevating its performance and efficacy in text summarization tasks.

This meticulous dataset preparation and preprocessing approach adhered to the standard practices followed in research papers, contributing to the credibility, reliability, and validity of our research outcomes.

## 4.3 Implementation Details

### 4.3.1 Language used: Python

For our work, we leveraged the latest version of Python, ensuring access to the most up-to-date features and improvements in the language. Additionally, we utilized a comprehensive suite of packages provided by Kaggle and Google Colab. These platforms offer a robust and versatile environment for data science and machine learning tasks, equipped with powerful tools and libraries essential for our project. By combining the latest Python advancements with the extensive resources of Kaggle and Google Colab, we ensured an efficient and effective workflow, facilitating seamless data processing, model training, and evaluation.

### 4.3.2 Libraries and tools used:

1. **CUDA** (Compute Unified Device Architecture) CUDA is a parallel computing platform and application programming interface (API) that allows software to use certain types of graphics processing units (GPUs) for general purpose processing.

   In our project, we utilized CUDA in conjunction with Google Colab, an online notebook environment. By leveraging CUDA in Colab, we were able to accelerate our computations and significantly improve performance. By offloading computationally intensive tasks to the GPU, we achieved faster execution times compared to using only the CPU. This was particularly beneficial in our abstractive text summarization task, as it involved processing large amounts of data and complex computations. CUDA allowed us to parallelize these computations and efficiently utilize the GPU resources, resulting in reduced execution times and improved overall productivity. The integration of CUDA with Google Colab provided us with a convenient and accessible platform for GPU-accelerated development, enabling us to leverage the power of GPUs without the need for dedicated hardware.

2. **Google Colab Notebook** Google Colab Notebook is a cloud-based platform that provides a free, cloud-based environment for running Python code. It is part of Google's suite of tools for machine learning and data science. One of the main advantages of Google Colab Notebook is that it provides access to powerful computing resources without the need for expensive hardware. This is because the code runs on Google's servers, which are optimized for large-scale data processing and machine learning tasks. Colab allows users to write and run Python code in a web browser. It provides a variety of features to support data analysis and machine learning, including support for popular libraries like NumPy, Pandas, and TensorFlow. It also includes tools for visualizing data and creating interactive plots and charts. One of the key benefits of using Google Colab Notebook is that it allows users to collaborate on projects in real time. Multiple users can work on the same notebook simultaneously, making it easy to share code and
   collaborate on data analysis and machine learning tasks.

3. **Kaggle** For our work, we utilized Kaggle, a platform renowned for its robust computational resources and user-friendly environment. Kaggle provides us with a powerful setup, including 30 GB of RAM and access to T4 GPUs, which significantly enhances performance for data-intensive tasks. Additionally, Kaggle Notebooks offer an integrated development environment with seamless access to a vast array of datasets, pre-installed machine learning libraries, and collaborative features. The platform also supports version control, ensuring that our work is systematically organized and easily reproducible. With Kaggle's extensive community support, rich documentation, and interactive coding environment, we were able to accelerate our workflow, from data preprocessing and model training to evaluation and iteration, thereby optimizing our project outcomes.

### 4.3.3 Implementation

In the implementation of our code, we have utilized several **key functions** that play crucial roles in different aspects of the task. These functions have been carefully designed and integrated to enable specific functionalities, such as data processing, model evaluation, and fine-tuning. So in this section we provide detailed descriptions of these important functions and discuss how they contribute to the overall functionality and performance of our code.

1. **preprocess_data function:** This function is designed to filter a dataset by removing articleabstract pairs that exhibit entity hallucination, ensuring that the filtered dataset only contains pairs where the abstract's entities are present in the corresponding article. This function takes a dataset as input. bstract pairs that satisfy the entity coherence criterion.

2. **process_data_to_model_inputs(batch) function:** The process_data_to_model_inputs function performs several important preprocessing steps on a batch of data. First, it removes leading and trailing spaces from the abstract and article strings to ensure consistency. Then, it initializes a list of empty sublists, one for each sample in the batch, to store the modified labels.

   The augmented abstracts are then truncated to the maximum output length to ensure they fit within the model's constraints. The modified labels are stored in the batch's labels field.

Furthermore, the function assigns the input IDs and attention masks obtained from the tokenizer to their respective fields in the batch. It creates a global attention mask, setting all elements to 0 initially, and changes the value of the first element for all samples to 1. This mask indicates which tokens are attended to during model training.

3. **calculate_bert_score , meteor_score function:** The compute_metrics function calculates various evaluation metrics for a given prediction. It takes the pred object as input, which contains label IDs (labels_ids) and predicted IDs (pred_ids).

   The function utilizes the rouge, meteor and bertscore libraries to compute several metrics. It uses the rouge.compute function to calculate Rouge-1, Rouge-2, and Rouge-L scores, passing the predicted strings (pred_str) and label strings (label_str) as references. Similarly, the meteor.compute function computes the Meteor score.
   The computed metrics are then returned as a dictionary, including the precision, recall, and FmeasureforRouge-1, Rouge-2, andRouge-Lmetrics, aswellastheMeteorscoreandBertScore. Each value is rounded to two decimal places.

To perform **fine-tuning** on the BART model, we utilized the Seq2SeqTrainer class from the Transformers library in conjunction with other functions described above. The process involves the following steps (hyperparameters mentioned in Table 4):

1. **Training Arguments:** The Seq2SeqTrainingArguments class is instantiated to define various training arguments and configurations. These arguments include parameters such as predict_with_generate to enable generation during prediction, evaluation_strategy to determine when to perform evaluation, per_device_train_batch_size and per_device_eval_batch_size to specify batch sizes, fp16 to enable mixed-precision training, output_dir to set the output directory for saving checkpoints, logging_steps to determine the frequency of logging training progress, eval_steps to specify the evaluation steps, save_steps to set the frequency of saving checkpoints, and other relevant parameters.

