Optimizing Abstractive Summarization With Fine-Tuned PEGASUS

by

Sadiul Arefin Rafi
20101120
Naimur Rahman
20101284
Kazi Nazibul Islam
20101372
Ha-mim Ahmad
20101286

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science and Engineering

Department of Computer Science and Engineering
School of Data and Sciences
Brac University
September 2023

Declaration

It is hereby declared that

1. The thesis submitted is my/our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

Student's Full Name & Signature:

Sadiul	[illegible]
Sadiul Arefin Rafi 20101120	Naimur Rahman 20101284
Nazib	Ha-mim
Kazi Nazibul Islam 20101372	Ha-mim Ahmad 20101286

Approval

The thesis/project titled "Optimizing Abstractive Summarization With Fine-Tuned PEGASUS" submitted by

1. Sadiul Arefin Rafi (20101120)
2. Naimur Rahman (20101284)
3. Kazi Nazibul Islam (20101372)
4. Ha-mim Ahmad (20101286)

Of Summer, 2023 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science and Engineering on September, 2023.

Examining Committee:

Supervisor:
(Member)

Farig Yousuf Sadeque

Dr. Farig Yousuf Sadeque
Assistant Professor
Department of Computer Science and Engineering
Brac University

Program Coordinator:
(Member)

Dr. Md. Golam Rabiul Alam
Professor
Department of Computer Science and Engineering
Brac University

Head of Department:
(Chair)

Dr. Sadia Hamid Kazi
Chairperson and Associate Professor
Department of Computer Science and Engineering
Brac University

Abstract

Abstractive text summarization is the technique of generating a short and concise summary comprising the salient ideas of a source text without making a subset of the salient sentences from the source text. The introduction of transformer models such as BART, T5, and PEGASUS has made this sort of summarization process more efficient and accurate. The objective of this paper is to analyze the performance of different transformer models, compare them to find an efficient model and fine-tune the model on csebuetnlp/xlsum English corpus. The performance of the generated summaries from the fine-tuned PEGASUS models is evaluated using the ROUGE metric, which basically compares the auto-generated summaries with human-created summaries. The fine-tuned PEGASUS model gives a state-of-the-art performance on the XLSum English Corpus.

Keywords: Text Summarization; Transformer; T5; PEGASUS; BART; ROUGE Score; SHAP.

Table of Contents

Chapter 1

Introduction

The amount of written content in modern culture is overwhelming and covers a wide range of topics, including news journalism, scholarly research, social media posts, and digital reviews. Automated text summarizing methods had to be created in order to effectively condense and understand the vast amount of information available due to the explosion of content. Text summarization is a specific branch of Natural Language Processing (NLP) that aims to extract the most important information from texts and display it in a concise, understandable manner. These approaches enable readers to quickly understand crucial ideas and facts without reading the entire document.

Extractive summarization and abstractive summarization are two broad categories of text summarizing approaches. In order to create the summary, extractive summarization chooses a subset of important sentences from the original text. In contrast, abstractive summarization produces a unique summary that captures the key points without necessarily recycling entire passages from the source material. Although employing pre-existing sentences simplifies extraction approaches, they frequently fail to preserve coherence and capture the entire context. While producing summaries that are more coherent and contextually accurate, abstractive approaches nevertheless add more complexity by necessitating the maintenance of factual integrity and the resolution of ambiguities.

The increased need for advanced NLP systems has sparked research projects to determine how well pre-trained language models like Pegasus can be tuned to enhance abstractive summarization. Traditional sequence-to-sequence architecture-based attention-enhanced abstractive summarization models have demonstrated promising results. They do have certain limitations, though, including the occasional use of redundant terminology and concerns with variable quality and terms that are not commonly used. Sequence-to-sequence models' slowness in concurrent data processing is one of their significant drawbacks. The self-attention mechanism built into the transformer architecture, which enables parallel computing over the whole input sequence, mitigates this. The transformer is restricted in terms of its computational complexity to $O(n^2d)$ as the layer operations are carried out on words that are arranged sequentially.

In our research, we deployed BART, T5, and PEGASUS transformer models and evaluated their performance using the CNN-Daily Mail dataset. Following this assessment, we fine-tuned the most effective model utilizing the XL-sum dataset to optimize its performance.

1.1 Research Problem

As the volume of scholarly publications, articles, and legal documents continues to expand, the task of identifying and retaining key information for specific research or legal purposes becomes increasingly challenging. Given the constraints of time and resources, it is often impractical for professionals to thoroughly read and comprehend every extensive document pertinent to their field. Consequently, succinct and accurate summaries of such texts can serve as invaluable tools, enabling researchers and other specialists to quickly assimilate essential information. These summaries facilitate the efficient extraction of vital insights from a range of sources, thereby advancing scholarly or legal inquiries.

Our forthcoming research paper aims to conduct a performance analysis of abstractive text summarization using state-of-the-art transformer models, specifically BART, T5, and PEGASUS. The goal is to identify the most effective model in terms of both training efficiency and parameter optimization. Upon determining the optimal model, we intend to fine-tune it using the newly available XL-sum dataset to further enhance its summarization capabilities.

1.2 Research Objective

The research objective of the study on the performance measurement among different transformer models for abstract text summarization is to compare and evaluate the effectiveness of these models on limited hardware resources. The following specific objectives can be formulated for the study:

- Analyze the strengths and weaknesses of different transformer models for abstract text summarization.
- This analysis aims to compare the respective performances of different transformer models on the proposed matrices, which are ROUGE-1, ROUGE-2, and ROUGE-L.
- Analyze and identify how models are making certain decisions, as well as diagnosing problems like over-fitting, under-fitting, data leakage, and so on.
- Examine the computational training time and parameters of different transformer models on training data. The main objective is to find an efficient model that works well with limited hardware resources and can be deployed on a large scale.

1.3 Workflow

In our upcoming thesis, we will take an in-depth look at all of the machine learning models suitable for abstractive text summarization. Our workflow starts with an initial analysis of prominent models such as PEGASUS, BART, and T5, and a selection of datasets like CNN/Daily Mail, XL-Sum, XSum, C4, and HugeNews Dataset. After a meticulous analysis, we will opt for the most suitable models and datasets for our work. Following this selection, we will proceed with data preprocessing, which involves train-test splitting and tokenization. Train-test splitting involves dividing the dataset into different subsets. In the conventional approach, a distinct subset is employed for training the model, while another one is utilized for testing its performance. The objective of this split is to assess the model's performance on unseen data. Tokenization is a crucial part of preparing textual data for many types of NLP models. For fine-tuning, parameters like batch size, number of epochs, and learning rate will be taken into consideration. After that, we will conduct a rigorous evaluation process based on suitable metrics for the summarization task. To better understand our model's behavior, we will employ an interpreter library like SHAP. This tool enhances the transparency and comprehension of the model's decision-making process by providing a prediction of the impact of each input token on the final summary. Finally, fine-tuned models will be compared against existing state-of-the-art models to gauge their performance and contribution to the field. The workflow ends with a conclusive discussion, synthesizing the findings, and laying down the path for future research in abstractive text summarization.