2. **SFTTrainer Initialization:**The SFTTrainer from Hugging Face is a specialized training utility designed to streamline the process of fine-tuning transformer models for specific tasks. It offers a user-friendly interface that simplifies the configuration and execution of the training process,

making it accessible even for those with limited machine learning experience. The SFTTrainer supports various functionalities such as gradient accumulation, mixed precision training, and distributed training across multiple GPUs, ensuring efficient utilization of computational resources. By leveraging the SFTTrainer, users can fine-tune pre-trained models like FLAN-T5 with ease, optimizing them for tasks such as text summarization, while benefiting from Hugging Face's robust ecosystem of tools and libraries.

3. **Training Execution:** The trainer.train() method is called to start the fine-tuning process. This executes the training loop over the specified number of epochs (num_train_epochs) using the provided training and evaluation datasets. The trainer automatically handles tasks such as batching, gradient accumulation, forward and backward passes, parameter updates, and checkpoint saving according to the specified training arguments.

Table 2: Hyperparameters used for Fine-tuning the BART model

| Hyperparameter | Value |
|---|---|
| MAX_ENCODER_SEQUENCE_LENGTH | 1000 |
| MAX_DECODER_SEQUENCE_LENGTH | 1000 |
| MAX_GENERATION_LENGTH | 40 |
| Min. output length | 100 |
| batch_size | 1 |
| No. of Batches | 600 |
| epochs | 3 |
| learning_rate | 5e-5 |
| weight_decay | 0.01 |
| epsilon | 1e-6 |
| global_clipnorm | 1.0 |

Table 3: Hyperparameters used for Fine-tuning the FLAN-T5 model

| Hyperparameter | Value |
|---|---|
| per_device_train_batch_size | 8 |
| per_device_eval_batch_size | 8 |

| | |
|---|---|
| predict_with_generate | True |
| fp16 | False |
| learning_rate | 5e-5 |
| num_train_epochs | 3 |
| logging_strategy | “steps” |
| logging_steps | 5 |
| evaluation_strategy | “epoch” |
| save_strategy | “epoch” |
| save_total_limit | 2 |
| load_best_model_at_end | True |

Table 4: Hyperparameters used for Fine-tuning the Llama3 and Gemma model

| Hyperparameter | Value |
|---|---|
| gradient_accumulation_steps | 4 |
| warmup_steps | 5 |
| max_steps | 375 |
| learning_rate | 2e-3 |
| fp16 | not torch.cuda.is_bf16_supported() |
| bf16 | torch.cuda.is_bf16_supported() |
| logging_steps | 10 |
| optim | “adamw_8bit” |
| weight_decay | 0.01 |
| seed | 3407 |
| lr_scheduler_type | “linear” |

# 5 Evaluation Details

## 5.1 ROUGE Evaluation Metric

ROUGE (Recall-Oriented Understudy for Gisting Evaluation) is a set of metrics used to evaluate the quality of summaries generated by automatic summarization systems. It compares the machinegenerated summaries against one or more reference (human-generated) summaries. ROUGE is widely used in natural language processing (NLP) for tasks such as summarization, translation, and paraphras-
ing. Here are the main ROUGE metrics:

### 5.1.1 ROUGE-N

Measures the overlap of n-grams (sequences of n words) between the generated summary and the reference summaries.

- **ROUGE-1**: ROUGE-1 focuses on the overlap of unigrams (single words) between the generated text and the reference text.

  ROUGE-1 calculates three primary components:
  1. **Precision** : The fraction of unigrams in the generated text that appear in the reference text.

  $$Precision = \frac{Number\ of\ Overlapping\ unigrams}{Total\ number\ of\ unigrams\ in\ the\ generated\ text}$$

  2. **Recall** : The fraction of unigrams in the reference text that appear in the generated text.

  $$Recall = \frac{Number\ of\ Overlapping\ unigrams}{Total\ number\ of\ unigrams\ in\ the\ reference\ text}$$

  3. **F1-Score** : The harmonic mean of precision and recall, providing a single measure of the test's accuracy.

**F1-Score** is calculated based on the Recall and Precision scores of the candidate summary. As a result it takes into consideration the relevant words captured without capturing irrelevant words.

For this reason F1-Score seems like the best way to score a candidate summary.

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall}$$

• **ROUGE-2**: ROUGE-2 metric is based on matching Bigrams i.e. sequence of two text matches is counted between candidate and reference summaries in order to evaluate the performance of the summary.

ROUGE-2 calculates three primary components:

1. **Precision** : The fraction of bigrams in the generated text that appear in the reference text.

$$Precision = \frac{Number\ of\ Overlapping\ bigrams}{Total\ number\ of\ bigrams\ in\ the\ generated\ text}$$

2. **Recall** : The fraction of bigrams in the reference text that appear in the generated text.

$$Recall = \frac{Number\ of\ Overlapping\ bigrams}{Total\ number\ of\ bigrams\ in\ the\ reference\ text}$$

3. **F1-Score** : The harmonic mean of precision and recall, providing a single measure of the test's accuracy.

**F1-Score** is calculated based on the Recall and Precision scores of the candidate summary. As a result it takes into consideration the relevant words captured without capturing irrelevant words.

For this reason F1-Score seems like the best way to score a candidate summary.

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall}$$

ROUGE-N calculates the recall, precision, and F1-measure for each model.

### 5.1.2 ROUGE-L

ROUGE-L scores the candidate summary based on the length of longest common sub-sequence (LCS) found between the candidate and the gold summary. Longer the overlapping sequence length better is the ROUGE-L score.

ROUGE calculates the summary score based on 3 different formula based on the n-gram matching - Recall, Precision, and F1 score. For both ROUGE-N and ROUGE-L, recall, precision and F1-score can be calculated. The only difference being ROUGE-N calculates on the basis of matching n-grams and ROUGE-L calculates based on matching LCS length.

**1. Precision** : The fraction of the length of the LCS to the total number of words in the generated text.

$$Precision = \frac{Length\ of\ LCS}{Total\ number\ of\ words\ generated\ in\ the\ generated\ text}$$

**2. Recall** : The fraction of the length of the LCS to the total number of words in the reference text.

$$Recall = \frac{Length\ of\ LCS}{Total\ number\ of\ words\ in\ the\ reference\ text}$$

**3. F1-Score** : The harmonic mean of precision and recall, providing a single measure of the test's accuracy.

**F1-Score** is calculated based on the Recall and Precision scores of the candidate summary. As a result it takes into consideration the relevant words captured without capturing irrelevant words. For this reason F1-Score seems like the best way to score a candidate summary.

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall}$$

## 5.2 BERTScore

BERTScore [60] is a metric commonly used in natural language processing and machine learning to evaluate the quality of generated text. It stands for **B**idirectional **E**ncoder **R**epresentations from **T**ransformers score. BERT, a powerful language model developed by Google, forms the foundation of this metric. It leverages the Bidirectional Encoder Representations from Transformers (BERT) model to compute a similarity score between the generated text and reference text. BERTScore measures the similarity between a reference sentence and a generated sentence by considering both their semantic meaning and syntactic structure. Unlike traditional evaluation methods that rely solely on lexical overlap, BERTScore takes into account the contextual information captured by BERT.

## 5.3 METEOR

METEOR is a metric for machine translation evaluation, and it claims to have better correlation with human judgement. The METEOR score is a widely used evaluation metric in the field of natural language processing (NLP) and machine translation. It stands for **M**etric for **E**valuation of **T**ranslation with **E**xplicit **O**rdering and it was introduced by Banerjee and Lavie in 2005 [61]. METEOR is designed to measure the quality of a generated translation by comparing it to one or more reference translations. The match is scored by METEOR based on a mix of unigram precision, unigram recall, and a fragmentation measure that is meant to accurately reflect how well-ordered the matched words are in both the model-generated summary and the author-written abstract. It determines the unigrams' precision and recall, or R and P, respectively. The formula is then used to get the F-Mean score:

$$F_{mean} = \frac{10 * Recall * Precision}{Recall + 9 * Precision}$$

$$Penalty = 0.5 * \left( \frac{\#chunks}{\#matched\_unigrams} \right)^{3}$$

Final Score is calculated as :

$$Score = F_{mean} * (1 - Penalty)$$

# 6 Results, Analysis and Discussion

## 6.1 Training and Validation Loss table

| Model | Epoch | Training Loss | Validation Loss |
|---|---|---|---|
| BART | 1 | 0.5867 | 0.1066 |
| | 2 | 0.3665 | 0.0300 |
| | 3 | 0.2743 | 0.0242 |
| FLAN-T5 | 1 | 1.7911 | 1.3899 |
| | 2 | 1.7296 | 1.3305 |
| | 3 | 1.4633 | 1.3100 |
| Llama-3-8B | 1 | 1.8122 | 1.3795 |
| | 2 | 1.7723 | 1.3483 |
| | 3 | 1.7003 | 1.3787 |
| Gemma-7B | 1 | 1.8143 | 1.3710 |
| | 2 | 1.7705 | 1.3527 |
| | 3 | 1.7825 | 1.3507 |

Table 5: Training and validation loss of used models

## 6.2 Learning Curves

Learning curves are a measure of model's learning progress over training or validation steps. Learning curves is a popular and accurate measure to check the performance of the neural network created to learn training properties. The learning curve is influenced by various factors like network complexity, learning rate, amount of training data, number of training steps etc. This helps in identifying if these network factors needs to be updated or not.

1. The training loss(shown in Table 5) metric is used to evaluate how a neural network model fits the training data. In out project we have calculated the training loss for every 3 epochs by taking a sum of the errors.

2. The validation loss(shown in Table 5) metric is used to evaluate how a neural network model fits the validation data. In our project we have calculated the validation loss for every 3 epochs as well by taking a sum of the errors.

We have compared the Training loss curve with the Validation loss curve to determine the model training performance for the approach proposed in the project along with the ROUGE scores used to evaluate the generated test summaries.