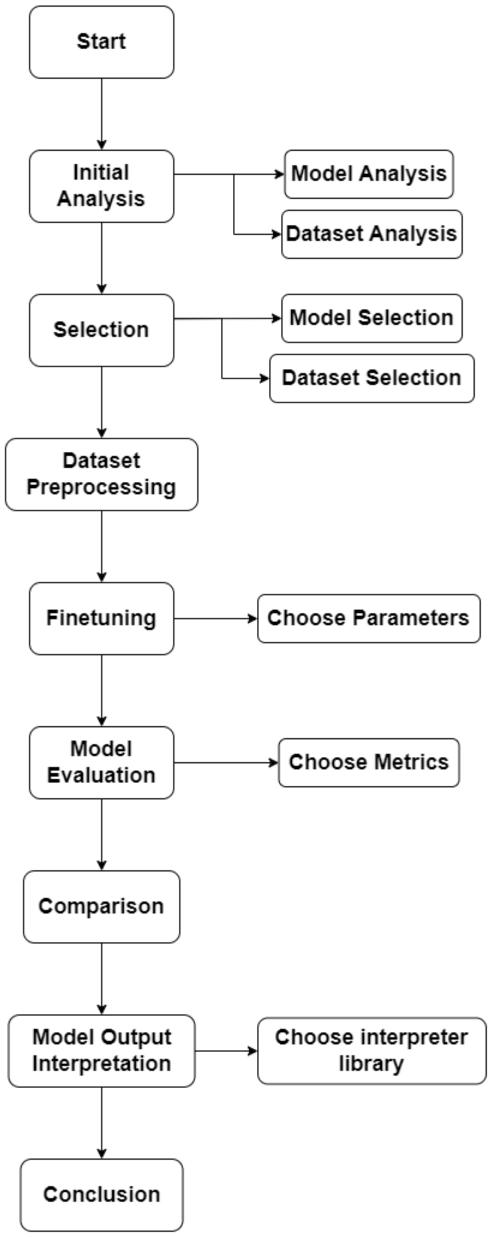


Figure 1.1: Workflow

Chapter 2

Literature Review

Summarization can either be extractive or abstract. The important terms from the source text are used to construct the summary in extractive summarization. Abstractive summarization generates a summary by comprehending the source text and rephrasing sentences in a shortened version. In 2022, Abdel-salam et al. studied the performance of extractive text summarization in different BERT models. They have used BERTSum as a baseline model for the research. A learning rate of $2 * 10 - 3$, a batch size of 3000, and a dropout rate of 10% were used to train the DistilBERT model with the Adam optimizer. After 30,000 iterations of training with the CNN/DM corpus, the ROUGE scores of the enhanced summarizer were calculated. DistilBERT had poorer ROUGE-1, ROUGE-2, and ROUGE-L scores than BERT-base by 1.6%, 3.5%, and 1.9%, respectively. The time needed to train the two models is proportional to the number of parameters they include, and the DistilBERT model contains 36% fewer parameters than the BERT model. In the fine-tuned SqueezeBERT summarizer, the hyperparameters are identical to those of the DistilBERT model. When applied to the SqueezeBERT experiment, the model produced ROUGE scores that were either on par with or marginally better than those obtained in the preceding DistilBERT experiment. SqueezeBERT is 1.6%, 3.35%, and 1.8% below BERT-base on the ROUGE-1, ROUGE-2, and ROUGE-L scales, respectively. When compared to the BERT model, the SqueezeBERT model requires 49% fewer parameters, which is closely correlated with the length of time required to train these models. One interesting finding is that the SqueezeBERT summarizer kept around 98% of the baseline model's summary performance despite a 49% drop in size. Since the original BERT-base summarizer has over 120 million parameters but the recommended model only has 62 million, the compression capabilities of the architecture allow for the design and use of the SqueezeBERT summarizer for real-time summary generation [25].

BERT Models	ROUGE Scores			Params
	ROUGE-1	ROUGE-2	ROUGE-L	
BERT-base	43.23	20.24	39.63	120.5M
DistilBERT	42.54	19.53	38.86	77.4M
SqueezeBERT	42.54	19.56	38.92	62.13M

Research from See et al. (2017) emphasized abstractive summarization more since it ensures abilities like paraphrasing, generalizing, and incorporation of real-life knowledge (See et al., 2017). The uncertainty of languages made the abstractive approach challenging until the development of sequence-to-sequence models. Though RNNs were used, the system still has issues, such as a lack of attention to detail, a failure to handle OOV (Out-of-Vocabulary), and word repetition. Therefore, to address such issues, See et al. (2017) proposed three models. 1) Baseline sequence-to-sequence model 2) Pointer generator model 3) Coverage mechanism model In the sequence-to-sequence model, as tokens were given to the encoder at each stage, a series of encoder hidden states h_i were generated. In order to construct the next word in the phrase, the decoder was given the embedding of the previous word and then used the probability distribution of the original words. The attention distribution a^t was used to produce the context vector h_t^*, which was then added to the decoder state to produce the vocabulary distribution with learnable parameters b, $b^{'}$, $V\ and\ V^{'}$.

$$h_t^* = \sum_i a_i^t h_i$$

$$P_{vocab} = softmax(V^{'}(V[s_t, h_t^*] + b) + b^{'})$$

The Pointer Generator Network, the second model, could manage OOV words while still preserving its capacity to produce new words, unlike the baseline sequence-to-sequence model. Using a predefined vocabulary, the network generated words, and using pointing, it copied those words verbatim from the source. Similar results were obtained when comparing the attention distribution a^t and the context vector h_t^* to the baseline model. The Generation Probability p_{gen} was utilized as a sort of soft switch to select whether to generate a word from the vocabulary or reproduce a word from the input sequence. The Coverage Vector reflected how well-covered a set of words was and was calculated by adding up the attention distributions for each timestep in the decoder. It consisted of a learnable parameter vector that took into account the attention mechanism's previous decisions when making a new one, thus facilitating the attention mechanism's ability to avoid producing redundant text. For experimentation, CNN/Daily Mail dataset containing long text was chosen. The standard ROUGE metric was used for results evaluation. The results showed the baseline model's poor performance, while the pointer generator model achieved a higher ROUGE score. But it still had repetition problems. Hence, the coverage mechanism was complemented with the pointer generator to eliminate such problems, and surprisingly, it surpassed the existing best abstractive models, achieving a much higher ROUGE score. In conclusion, See et al. (2017) model had numerous abstractive capabilities; nevertheless, how to achieve further levels of abstraction is still an issue that has to be researched [9].

Generating short summaries of a text may provide a brief knowledge of the contents of the text, but it may still miss out on some of the essential features of the given text. A new summarization method: the hierarchical structure was used to extract the more salient information about longer research papers, books, or legal documents. It primarily focused on extractive summarization using the multitasking approach. Three long datasets, such as arXiv-Long, PubMed-Long, and Longsumm, were used in this method, which generated similar or better performance than the baseline (BERTsum). First, the Longsumm dataset was used, where both extractive and abstractive sets were used to train the dataset. It contained 1969 papers, of which 20% were used for validation. To use the arXiv-Long and PubMed-Ling datasets, only the datasets containing 350 tokens were considered. That generated a total of 88,035 PubMed-Long instances and 11,149 arXiv-Long instances. Then the trained datasets were passed on to the model, which was named a section-aware summarizer. It considers the hierarchical structures of the documents. After the BERTsum representation of the texts, a linear classification model was added on top of that. For each text output, there are two losses incurred: sentence selection loss and section prediction loss. The sentence selection predicts if the sentence will be included in the extractive summary or not, whereas the section prediction judges if the sentence is from the correct section. After analyzing the result, it was found that irrespective of the token length, the multitasking approach performed better than the BERTsum baseline in Rouge-2. And the multitasking method chooses sentences from all parts of the texts for the extractive summary, whereas BERTsum majorly focused on the initial parts of the texts. As a result, salient information from all the segments of the texts can be found in the summaries [20].

To keep up with the ever-increasing output of articles and books, researchers have been exploring new techniques to condense lengthy works into understandable summaries. The pointer-generator approach is one of them. It's an improvement from the sequence-to-sequence model in almost all aspects. The pointer-generator method can be used for both abstractive and extractive summarization. It uses a multi-head attention mechanism, pointer-dropout, and two new loss functions, where the first two improve the ROUGE scores without improving the N-gram novelty, and the loss functions improve the N-gram novelty with a lower ROUGE score. The CNN Daily-mail dataset was used, where the training set contained approximately 287 thousand training pairs, with 13 thousand validation sets and 11 thousand training sets. Then the datasets were trained by the pointer-generator model. By using the pointing mechanism, this model extracts texts from the source text and also generates novel words. This is a major advantage of this model, as it is able to generate new words for its vocabulary. The source text is first given to a bidirectional LSTM, where, through encoding and decoding, the attention distribution of source words is calculated. The model also uses a coverage mechanism to prevent repetition. Following the calculation of a probability distribution, a trade-off between the probability distribution and the attention distribution takes place. The losses that occurred in each step are also kept track of. To reduce the model's dependency on the pointer network, a dropout mechanism was used that allowed the use of a copy mechanism only when it was essential. To maximize the usefulness of the pointer network, a multi-head attention mechanism was subsequently implemented. After the model's implementation, it was found that the ROUGE score of the model was similar to or slightly worse than the baseline's ROUGE score, but the N-gram novelty improved [12].

Relevance Measure and Latent Semantic Analysis are some noteworthy approaches among several approaches to text summarization. Relevance measure can be performed using the Standard IR method while identifying and extracting semantically important sentences required by the Latent Semantic Analysis technique. Both approaches choose highly ranked and unique phrases. Between the Query-relevant summary and Generic summary, Gong and Liu (2001) put emphasis on the latter. The aim was to summarize the document's essential points with fewer repetitions.