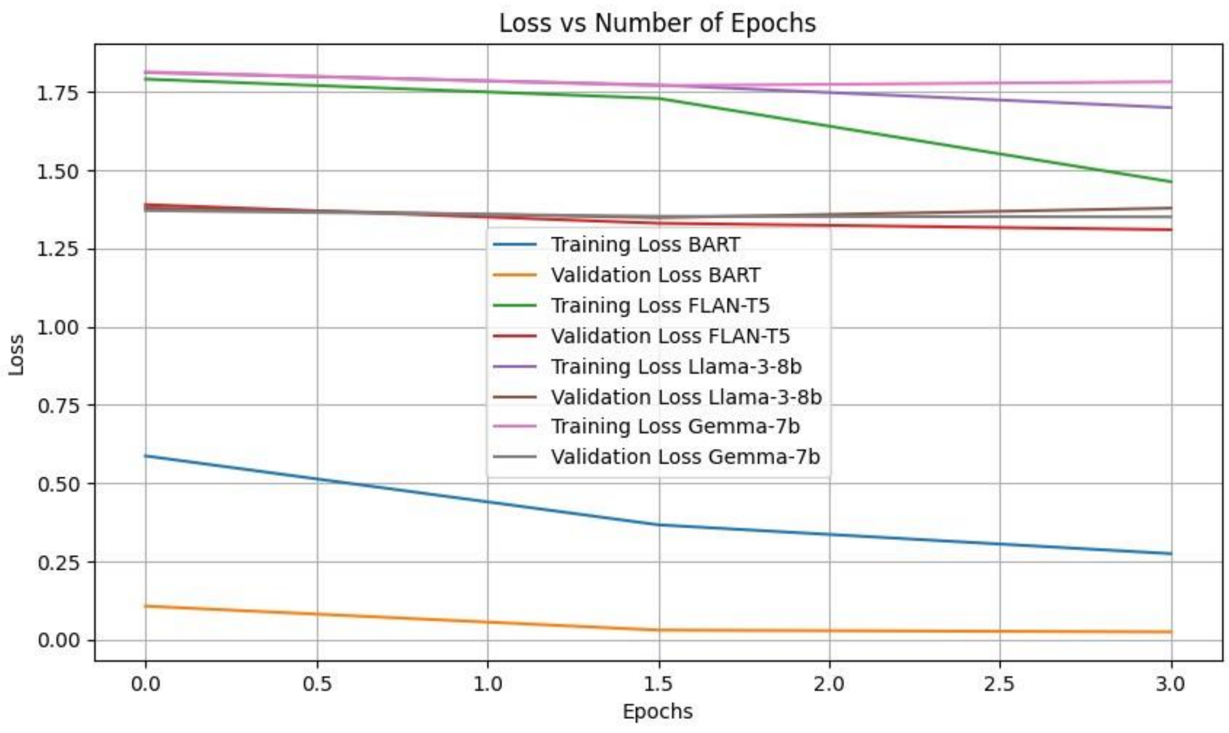

Figure 1: Training and Validation Loss v/s No. of Epochs

**Observations from Figure 1:**

- **Training Loss**: The training loss values decrease initially but then fluctuate around a certain range. This indicates that the model is learning initially but may struggle to further improve its performance. The fluctuations in training loss suggest that the model may not be converging well or could be experiencing overfitting.

- **Validation Loss**: The validation loss values consistently decrease over the course of training, indicatingthatthemodelisgeneralizingwelltounseendata. Thisisapositivesign, asitsuggests that the model is not overfitting to the training data. Lower validation loss values typically indicate better performance on unseen data.

- **Comparison**: Comparing the training and validation loss values, we can observe that the validation loss is consistently lower than the training loss. This indicates that the model is generalizing well and not overfitting to the training data. However, the fact that the training loss is not de-
creasing consistently suggests that the model might not be learning as effectively as desired.

## 6.3 Model Comparison

In this section, we present a comparison of the models across various datasets used in this study. We evaluate the performance using multiple metrics, including the F1 scores for ROUGE-1, ROUGE2, and ROUGE-L, as well as BERTScore and METEOR. This multi-metric approach allows for a
comprehensive assessment of each model’s ability to generate high-quality summaries, highlighting their strengths and weaknesses in different contexts.

| Models | **ROUGE-1** | **ROUGE-2** | **ROUGE-L** | **BERTScore** | **METEOR** |
|---|---|---|---|---|---|
| BART | 0.25 | 0.07 | 0.18 | 0.53 | 0.23 |
| FLAN-T5 | **0.34** | **0.14** | **0.24** | **0.58** | **0.34** |
| Llama-3-8B | 0.32 | 0.12 | 0.23 | 0.56 | 0.25 |
| Gemma-7B | 0.33 | 0.14 | 0.24 | 0.58 | 0.29 |

Table 6: Evolution of the models: F1 scores for ROUGE, METEOR and BERTScore on **CNN/DM** dataset.

| Models | **ROUGE-1** | **ROUGE-2** | **ROUGE-L** | **BERTScore** | **METEOR** |
|---|---|---|---|---|---|

| | | | | | |
|---|---|---|---|---|---|
| BART | 0.29 | 0.10 | 0.26 | 0.53 | 0.18 |
| FLAN-T5 | 0.33 | 0.12 | 0.28 | **0.57** | **0.35** |
| Llama-3-8B | 0.29 | 0.09 | 0.25 | 0.50 | 0.25 |
| Gemma-7B | **0.35** | **0.14** | **0.32** | 0.55 | 0.29 |

Table 7: Evolution of the models: F1 scores for ROUGE, METEOR and BERTScore on **Gigaword** dataset.

| Models | **ROUGE-1** | **ROUGE-2** | **ROUGE-L** | **BERTScore** | **METEOR** |
|---|---|---|---|---|---|
| BART | **0.56** | **0.36** | **0.45** | **0.73** | **0.49** |
| FLAN-T5 | 0.52 | 0.29 | 0.39 | 0.70 | 0.44 |
| Llama-3-8B | 0.51 | 0.27 | 0.37 | 0.67 | 0.41 |
| Gemma-7B | 0.51 | 0.28 | 0.38 | 0.68 | 0.38 |

Table 8: Evolution of the models: F1 scores for ROUGE, METEOR and BERTScore on **News Summary** dataset.

| Models | **ROUGE-1** | **ROUGE-2** | **ROUGE-L** | **BERTScore** | **METEOR** |
|---|---|---|---|---|---|
| BART | 0.27 | 0.07 | 0.21 | 0.57 | 0.22 |
| FLAN-T5 | 0.35 | 0.13 | 0.27 | 0.61 | 0.30 |
| Llama-3-8B | 0.37 | 0.15 | 0.29 | 0.56 | 0.27 |
| Gemma-7B | **0.39** | **0.18** | **0.32** | **0.61** | **0.30** |

Table 9: Evolution of the models: F1 scores for ROUGE, METEOR and BERTScore on **XSum** dataset.

1. **Dataset: CNN/DM**

   Results are shown in Table 6 for ROUGE-1,ROUGE-2,ROUGE-L,BERTScore and METEOR whenthedatasetisCNN/DM.WeobservethatamongthefourmodelsFLAN-T5modelachieves the highest ROUGE, BERTScore and METEOR values.

2. **Dataset: Gigaword**

Results are shown in Table 7 for ROUGE-1,ROUGE-2,ROUGE-L,BERTScore and METEOR when the dataset is Gigaword. We observe that among the four models Gemma-7B model

| Models | **ROUGE-1** | **ROUGE-2** | **ROUGE-L** | **BERTScore** | **METEOR** |
|---|---|---|---|---|---|
| BART | **0.61** | 0.47 | **0.42** | **0.75** | **0.51** |
| FLAN-T5 | 0.27 | 0.23 | 0.25 | 0.64 | 0.15 |
| Llama-3-8B | 0.61 | **0.49** | 0.40 | 0.74 | 0.51 |
| Gemma-7B | 0.51 | 0.39 | 0.36 | 0.71 | 0.33 |

Table 10: Evolution of the models: F1 scores for ROUGE, METEOR and BERTScore on **BBC News** dataset.

achieves the highest ROUGE values but FLAN-T5 model achives the highest BERTScore and METEOR score values.

3. **Dataset: News Summary**

   Results are shown in Table 8 for ROUGE-1,ROUGE-2,ROUGE-L,BERTScore and METEOR when the dataset is News Summary. We observe that among the four models BART model achieves the highest ROUGE, BERTScore and METEOR values.

4. **Dataset: XSum**

   Results are shown in Table 9 for ROUGE-1,ROUGE-2,ROUGE-L,BERTScore and METEOR when the dataset is XSum. We observe that among the four models Gemma-7B model achieves the highest ROUGE, BERTScore and METEOR values.

5. **Dataset: BBC News**

   Results are shown in Table 10 for ROUGE-1,ROUGE-2,ROUGE-L,BERTScore and METEOR when the dataset is BBC News. We observe that among the four models BART model achieves highestROUGE-1,ROUGE-L,BERTScoreandMETEORvaluesbutLlama-3-8Bmodelsachieves highest ROUGE-2 values.

## 6.4 Observation on Computational Resource

| **Factors** | **BART** | **FLAN-T5** | **Llama-3-8B** | **Gemma-7B** |
|---|---|---|---|---|

| Trainable Params | 139M | 248M | 8B | 7B |
|---|---|---|---|---|
| Available RAM:12.7GB | 4.8 | 3.5 | 4 | 4.5 |
| Available GPU:15GB | 8 | 9 | 9.8 | 9.2 |
| Available Disk:78GB | 7.7 | 10.4 | 33.6 | 33.5 |
| GPU utilization | 100% | 97% | 95% | 100% |
| CPU utilization | 100% | 100% | 96% | 98% |
| Time for one epoch(min) | 6.45 | 1.5 | 50 | 67 |

Table 11: Power consumption, compute expenditure statistics for summarization models.

## 6.5 Case Studies

We now present a few examples demonstrating the outputs produced by our models used in this paper. In all the case studies reported below, blue color represents errors or wrong information and yellow color shows repeating words .

### 6.5.1 Case Study 1

In this section, we present an illustrative example, as shown in Table 12, to demonstrate the impact of training the various models on gigaword dataset of inadequately small size, specifically 1000 samples. This limited dataset size significantly affects the quality of the generated summaries. In the example provided, the incorrect portions of the summary are highlighted in blue to clearly indicate the errors. Additionally, the model-generated summary contains extra information that is irrelevant and does not pertain to the context of the original article. This irrelevant information highlights the model's tendency to introduce inaccuracies when trained on insufficient data, emphasizing the importance of using adequately large and diverse datasets to improve the reliability and accuracy of AI-generated summaries.

Table 12: Comparison of generated and ground truth summary examples from the test set of Gigaword dataset .The wrong word here marked in blue color. The example is taken from https://www.cfr.org/backgrounder/sri-lankan-conflict

| |
|---|
| **Ground-truth summary (abstract):** b'sri lanka closes schools as war escalates' |
| **BART generated:** sri lanka government shuts government schools amid military campaign against rebels |
| **FLAN-T5 generated:** sri lanka announces closure of government schools as tamil separatists campaign intensifies in north of london |
| **Llama-3-8B generated:** srilanka shuts schools as war intensifies |
| **Gemma-7B generated:** sri lanka closes schools amid escalating war |

### 6.5.2 Case Study 2

In this section, we present an example highlighting word repetition issues encountered by various models when fine-tuned on the XSum dataset with only 1000 training examples. This limited training data often leads to suboptimal model performance, as evidenced by the repeated words in the generated summary. The repeated words, marked in yellow, illustrate the model's difficulty in generating diverse and coherent content under these conditions. This example underscores the importance of sufficient training data for achieving high-quality summarization results and highlights the challenges of working with smaller datasets. Through this example, we aim to shed light on the potential pitfalls of training language models with limited data and emphasize the need for adequate training resources to ensure the generation of accurate and varied summaries.