It was quite challenging to develop a generic summary that included the document's crucial points with minimum redundancy since no query or topic was passed to the process. Hence, the process of evaluation was even more challenging. Generic text summaries are important for defining the category and presenting the key points of the documents. In the process, the Relevance Measure was performed first to attain the highest relevance score. Based on the Relevance score, a sentence was added to the summary. This addition was to be done in a loop until the summary reached the predefined value. Relationships between the terms were captured and modeled with the help of the Latent Semantic Analysis technique, thanks to the use of the SVD (Singular Value Decomposition) concept. The conventional IR method lacks such modeling. The results of the Standard IR method, the Latent Semantic Analysis method, and three distinct human summarizers were compared and contrasted. In the case of lengthy documents, the level of disagreement amongst these individuals seemed to be rather significant. For summary performance evaluation, recall, precision, and f-score were used. In addition, the effect of several VSM weighting strategies on text summarizing performances was explored as part of the evaluations. Finally, the factors that contribute to the discrepancies between evaluators' handwritten summaries were investigated, and human text summarizing patterns were examined [1].

Nallapati et al. (2016) modeled abstractive text summarization with Attentional Encoder-Decoder Recurrent Neural Networks and obtained state-of-the-art performance on two independent datasets. New models were developed to deal with the specific issues of abstractive summarization. The principal computational barrier is often avoided by reducing the size of the soft-max layer in the decoder, which is the focus of Encoder-Decoder RNN with Attention and a Large Vocabulary Trick. Additionally, this method accelerates convergence by restricting the modeling effort to the absolutely necessary words for the example being studied. Finding the main concepts and objects in the text that the summary is based on might be challenging. To do this, they created separate embedding matrices based on lookups for the vocabulary of each type of tag. One natural approach to dealing with out-of-vocabulary (OOV) words is to simply make reference to their original context. They modeled this idea using a novel switching decoder/pointer architecture. This architecture uses a "switch" in the decoder to choose between using a pointer or the generator at each time step. When the switch is activated, the decoder functions normally, generating a word from the target vocabulary. The decoder will generate a pointer to one of the original document's word positions if the switch is turned off. In order to construct a summary from a vast corpus of source documents, they proposed a model that uses two bi-directional RNNs on the source side, one at the word level and the other at the sentence level, to identify the key phrases. Each of the proposed novel models yielded improved performance [5].

The usage of the graph-based algorithm "TextRank" for text processing and its integration into natural language applications are discussed in the research paper (Mihalcea, n.d., 2004). Graph-based ranking algorithms evaluate the importance of each vertex in a graph. "Voting" or "recommendation" refers to the process by which they spread significant and less important information. When a vertex is joined to another, a vote is cast. The importance of a vertex is proportional to the number of votes cast for it. An individual vertex's score is derived from the votes cast for it. The text has been portrayed as a graph in the paper. The text is converted into a node in a graph by adhering to four rules. Mihalcea, n.d. (2004), has talked about two use cases: one is keyword extraction, and another is sentence extraction, which can be seen as an extractive summarization problem. For keywords, phrases or unigrams are considered nodes in the graph, and for the second use case, the entire sentence is considered a node in the graph. The text units functioning as an edge between the vertices are then related to one another. When PageRank's value for the nodes does not significantly change and stays within predetermined bounds, the algorithm stops. The vertices are then arranged according to their scores. TextRank is effective because it incorporates data that is recursively taken from the entire text. Based only on data taken from the text itself, it was successful in determining which phrases were the most significant in a text [2].

The concept of information overload is attributed to the increasing prevalence of internet usage. Text summary is a potential technique that might be employed to address this issue. Document summarization is a technique employed to decrease the length of a document while retaining every important detail. The resulting summary often spans approximately half the length of the original text. The initial step involves the identification and extraction of key concepts from the original text through the creation of a symbolic representation. During the second stage, a summary is generated using the intermediate form. Many methods have been created specifically for summarizing texts. There are two distinct approaches to condensing information: extraction and abstraction. The procedure of extracting information from a source entails the act of isolating terms from their initial context and integrating them into a concise summary. The process of abstraction involves generating novel sentences for the purpose of summarization. The nature of a summary can vary between being informative or suggestive, contingent upon the specific details included. Based on content, there are two primary summarization methods: generic summarization and query-based summarization. Since it is independent of the text's subject matter, users of all kinds can use the generic summarizing approach. The query-based summarizing technique employs a question-and-answer structure, where the summary is derived from the response provided to the inquiry. The input for a single-document summary is constrained to a singular document. A multi-document summary can be generated by inputting multiple papers pertaining to the same subject. A monolingual system is a computer system that only understands one language and can only process and provide results in that language. One notable benefit of multilingual systems is their capacity to effectively analyze and interpret material written in multiple languages while also generating summaries in various languages. With the statistical method, the score of a sentence is determined after taking into account characteristics such as the sentence's title, its location within the text, and the frequency of the phrase. Additionally, weights are assigned to keywords to further influence the computation. The selection of the summary is subsequently determined by identifying the phrase that possesses the highest score. Statistical approaches employ several methodologies, such as the location method, TF-IDF method, Lexical chain method, and Graph theory method, among others. When endeavoring to offer a succinct summary of an extensive text, the use of clustering proves to be a valuable technique for discerning and categorizing segments that exhibit resemblances in terms of content or vocabulary. The K-means algorithm is used to find the best arrangement of clusters [3].

Sankarasubramaniam et al. (2014) took an approach to text summarization by linking Wikipedia with graph-based ranking. Sankarasubramaniam et al. (2014) suggested an iterative ranking method for summarizing inside a bipartite sentence-concept framework. Numerous convergence points were produced through a mathematical study and the design of the iterative ranking algorithm. This was followed by a generalization and analysis of the framework to include directional/weighted edges, query-focused summaries, and multi-document summarization. In the end, the ROUGE measure and a user were both used to assess how well the suggested algorithm performed. To gauge the efficacy of NLP-based automated summarization, researchers have developed the ROUGE software suite. Sankarasubramaniam et al. (2014) also discussed incremental summarization, which allows viewers to instantly read a summary of extra material. Sankarasubramaniam et al. (2014) talked about Wikipedia-based single-document summarization and Wikipedia-based summarization of multiple documents. A news story, a chapter of a book, or any other discrete chunk of text may serve as the input for a single document summarizing. The ideas in the bipartite sentence-concept graph correspond to the Wikipedia article titles that most closely resemble the input sentences. The sentences are then sorted in order of their likelihood of being used in a summary. The input of a multiple-document summarizing process is often a group of linked documents. The same sentences may appear in the inputs, which might make it difficult to choose summary statements. The steady-state sentence was determined with the help of iterative updates, and sentence ranks were derived based on a diversification objective. Finally, it was found that by leveraging Wikipedia, the quality of the summary can be significantly improved [4].

A further investigation was conducted to provide a synthesis of scholarly articles pertaining to the topic of COVID-19. The rapid development of the COVID-19 epidemic resulted in a scarcity of available data. The study was split into two separate parts: an unsupervised extractive component and a novel abstractive component. The extractive method involves the extraction of a significant portion of text from the original document, whereas the abstractive approach generates new summaries by paraphrasing the content of the original paper. The initial approach involved converting each text into a 768-dimensional representation and then applying K medoid clustering. Next, the abstractive approach was employed to construct novel summaries utilizing the provided keywords. To provide outcomes, many model designs were employed. The BERT model employed in the aforementioned paper utilized a transformer encoder model that had been pre-trained. The GPT-2 model is utilized to train both the language modeling problem and the multiple-choice prediction task. Language modeling is a computational technique that utilizes contextual information and prior words to predict the next word in a given text. On the other hand, multiple-choice prediction involves selecting the most appropriate summary from a set of summary possibilities. The utilization of the GPT-2 model for training these models was motivated by its auto-regressive nature, which renders it very compatible with text summarizing tasks. The model's findings did not exhibit any signs of overfitting. The assessment employed the F score variant of the ROGUE measure, known for its high precision in evaluating performance. The provided strategy proved beneficial for text summarizing. This paradigm possesses the potential for use in various other domains of medical research, enabling researchers to effectively cope with the exponential growth of scientific material [15].

The PEGASUS model is a sequence-to-sequence model optimized for abstractive text summarization through the use of gap-sentence generation as a pre-training objective. Similar to an extractive summary, PEGASUS takes an input document and masks important sentences before generating a single output sequence from the remaining text. As a pre-training self-supervised objective for downstream summarization tasks, it was observed that masking full sentences and producing gap sentences from the rest of the text was useful, which was named the Gap Sentences Generation (GSG) by Zhang et al. (2019). In GSG, each chosen gap sentence has a mask token to replace with its corresponding position in order to provide information to the model. Three strategies were selected for selecting gap sentences. Such as RANDOM, LEAD, and PRINCIPAL (Zhang et al., 2019). RANDOM selects m sentences randomly, while LEAD chooses the first m sentences. PRINCIPAL prioritizes sentences by calculating scores based on how they compare to the rest of the document. Though the initial model $PEGASUS_{BASE}$ included both GSM and MLM (Masked Language Model), MLM did not enhance downstream tasks at a significant number of pre-training stages. Hence, it was decided to leave it out of the final model $PEGASUS_{LARGE}$. (Zhang et al., 2019). In terms of selecting a pre-training corpus, C4 and HugeNews were considered for training the $PEGASUS_{LARGE}$ model on them. For downstream summarization, 12 public abstractive summarization datasets were used (Zhang et al., 2019). For example, XSum, CNN, NEWSROOM, Gigaword, and so on. Initially, a reduced-size model with 223 million parameters, $PEGASUS_{BASE}$ was conducted by using 4 out of 12 datasets for faster computation. Then the pre-training was scaled up by introducing $PEGASUS_{LARGE}$ with 568 million parameters, which used all 12 datasets. Finally, the result showed that $PEGASUS_{BASE}$ and $PEGASUS_{LARGE}$ had huge performance improvements on downstream datasets. $PEGASUS_{BASE}$ was able to achieve SOT on a number of datasets, but $PEGASUS_{LARGE}$ was successful in exceeding SOT on all downstream datasets. Moreover, since it is often challenging to accumulate a large number of supervised instances for the purpose of fine-tuning a summarization model, the top 10k training examples from each dataset were selected for use in fine-tuning $PEGASUS_{LARGE}$ for low resource summarization. From the data, it was seen that with just 1000 fine-tuning samples, it outperformed the state-of-the-art on 6 of 12 datasets (Zhang et al., 2019). Finally, the human assessment was used to validate the results, and it proved that the proposed model summaries could deliver human-level summarization performance on a variety of datasets [22].

Gupta et al. (2022) made a comparison between pre-trained models that are used for summarizing text, and these models are based on transformer architecture. The transformer models are compared based on the dataset from BBC News. Manually summarizing texts can be highly time-consuming. Automating the summarization process will ease the situation. Besides, text summarization will also reduce the size of files and eliminate problems related to storage. They focused on abstractive summarization, as this process creates accurate and coherent summaries. They pre-processed the dataset by using the following pre-processing techniques: lower casing the input text, eliminating punctuation from the text, eliminating frequently occurring words, stemming, lemmatization, and contraction mapping. They made use of the Hugging Face transformer library, which serves as an open source for numerous NLP libraries and datasets. They used some pre-trained models from the transformer library for summarizing texts. They used the pipeline model, the BART model, which contains 10% more features than BERT, and words are encoded differently based on their position in the sentence, the T5 model, and the PEGASUS model. Text summarization was done by each of the models, and they were compared based on their ROUGE scores. It was observed that T5 had the highest ROUGE score and the pipeline model had the least ROUGE score. So, the pipeline model gave the least admissible results. From their study, we can see that the T5 model outperformed all the other models [26].

In 2020, Pilault et al. presented a method employing neural abstractive summarization to generate abstractive summaries of extensive texts, often exceeding a few thousand words. Their approach involves an initial extractive step to train the transformer language model on relevant data before it produces a summary. Notably, this technique achieves superior ROUGE scores and yields more abstractive summaries in comparison to previous methodologies that utilized a copy mechanism. Two separate trainable components make up their model: 1) An extractive model, which consists of a hierarchical encoder that generates sentence representations and is used to either identify or categorize sentences in the input and 2) The transformer language model in the study by Pilault et al. relies on both the sentences extracted via the extractive model and either the entire or a subset of the input document. In terms of implementation, they utilized two neural extractive models, each featuring a hierarchical bidirectional LSTM encoder with both word and sentence-level LSTMs. In both configurations, an LSTM was employed as the decoder. Their approach aligns with earlier work, notably the study by Nallapati et al. in 2017. In order to identify ground truth extraction targets, the average ROUGE scores (ROUGE-1, ROUGE-2, and ROUGE-L) were calculated for each sentence within the document in comparison to each sentence within the generated summary. This computational approach informs the selection of sentences that most effectively represent the essence of the original text.

$$\text{SCORES extraction} = \left\{ \tfrac{1}{3} \textstyle\sum_{r \in 1,2,L} \text{ROUGE}_r \left(S_i, S'_j \right) \mid S_i \in D; S'_j \in T \right\}.$$

They increased the number of ground-truth extraction sentences per output summary sentence to two because a single-sentence extraction may not always offer the same information content as a target summary. To do this, the top 2 sentences in D with the greatest $SCORES_{extraction}$ in relation to a particular sentence in T are chosen. The 2M ordered phrases that arise serve as the context for the TLM. The extractive summarization model provides a more structured and larger context for the TLM during training. Similar to the pointer network, they also utilize a hierarchical LSTM to encode the text and generate a series of sentence representations, denoted by the notation d_1 to d_N, where N is the total number of sentences in the input. Following are the steps taken to calculate the final document representation: where b, d and W are learnable parameters. Last but not least, the following equation provides the likelihood that each sentence is an extractive summary:

$$\mathbf{d} = \tanh\left(\mathbf{b}_d + \mathbf{W}_d \frac{1}{N} \sum_{i=1}^{N} \mathbf{d}_i\right)$$

where $\mathbf{b}_d$ and $\mathbf{W}_d$ are learnable parameters. Finally, the probability of each sentence belonging to the extractive summary is given by:

$$o_i = \sigma\left(\mathbf{W}_o \begin{bmatrix} \mathbf{d}_i \\ \mathbf{d} \end{bmatrix} + \mathbf{b}_o\right)$$

where σ is the sigmoid activation function.

Pilault et al. opted for a distinct approach by using a single transformer language model that was trained from scratch with appropriately formatted data. This deviates from the more conventional sequence-to-sequence (seq2seq) frameworks that utilize encoder-decoder architectures for abstractive summarization. Their methodology employs an initial extractive step followed by an abstractive phase, demonstrating that transformer language models are capable of generating high-quality summaries for extended text passages.

Importantly, the researchers were able to quantitatively evaluate the advantages of incorporating an extractive stage. They accomplished this by comparing their approach with an alternative abstractive model that only had access to the input text. Their method outperformed previous extractive and abstractive summarization techniques when tested on the arXiv, PubMed, and bigPatent datasets. Additionally, their model exhibited a lower propensity for verbatim repetition of phrases or sentences from the source material, indicating a greater level of abstraction in the generated summaries [17].

Ensuring the accuracy of information between the produced summary and the original content poses a significant challenge in the field of abstractive summarization. Contemporary models that have been trained on preexisting datasets exhibit a phenomenon known as entity hallucination, wherein they generate references to entities that do not actually exist within the original text. One potential solution to address the issue of entity hallucination is to apply a filtering process to the training data and incorporate supplementary metrics for assessing the factual consistency of the summaries (Nan et al., 2021). In 2021, Nan et al. introduced three metrics specifically designed for Named Entity Recognition (NER) in summarization tasks. Let $N(t)$ and $N(h)$ denote the number of named entities identified in the target (gold summary) and the hypothesis (generated summary), respectively. The variable $N(h \wedge s)$ represents the number of entities in the generated summary that have corresponding entities in the source text.
The precision of a system, denoted by Prec_s, is calculated as follows:

$$\text{Prec}_s = \frac{\text{Number of True Positives}}{\text{Total Number of Instances}}$$

Here, $N(h)$ is used to measure the extent of hallucination with respect to the original text. This metric alone signifies the proportion of named entities that are both present in the source text and included in the summary.