Table 13: Comparison of generated and ground truth summary examples from the test set of XSum dataset.Repeated words are marked in yellow.The example is taken from https://www.atptour.com/en/news/djokovic-1100-wins-2024

| |
|---|
| **Ground-truth summary (abstract):** Of his last 30 matches in 2016, Andy Murray won 28 and lost just two. |
| **BART:** Andy Murray missed a week of practice in preparation for this week's French Open at the All England Club in Rome. |
| **FLAN-T5:** Andy Murray's record at Wimbledon has been shattered by a knee injury, but he is still the world number one in the world . |
| **Llama-3-8B:** The world number one is playing in the first round of the Fever-Tree Championships at Queen's Club. |
| **Gemma-7B:** Andy Murray's Wimbledon title defence is under threat after he limped out of practice on Friday. |

Table 14: Comparison of generated and ground truth summary examples of the BART model on test set of CNN/DM dataset. Repeated words are marked in yellow. The example is taken from https://edition.cnn.com/2015/04/03/us/kentucky-ge-fire/index.html

| |
|---|
| **Ground-truth summary (abstract):** b'Fire breaks out at the General Electric Appliance Park in Louisville, Kentucky .City official: No is believed to be injured or trapped . |
| **BART:** Fire damages industrial park in Kentucky, official says .Video shows blaze spreading thick smoke over area .No injuries or trapped, official says .Officials didn't know what had caused the fire, which had gone to at least four alarms . Officials didn't know what had caused the fire. |
| **FLAN-T5:** Fire breaks out in Kentucky industrial park, sending plumes of thick smoke over the area . No reports of anyone injured or trapped in the blaze, officials say . GE's facility in Louisville Appliance Park is "revitalizing manufacturing in the United States" |

| |
|---|
| **Llama-3-8B:** NEW: Fire began shortly before 7 a.m. at General Electric Appliance Park. NEW: No reports of anyone injured or trapped, official says. NEW: Firefighters spray water from the periphery of affected buildings. NEW: Park is large, such that 34 football fields could fit in one of its warehouses. |
| **Gemma-7B:** Massive fire breaks out at General Electric Appliance Park in Louisville, Kentucky .Firefighters take up positions around affected buildings, spraying water .Authorities don't know what caused the fire, which has gone to at least four alarms . |

## 6.6 Discussion

**Why did we choose these models?**

We chose these models due to several compelling reasons. Firstly, their compatibility with easy training on platforms like Google Colab allows for efficient experimentation and development. The models, including BART, FLAN-T5, Llama-3-8B, and Gemma-7B, are designed to be highly adaptable and effective across various tasks, particularly in text summarization. Their lower RAM and GPU usage compared to other state-of-the-art models make them more accessible and practical for a wider range of users, including those with limited computational resources. Additionally, these models have demonstrated strong performance in generating coherent and concise summaries, thanks to their advanced architectures and pretraining on large, diverse datasets. This combination of ease of use, computational efficiency, and high-quality output makes them ideal choices for our summarization tasks.

**Why does our model perform much worse compared to the state of the art text summarization models?**

Another important factor is the computational resources available for training the model. We trained our model using Google Colab , which has limitations on compute units and runtime. As a result, we were constrained in terms of the number of experiments we could run and the complexity of the models we could train. State-of-the-art models, on the other hand, are typically trained on high-performance computing clusters or specialized hardware, enabling them to leverage more computational power for training.

# 7 Conclusion and Future Work

In this paper, we conducted a comprehensive analysis comparing various pre-trained models and large language models (LLMs) in the realm of abstractive text generation. Our detailed evaluation sheds light on the strengths and weaknesses of each model, particularly focusing on their ability to generate coherent, informative, and concise summaries. This comparative study is essential for understanding how different models perform under various conditions and datasets, providing valuable
insights into their practical applications in text summarization tasks.

To further validate our findings and enhance the reliability of the generated summaries, we plan to incorporate human evaluations in our future work. This next step will involve human judges assessing the quality of the summaries produced by the models. By doing so, we aim to obtain a more nuanced understanding of the models' performance from a human perspective. This assessment will help identify the areas where the AI-generated summaries align with or diverge from human expectations of
coherence, informativeness, and readability.

Additionally, we aim to conduct a thorough comparison between human-generated summaries and those produced by our models. This comparative analysis will offer deep insights into how well the models replicate the quality and depth of human-generated content. By juxtaposing the two, we can identify specific strengths and limitations of the AI models, guiding future improvements and refinements.

Ultimately, our goal is to bridge the gap between AI and human summarization capabilities, ensuring that the models not only perform well on quantitative metrics but also meet qualitative standards expected by human users. This future work will be crucial in advancing the field of text summarization, making AI-generated content more reliable, trustworthy, and applicable in real-world scenarios where the quality of information is paramount.

# Appendix

**Availabilityofthesourcecode:** https://github.com/Soumabhaghosh/Final_year_project

**Availabilityofthedatasets:** https://huggingface.co/datasets/abisee/cnn_dailymail

, https://huggingface.co/datasets/Harvard/gigaword, https://huggingface.co/datasets/

EdinburghNLP/xsum, https://huggingface.co/datasets/gopalkalpande/bbc-news-summary,
https://www.kaggle.com/datasets/sunnysai12345/news-summary

# References

[1] Romain Paulus, Caiming Xiong, and Richard Socher. A deep reinforced model for abstractive summarization. *arXiv preprint arXiv:1705.04304*, 2017.