A low value of Prec$_s$ suggests a high degree of hallucination. According to the authors, the root cause of the hallucination problem often resides in the training data itself.
The Spacy Named Entity Recognition (NER) tool was employed to identify all named entities within the gold summary of each dataset. If no named entity in the ground truth summary corresponds to the source text, that particular sentence is discarded.
To provide a comprehensive evaluation of the entity-level accuracy of the generated summary, Nan et al. introduced two additional metrics: the precision-target Prec$_t$ score and the recall-target Recall$_t$ score.

$$prec_t = N(h \wedge t)/N(h)$$

$N(h \wedge t)$ = Total named entities in the generated summary that can be matched with those in the ground truth summary.

$$recall_t = N(h \wedge t)/N(t)$$

$N(t)$ = the number of named entities in the ground truth summary.[24]

Redundancy is considered a serious problem while summarizing long documents, as there is always a tradeoff between importance and redundancy in phases called sentence scoring and sentence selection, respectively. The phrase redundancy reduction has been nearly forgotten in recent efforts on neural extractive summarization. Some of the suggested neural approaches do care for redundancy, but they either do so implicitly or limit themselves to relatively short documents. Hence, to address this vague situation, Xiao and Carenini (2020) classified redundancy reduction methods into three categories. 1) Decoders for implicitly taking redundancy into account. 2) Learning of redundancy reduction explicitly in the sentence scoring phase. 3) Selection of the less redundant sentence in the sentence selection phase. Two new metrics named Unique n-gram ratio and Normalized Inverse of Diversity (NID) were used in redundancy measurement. Additionally, three new methods were introduced that allowed for more explicit and adaptable redundancy reduction throughout the sentence scoring and selection phases. Xiao and Carenini (2020) initially examined two foundational models: Naive MMR and ExtSum-LG. Naive MMR re-ranks candidate sentences by optimizing a trade-off between query relevance and document novelty. In contrast, ExtSum-LG serves as a benchmark for evaluating different techniques for reducing redundancy.
Within the first category, the study investigated SummaRuNNer Decoder and NeuSum Decoder. The SummaRuNNer Decoder's drawback is its inability to explicitly manage redundancy, as its novelty component is not directly supervised. To address this, the NeuSum Decoder utilizes collaborative learning to score and select sentences with the highest scores.
In the second category, a new method known as RdLoss was introduced. Unlike its predecessors, RdLoss allows the decoder to exert control over the level of redundancy in the generated summaries.
The third category comprised the existing Trigram Blocking method and the newly proposed MMR-Select and MMR-Select+ methods. Trigram Blocking prevents a

sentence from being included in the summary if it does not share a trigram with the sentences already in the summary. MMR-Select enhances flexibility by relying on a trained neural model to compute importance scores. However, because sentence scoring and selection are decoupled in MMR-Select, the two phases could not mutually benefit. To address this, an extension, termed MMR-Select+, was proposed.
In summary, the study conducted a comparative analysis of the informativeness and redundancy of various approaches falling into three distinct categories. The authors demonstrated that their proposed methods achieved state-of-the-art performance on ROUGE scores while significantly reducing redundancy across two datasets sourced from scientific papers (Pubmed and arXiv). Furthermore, the study validated the efficacy of the sentence selection phase over the sentence scoring phase in reducing redundancy. [21]

Chapter 3

Description of the models

3.1 PEGASUS

The PEGASUS model, which stands for Pre-Training with Extracted Gap-Sentences for Abstractive Summarizing, is a text summarizing model that is constructed based on the transformer model architecture. It takes advantage of self-attention mechanisms, such as the ability to deal with long-range textual dependencies and perform computations in parallel. Unlike the classic Transformer model, PEGASUS optimizes pre-training for abstractive text summarization.

3.1.1 Internal architecture & functionality

The PEGASUS model employs a transformer-based design that incorporates an encoder-decoder framework. The text given by the user is subjected to encoding, resulting in a series of representations that can be utilized by the decoder to generate the intended output text. Both the encoder and the decoder consist of multilayer feedforward and self-attention neural networks. PEGASUS inherits this structure but alters the pre-training procedure. Self-attention is at the core of PEGASUS. This is the idea that a model can weigh the relative significance of words in a sentence, which allows for an understanding of context, dependencies, and relationships among words zhang2020pegasus.

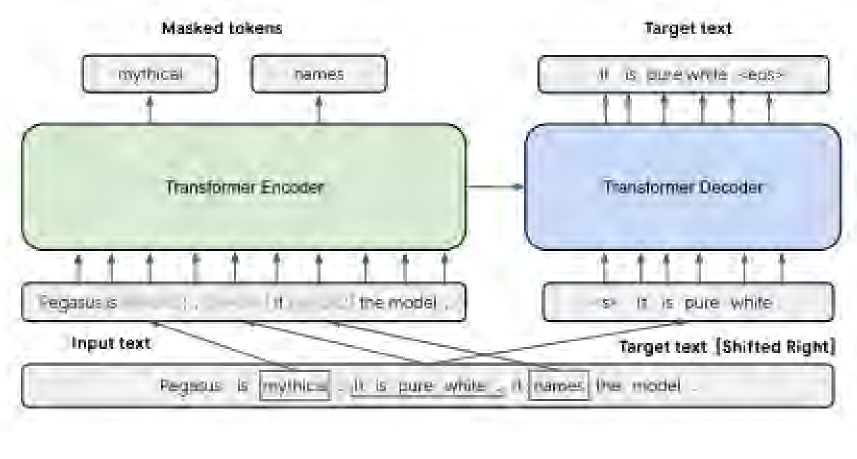


Figure 3.1: PEGASUS internal architecture

3.1.2 Pre-training

PEGASUS relies on a distinctive approach to self-supervised learning known as ”gap sentence generation” in its pre-training phase. The model employs a random selection process to mask specific words inside the document and then trains itself to reconstruct these omitted or ”gap” sentences. This approach differs from the prevalent practice of masking words within a text, as observed in BERT or RoBERTa. The utilization of pretraining is particularly well-suited for tasks like summarization, as it facilitates the model in acquiring an internal representation of the information, which helps in understanding its gist. Sentences are chosen to be masked depending on their significance to the overall semantics of the document. PEGASUS applies a standard masking strategy by masking the sentences that contain the most frequently occurring principal words (other than stop words). There are three primary techniques for choosing m gap sentences from a text without replacement from a document where the document has a total of n sentences. The first one chooses m sentences uniformly at random. Another one, chooses the initial m sentences. The last one selects sentences with the highest scores based on their level of significance. The calculation of ROUGE1-F1 (Lin, 2004) serves as a measure of relevance between the sentence and the rest of the document.

Pegasus is pre-trained on two large text corpora, namely C4 and HugeNews. The C4 corpus encompasses text derived from 350 million web pages, amounting to around 750GB of data. On the other hand, the HugeNews dataset comprises 1.5 billion articles, totaling 3.8TB in size, which were collected from news and news-like websites spanning the period from 2013 to 2019.

3.1.3 Sizes and Parameters

Depending on the application and computational resources, the size of the PEGASUS model may vary significantly. Various model sizes have been employed for the implementation, with each being specified by the number of attention heads, and the number of transformer layers. Google’s basic model consists of 12 encoder and decoder layers, with a hidden size of 768 and 12 attention heads. As a result, there are a total of 110 million potential parameters. Larger iterations, such as PEGASUS-large, have a substantial parameter count of 568 million, which is split among 24 encoder and decoder layers. These models possess a hidden size of 1024 and are equipped with 16 attention heads. PEGASUS has a complex architecture, yet it adheres to the idea of simplicity in its architecture to facilitate interpretability and modifications. The number of masked sentences used for pre-training is a crucial parameter in PEGASUS. This is a hyperparameter that can be adjusted based on the specific task at hand. A common value is to mask 15% of all sentences in the document.

3.1.4 Fine-tuning

Fine-tuning requires providing the model with a dataset that was particularly built for abstractive summarization. This dataset consists of pairs of source documents and their matching human-generated summaries, such as CNN/DailyMail, and XSum. The PEGASUS model is trained to generate summaries by conditioning on the source material and the gap-sentence(s), which takes place during the fine-tuning phase. The fine-tuning procedure often involves multiple iterations, where several gradient-based optimization techniques, like backpropagation and stochastic gradient descent, are employed. The objective is to minimize the level of discrepancy between the model-generated summaries and the human-generated summaries found in the training dataset.

3.1.5 Results

The model has demonstrated exceptional performance on many benchmark datasets, reaching state-of-the-art results. In the CNN/Daily Mail dataset, the PEGASUS model demonstrated a ROUGE-1 score of 44.17, along with ROUGE-2 and ROUGE-L scores of 21.47 and 41.11, respectively. In XSum, the model achieved a ROUGE-1 score of 47.21, a ROUGE-2 score of 24.56, and a ROUGE-L score of 39.25, respectively.