[2] Yang Liu, Ivan Titov, and Maria Lapata. Single document summarization as tree induction. In *2019 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 1745–1755. Association for Computational Linguistics (ACL), 2019.

[3] Sanchit Agarwal, Nikhil Kumar Singh, and Priyanka Meel. Single-document summarization using sentence embeddings and k-means clustering. In *2018 International Conference on Advances in Computing, Communication Control and Networking (ICACCCN)*, pages 162–165. IEEE, 2018.

[4] Naveen Saini, Sriparna Saha, Dhiraj Chakraborty, and Pushpak Bhattacharyya. Extractive single document summarization using binary differential evolution: Optimization of different sentence quality measures. *PloS one*, 14(11):e0223477, 2019.

[5] Jian-Ping Mei and Lihui Chen. Sumcr: A new subtopic-based extractive approach for text summarization. *Knowledge and information systems*, 31:527–545, 2012.

[6] Rasim M Alguliev, Ramiz M Aliguliyev, and Nijat R Isazade. Multiple documents summarization based on evolutionary optimization algorithm. *Expert Systems with Applications*, 40(5):1675–1689, 2013.

[7] Giang Tran, Eelco Herder, and Katja Markert. Joint graphical models for date selection in timeline summarization. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 1598–1607. Association for Computational Linguistics, 2015.

[8] Melissa Ailem, Bowen Zhang, and Fei Sha. Topic augmented generator for abstractive summarization. *arXiv preprint arXiv:1908.07026*, 2019.

[9] Liu Na, Tang Di, Lu Ying, Tang Xiao-Jun, and Wang Hai-Wen. Topic-sensitive multi-document summarization algorithm. *Computer Science and Information Systems*, 12(4):1375–1389, 2015.

[10] Lu Wang and Claire Cardie. Unsupervised topic modeling approaches to decision summarization

in spoken meetings. *arXiv preprint arXiv:1606.07829*, 2016.

[11] Zongda Wu, Li Lei, Guiling Li, Hui Huang, Chengren Zheng, Enhong Chen, and Guandong Xu. A topic modeling based approach to novel document automatic summarization. *Expert Systems with Applications*, 84:12–23, 2017.

[12] Nikolaos Gialitsis, Nikiforos Pittaras, and Panagiotis Stamatopoulos. A topic-based sentence representation for extractive text summarization. In *Proceedings of the Workshop MultiLing 2019: Summarization Across Languages, Genres and Sources*, pages 26–34, 2019.

[13] H. P. Luhn. The automatic creation of literature abstracts. *IBM J. Res. Dev.*, 2(2):159–165, apr 1958.

[14] Chandra Shekhar Yadav and Aditi Sharan. Hybrid approach for single text document summarization using statistical and sentiment features. *International Journal of Information Retrieval Research (IJIRR)*, 5(4):46–70, 2015.

[15] H. P. Edmundson. New methods in automatic extracting. *J. ACM*, 16(2):264–285, apr 1969.

[16] Mahak Gambhir and Vishal Gupta. Recent automatic text summarization techniques: A survey. *Artif. Intell. Rev.*, 47(1):1–66, jan 2017.

[17] G. Erkan and D. R. Radev. LexRank: Graph-based lexical centrality as salience in text summarization. *Journal of Artificial Intelligence Research*, 22:457–479, dec 2004.

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

[19] Elena Filatova and Vasileios Hatzivassiloglou. Event-based extractive summarization. In *Text Summarization Branches Out*, pages 104–111, Barcelona, Spain, July 2004. Association for Computational Linguistics.

[20] Dragomir R. Radev, Hongyan Jing, and Malgorzata Budzikowska. Centroid-based summarization of multiple documents: sentence extraction, utility-based evaluation, and user studies. In
*NAACL-ANLP 2000 Workshop: Automatic Summarization*, 2000.

[21] Julian Kupiec, Jan Pedersen, and Francine Chen. A trainable document summarizer. In *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '95, page 68–73, New York, NY, USA, 1995. Association for Computing Machinery.

[22] Horacio Saggion and T. Poibeau. Automatic text summarization: Past, present and future. In *Multi-source, Multilingual Information Extraction and Summarization*, 2013.

[23] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. *CoRR*, abs/1409.3215, 2014.

[24] Sumit Chopra, Michael Auli, and Alexander M. Rush. Abstractive sentence summarization with attentive recurrent neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*,
pages 93–98, San Diego, California, June 2016. Association for Computational Linguistics.

[25] Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Çağlar Gulçehre, and Bing Xiang. Abstractive text summarization using sequence-to-sequence RNNs and beyond. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning*, pages 280–290, Berlin, Germany, August 2016. Association for Computational Linguistics.

[26] Alexander M. Rush, Sumit Chopra, and Jason Weston. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 379–389, Lisbon, Portugal, September 2015. Association for Computational Linguistics.

[27] Romain Paulus, Caiming Xiong, and Richard Socher. A deep reinforced model for abstractive summarization. *CoRR*, abs/1705.04304, 2017.

[28] Abigail See, Peter J. Liu, and Christopher D. Manning. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083, Vancouver, Canada, July 2017. Association for Computational Linguistics.

[29] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017.

[30] Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter J. Liu. Pegasus: Pre-training with extracted gap-sentences for abstractive summarization, 2020.

[31] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. BART: Denoising sequence-to-sequence pretraining for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online, July 2020. Association for Computational Linguistics.

[32] Dragomir Radev, Eduard Hovy, and Kathleen McKeown. Introduction to the special issue on summarization. *Computational linguistics*, 28(4):399–408, 2002.

[33] Rasim M Alguliev, Ramiz M Aliguliyev, and Chingiz A Mehdiyev. Sentence selection for generic document summarization using an adaptive differential evolution algorithm. *Swarm and Evolutionary Computation*, 1(4):213–222, 2011.

[34] Inderjeet Mani and Mark T Maybury. Advances in automatic text summarization (vol. 293). *Camb MA*, 1999.

[35] Eduard Hovy and Chin-Yew Lin. Automated text summarization and the summarist system. In *TIPSTER TEXT PROGRAM PHASE III: Proceedings of a Workshop held at Baltimore, Maryland, October 13-15, 1998*, pages 197–214, 1998.

[36] Hans Peter Luhn. The automatic creation of literature abstracts. *IBM Journal of research and development*, 2(2):159–165, 1958.

[37] Phyllis B Baxendale. Machine-made index for technical literature—an experiment. *IBM Journal of research and development*, 2(4):354–361, 1958.

[38] Harold P Edmundson. New methods in automatic extracting. *Journal of the ACM (JACM)*, 16(2):264–285, 1969.

[39] You Ouyang, Wenjie Li, Sujian Li, and Qin Lu. Applying regression models to query-focused multi-document summarization. *Information Processing & Management*, 47(2):227–237, 2011.

[40] Jaime Carbonell and Jade Goldstein. The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 335–336, 1998.

[41] Ming Che Lee. A novel sentence similarity measure for semantic-based expert systems. *Expert Systems with Applications*, 38(5):6392–6399, 2011.

[42] Danushka Bollegala, Naoaki Okazaki, and Mitsuru Ishizuka. A bottom-up approach to sentence ordering for multi-document summarization. *Information processing & management*, 46(1):89–109, 2010.

[43] Kamal Sarkar. Syntactic trimming of extracted sentences for improving extractive multi-document summarization. *Journal of Computing*, 2(7):177–184, 2010.

[44] Xiao Pu, Mingqi Gao, and Xiaojun Wan. Summarization is (almost) dead. *arXiv preprint arXiv:2309.09558*, 2023.

[45] Tanya Goyal, Junyi Jessy Li, and Greg Durrett. News summarization and evaluation in the era of gpt-3. *arXiv preprint arXiv:2209.12356*, 2022.

[46] Haopeng Zhang, Xiao Liu, and Jiawei Zhang. Summit: Iterative text summarization via chatgpt. *arXiv preprint arXiv:2305.14835*, 2023.

[47] Yixin Liu, Kejian Shi, Katherine S He, Longtian Ye, Alexander R Fabbri, Pengfei Liu, Dragomir Radev, and Arman Cohan. On learning to summarize with large language models as references. *arXiv preprint arXiv:2305.14239*, 2023.

[48] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. Bart: Denoising sequence-to-sequence

pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*, 2019.

[49] Yixin Liu, Pengfei Liu, Dragomir Radev, and Graham Neubig. Brio: Bringing order to abstractive summarization. *arXiv preprint arXiv:2203.16804*, 2022.

[50] Yiming Wang, Zhuosheng Zhang, and Rui Wang. Element-aware summarization with large language models: Expert-aligned evaluation and chain-of-thought method. *arXiv preprint arXiv:2305.13412*, 2023.

[51] Griffin Adams, Alexander Fabbri, Faisal Ladhak, Eric Lehman, and Noémie Elhadad. From sparse to dense: Gpt-4 summarization with chain of density prompting. *arXiv preprint arXiv:2309.04269*, 2023.

[52] Mathieu Ravaut, Shafiq Joty, and Nancy F Chen. Towards summary candidates fusion. *arXiv preprint arXiv:2210.08779*, 2022.

[53] Simeng Sun, Kalpesh Krishna, Andrew Mattarella-Micke, and Mohit Iyyer. Do long-range language models actually use long-range context? *arXiv preprint arXiv:2109.09115*, 2021.

[54] Ruoqi Shen, Sébastien Bubeck, Ronen Eldan, Yin Tat Lee, Yuanzhi Li, and Yi Zhang. Positional description matters for transformers arithmetic. *arXiv preprint arXiv:2311.14737*, 2023.

[55] Chujie Zheng, Hao Zhou, Fandong Meng, Jie Zhou, and Minlie Huang. Large language models are not robust multiple choice selectors. In *The Twelfth International Conference on Learning Representations*, 2023.

[56] Norah Alzahrani, Hisham Abdullah Alyahya, Yazeed Alnumay, Sultan Alrashed, Shaykhah Alsubaie, Yusef Almushaykeh, Faisal Mirza, Nouf Alotaibi, Nora Altwairesh, Areeb Alowisheq, et al. When benchmarks are targets: Revealing the sensitivity of large language model leaderboards. *arXiv preprint arXiv:2402.01781*, 2024.

[57] Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. Mixtral of experts. *arXiv preprint arXiv:2401.04088*, 2024.

[58] Raphael Tang, Xinyu Zhang, Xueguang Ma, Jimmy Lin, and Ferhan Ture. Found in the middle: Permutation self-consistency improves listwise ranking in large language models. *arXiv preprint arXiv:2310.07712*, 2023.

[59] Anshuman Chhabra, Hadi Askari, and Prasant Mohapatra. Revisiting zero-shot abstractive summarization in the era of large language models from the perspective of position bias. *arXiv preprint arXiv:2401.01989*, 2024.

[60] Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. Bertscore: Evaluating text generation with bert, 2020.

[61] Satanjeev Banerjee and Alon Lavie. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72, Ann Arbor, Michigan, June 2005. Association for Computational Linguistics.