3.2 BART

Bidirectional and Auto-Regressive Transformers (BART) is a sequence-to-sequence model developed by Facebook AI Research that is both flexible and powerful. BART stands apart from the competition thanks to its innovative pre-training process, which involves denoising corrupted text. BART can read from both the left and the right sides, unlike standard Transformer models. It takes in a sequence, applies a random mask to some of it (the ”noise”), and then learns to predict the original sequence. BART's ability to accurately model the left and right contexts is greatly improved by this denoising autoencoder setup. Because of its bidirectional nature and the power of the denoising objective, the application of this technique exhibits robust performance over a wide array of subsequent tasks, rendering it a highly valuable asset within the realm of natural language processing. Besides, by fine-tuning specific tasks, BART has shown impressive performance in several applications, including text generation, translation, summarization, and comprehension.

3.2.1 Internal architecture & functionality

The BART model is based on the transformer architecture but uses various encoding and decoding methods. BART is made up of a denoising autoencoder, which has two primary components: an encoder and a decoder. After becoming corrupted during pre-training, it learns how to recover the original text. A contextualized representation of the input text is created by the encoder using a corrupted version of the text. The decoder then uses this representation to reconstruct the original input text. This enhances BART's understanding of semantic and syntactic patterns and enables it to produce more fluid, natural language.

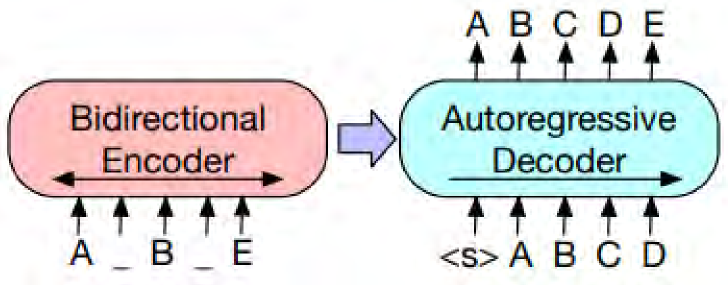


Figure 3.2: BART internal architecture

3.2.2 Pre-training

The BART model is pre-trained using a two-step procedure that includes destroying the text initially and then teaching it how to recreate it from scratch. In contrast to BERT, where certain tokens are just hidden, BART entails constructing a ”noisy” version of the input text. For this, there are several effective techniques: token masking, token deletion, text infilling, sentence permutation, and document rotation.
The reconstruction step is where BART learns. The model is trained to predict the original, uncorrupted text given the corrupted version. Throughout this stage,

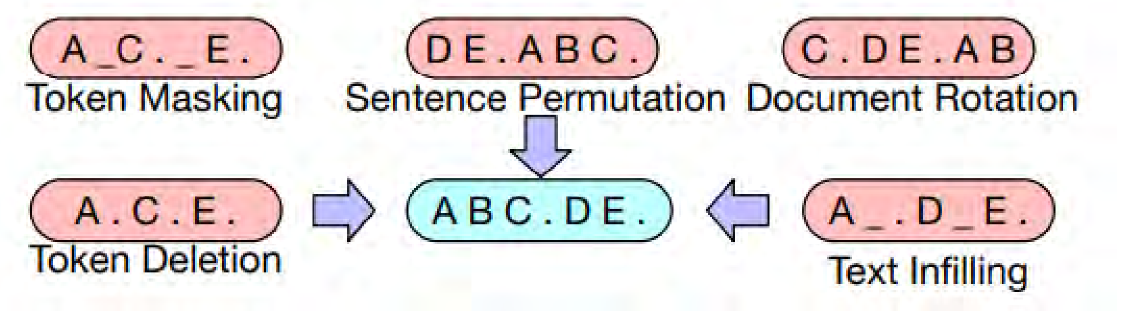


Figure 3.3: BART Pre-training

BART acquires a richer, more bidirectional understanding of the text and its underlying semantic and syntactic structure [13].

3.2.3 Parameters

BART is offered in a variety of versions that can be identified from one another based on how many parameters they each contain. The fundamental components of BART are a decoder with six layers and an encoder with six layers. The model contains close to 140 million parameters, with 12 self-attention heads per layer. The BART-large model, on the other hand, is the largest iteration of the model and has 16 attention heads distributed across 12 layers in each of the encoder and decoder. In BART, the model's weights and biases are referred to as the parameters. The BART model's large version, which has about 406 million parameters, can discern intricate linguistic patterns.

3.2.4 Fine-tuning

For sequence-generating tasks like abstractive summarization, BART's autoregressive decoder allows for direct fine-tuning. Denoising pre-training is closely related to such work since information is copied from the input but manipulated. When fine-tuned, the BART model is fed the full-length text and trained to produce the corresponding summary as output. The model's parameters are adjusted such that the difference between its predictions and the human-written summary is as small as possible. This is often quantified with a cross-entropy loss or similar sequence generation loss function. In addition, during the last stages of tuning, it is essential to fine-tune the training hyperparameters correctly. Variables like the training rate, batch size, and epoch count are examples. Depending on the dataset and the task, the ideal values for these hyperparameters may change.

3.2.5 Results

On summarization tasks, BART models received potent ROUGE scores, outperforming earlier state-of-the-art models on benchmark datasets. For instance, on the CNN/Daily Mail dataset, the original research says that BART obtained a ROUGE-1 score of 44.16, a ROUGE-2 score of 21.28, and a ROUGE-L score of 41.21. However, these ratings are subject to change based on the specific task, dataset, and process used to fine-tune them.

3.3 T5

Researchers at Google have proposed a ground-breaking new way of approaching natural language processing (NLP) challenges using their T5 (Text-to-Text Transfer Transformer) model. In contrast to prior transformer models, T5 treats every NLP task as a text-to-text issue, giving a uniform framework that greatly simplifies the implementation of different tasks. The pretraining consists of both supervised and self-supervised training. The GLUE and SuperGLUE benchmarks offer the downstream tasks that are used for supervised training. In self-supervised training, 15 percent of tokens are unilaterally removed to take advantage of tainted tokens, remove them, and exchange them with singular sentinel tokens (if a string of tokens is chosen for elimination, the whole string is substituted with a single sentinel token) (T5, n.d.). The distorted phrase is what goes into the encoder, whereas the original sentence is what goes into the decoder. Both of these sentences are inputs (T5, n.d.). The eliminated tokens, bordered by the sentinel tokens, make up the target. Padding for encoder input may be performed on either the left or the right. T5 is available in a range of sizes:

- t5-small
- t5-base
- t5-large
- t5-3b
- t5-11b

The t5-small model is what we've settled on for our thesis. The size and capacity of the model are the key differentiators between the T5 and the T5-Small. T5, the original model, is bigger than T5-Small, which is the scaled-down variant. The parameter of the T5 transformer model is 11 billion. On the other hand, there are 60 million parameters at the T5-Small checkpoint (T5, n.d.).

3.3.1 The Architecture of T5

The T5 model incorporates the encoder-decoder structure of the original transformer model presented by Vaswani et al. in 2017. Multiple layers of neural networks that execute self-attention and feed-forward make up both the encoder and the decoder. The feed-forward neural network analyzes each point in the sequence independently, yet, due to its built-in self-attention mechanism, the model may focus on different parts of the input sequence when generating an output sequence.
The transformer block, which can be expressed mathematically as follows, is the key component of the T5 design.

$$\begin{aligned}\text{Layer Normalization: } & LN(x) = \text{LayerNorm}(x) \\ \text{Self-Attention: } & SA(x) = \text{SelfAttention}(LN(x)) \\ \text{Feed-forward Network: } & FFN(x) = \text{FeedForward}(LN(x))\end{aligned}$$

A transformer block is then represented as:

$$\begin{aligned}x &= x + SA(x) \\ x &= x + FFN(x)\end{aligned}$$

where x is the input to the block, SA(x) is the output of the self-attention layer, and FFN(x) is the output of the feed-forward neural network layer. The residual connections are represented by the "+" operation.

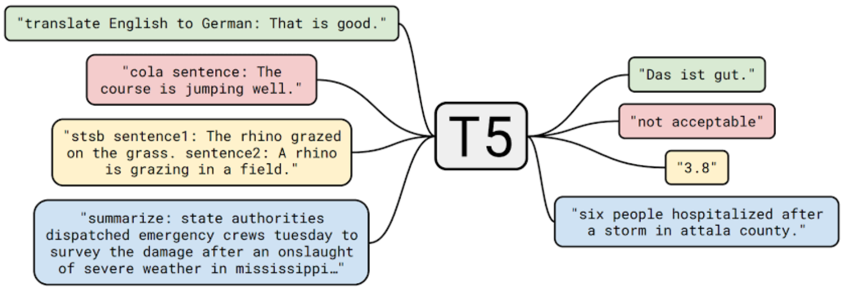

"translate English to German: That is good."
"cola sentence: The course is jumping well."
"stsb sentence1: The rhino grazed on the grass. sentence2: A rhino is grazing in a field."
"summarize: state authorities dispatched emergency crews tuesday to survey the damage after an onslaught of severe weather in mississippi…"
T5
"Das ist gut."
"not acceptable"
"3.8"
"six people hospitalized after a storm in attala county."

Figure 3.4: T5 transformer model (Raffel, 2019)

An illustration of the T5 model. Every job that we take into consideration, from translating and answering questions to sorting and categorizing, is reframed as training our model with some text as input in order to produce some output text. That allows us to utilize the same model across all of our different kinds of work.

3.3.2 The Text-to-Text Framework

The text-to-text foundation is what distinguishes T5 from other systems of a similar nature. Because both the input and the output of this method are text sequences, it treats every natural language processing challenge as a text generation task.
In this example, the "input" is a string of text written in the "source" language, and the "output" is a string of text written in the "target" language. The sequence of text that needs to be classified is the input, and the textual representation of the class label is the output, while text classification is being done.
To make this framework applicable to any NLP activity, the T5 model prefixes the text being entered with a task-specific prefix. As a result, the framework can be applied in any circumstance where NLP is necessary. When given information like "classify: The movie was amazing," T5 will output the word "positive," indicating that the review should be classified as "positive" or "negative."

3.3.3 Training the T5 Model

The "denoising autoencoding" method is used to train the T5 model. Some parts of the input text must be masked off (i.e., replaced with a particular symbol) before the model can be trained to predict the original text.
T5's training loss function is the ubiquitous cross-entropy loss, a metric for gauging how different a given word sequence is from the one that was really spoken:

$$L(y, \hat{y}) = -\Sigma y \log(\hat{y})$$

The Adam algorithm, a variation on stochastic gradient descent, is used during training to minimize this loss function [19].

3.3.4 Conclusion

The T5 model stands out from its competitors due to both its versatility and its uncomplicated style. This is accomplished by treating every NLP task as a text-to-text problem, which enables it to perform at the cutting edge on a variety of NLP tasks. Additionally, because of its design, which is based on the transformer model, it can effectively capture the relationships between words in a text even though they may be separated by a significant distance. [19].

Chapter 4

Description of the datasets

4.1 CNN/Daily Mail Dataset

4.1.1 Introduction

In our paper, we used the CNN/Daily Mail dataset. The CNN/DailyMail dataset is a widely used dataset. It is an English-language dataset containing slightly more than 300k unique news articles written by CNN and Daily Mail journalists. It was primarily used for machine learning and comprehension. But now the datasets are used in both abstractive and extractive summarizations. So, for our abstractive summarization, the CNN/Data Mail dataset is really useful. We have used version 3.00 CNN/Data Mail for our dataset. We have used pre-trained BART, Pegasus, and T5 models, which were trained on this dataset. The performances of these models are measured by rouge-1, rouge-2, rouge-L, and rouge-Lsum scores.
The average token count for articles and highlights:

Feature	Mean token count
Articles	781
Highlights	56

Table 4.1: The average token count for articles and highlights

4.1.2 Description and Split

A detailed description of the dataset:

Dataset Split	Number of instances in Split
Training Pairs	287,113
Validation Pairs	13,368
Testing Pairs	11,490

Table 4.2: Description of the dataset

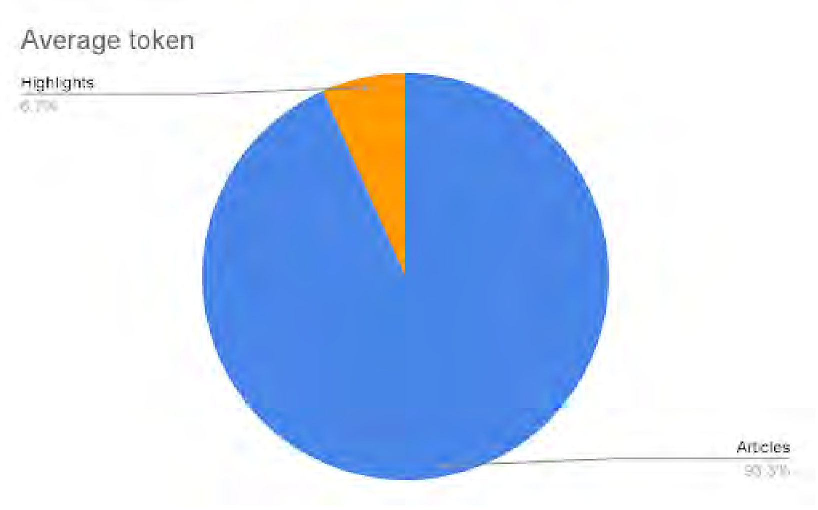


Figure 4.1: Average token count for articles and highlights

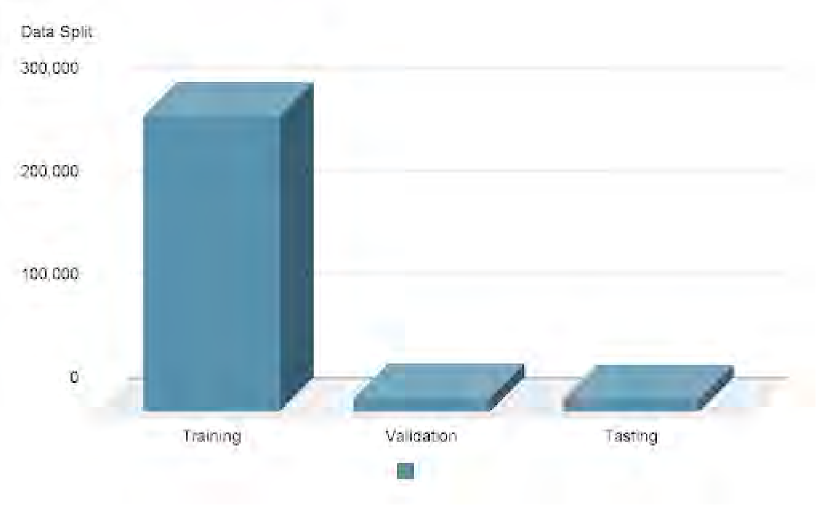
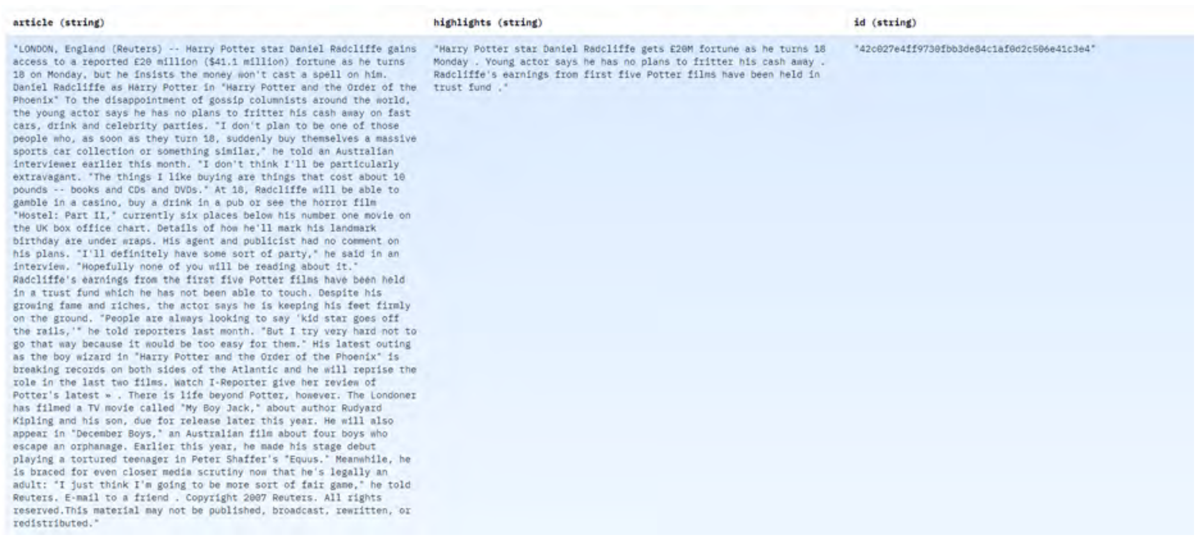


Figure 4.2: Split of the dataset

4.1.3 Sample Dataset



article (string)	highlights (string)	id (string)
"LONDON, England (Reuters) -- Harry Potter star Daniel Radcliffe gains access to a reported £20 million ($41.1 million) fortune as he turns 18 on Monday, but he insists the money won't cast a spell on him. Daniel Radcliffe as Harry Potter in "Harry Potter and the Order of the Phoenix" To the disappointment of gossip columnists around the world, the young actor says he has no plans to fritter his cash away on fast cars, drink and celebrity parties. "I don't plan to be one of those people who, as soon as they turn 18, suddenly buy themselves a massive sports car collection or something similar," he told an Australian interviewer earlier this month. "I don't think I'll be particularly extravagant. "The things I like buying are things that cost about 10 pounds -- books and CDs and DVDs." At 18, Radcliffe will be able to gamble in a casino, buy a drink in a pub or see the horror film "Hostel: Part II," currently six places below his number one movie on the UK box office chart. Details of how he'll mark his landmark birthday are under wraps. His agent and publicist had no comment on his plans. "I'll definitely have some sort of party," he said in an interview. "Hopefully none of you will be reading about it." Radcliffe's earnings from the first five Potter films have been held in a trust fund which he has not been able to touch. Despite his growing fame and riches, the actor says he is keeping his feet firmly on the ground. "People are always looking to say 'kid star goes off the rails,'" he told reporters last month. "But I try very hard not to go that way because it would be too easy for them." His latest outing as the boy wizard in "Harry Potter and the Order of the Phoenix" is breaking records on both sides of the Atlantic and he will reprise the role in the last two films. Watch I-Reporter give her review of Potter's latest » . There is life beyond Potter, however. The Londoner has filmed a TV movie called "My Boy Jack," about author Rudyard Kipling and his son, due for release later this year. He will also appear in "December Boys," an Australian film about four boys who escape an orphanage. Earlier this year, he made his stage debut playing a tortured teenager in Peter Shaffer's "Equus." Meanwhile, he is braced for even closer media scrutiny now that he's legally an adult: "I just think I'm going to be more sort of fair game," he told Reuters. E-mail to a friend . Copyright 2007 Reuters. All rights reserved.This material may not be published, broadcast, rewritten, or redistributed."	"Harry Potter star Daniel Radcliffe gets £20M fortune as he turns 18 Monday . Young actor says he has no plans to fritter his cash away . Radcliffe's earnings from first five Potter films have been held in trust fund ."	"42c027e4ff9730fbb3de84c1af0d2c506e41c3e4"

Figure 4.3: Example of the CNN/DailyMail dataset

4.1.4 Data Fields

ID: This is the SHA1 hash in hexadecimal format, which identifies the URL where we got the story for the dataset.

Article: This is the actual text of the news article.

Highlights: These are the key points or highlights of the article, as provided by the author.

Both whole news stories and their important highlights are included in the dataset. These articles provide background information when questions are answered. The highlighted phrases contain discrete things that are hidden, resulting in Cloze-style problems where the model's goal is to accurately guess the hidden entity. The highlighted sentences are combined to create a succinct summary of the article when it comes to summarizing.

The dataset includes stories from CNN and the Daily Mail, both of which were published between June 2010 and April 2015, respectively. These articles were taken from the Daily Mail and CNN archives, respectively.

This dataset is especially helpful for developing algorithms that can reduce lengthy passages of text into succinct summaries. In addition to making it simpler to understand dense quantities of information, summaries produced by models trained on this dataset will closely resemble the language used in the original articles.

4.1.5 Limitations

The dataset doesn't provide a significant number of summaries for the abstractive summarization. So it was a challenging task due to the lack of evolution of the abstractive summarization models. Also, as most of the CNN/ Daily Mail articles are based on the perspectives of the USA and the UK, there are some aspects of bias. The lengths of some articles in the dataset are significantly short, which makes it very challenging for the models to handle.

4.2 XL-Sum

4.2.1 Introduction

An enormous dataset called XL-Sum was created especially for the job of abstractive summarization. It stands out for its diversity; it includes more than a million pairs of articles with expert annotations and their summaries, all taken from the BBC.

Language diversity is one of its unique characteristics; XL-Sum offers information in 44 different languages, some of which lack widely accessible datasets. It is therefore extremely useful for research and other applications that need multilingual capabilities.

The dataset performs exceptionally well in a number of domains, according to both human assessments and intrinsic measures. It enables summaries that are both succinct and abstract while maintaining a high standard of quality. In general, XL-Sum is a reliable tool for creating and refining abstractive summarization models.

4.2.2 Variations

There are two versions of the XL-Sum dataset. The newer version of the dataset includes the Traditional Chinese language. By adding this language, the XL-sum dataset will enable better formatting, better extraction, larger evaluation splits, and more data. With the inclusion of the Traditional Chinese language, XL-Sum became the largest text summarization dataset publicly available.[23]

4.2.3 Split

Language	**Train**	**Dev**	**Test**	**Total**
Almost all other languages	80%	10%	10%	100%
English	93%	3.5%	3.5%	100%

4.2.4 Increased Usage

For abstractive text summarization, the usage of the XL-Sum dataset is now growing. In recent years, the usage of this dataset has increased to a great extent.

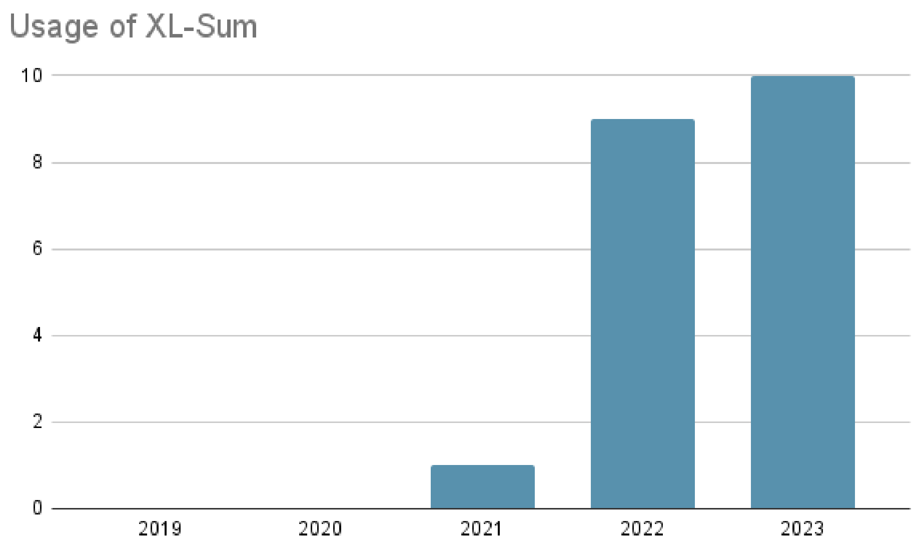


Figure 4.4: Usage of XL-Sum

4.2.5 Challenges

A lot of the languages had very little resources. As a result, those language samples were very less. So, those language evaluation sets were increased for a more reliable evaluation

4.3 XSum Dataset

4.3.1 Introduction

The XSum dataset is a massive collection of data that was developed specifically for the purpose of performing text summarizing tasks. The dataset was developed in 2019 and made public by academics at the University of Edinburgh. The XSum dataset is comprised of news articles obtained from a wide variety of sources, one of which is the internet's collection of online news websites. The articles that are included in the dataset cover a broad variety of subjects, including politics, sports, entertainment, technology, and many more. The articles were published during the years of 2015 and 2018, offering a temporal range of news coverage. The XSum dataset has been crucial in the advancement of research in extreme summarization and has acted as a standard benchmark for measuring the performance of a variety of different summarizing methods.

4.3.2 Variation

There are a number of different iterations or subsets of the XSum dataset that have been produced from it or utilized in research that is linked to it. Researchers at the University of Edinburgh developed XSum-RoBerta by first extracting a subset of articles from the initial XSum dataset and then creating additional summaries utilizing the Roberta model. A variant of the XSum dataset known as XSum-Newsroom has been created by the combination of the XSum dataset with the Newsroom dataset.

The Newsroom dataset includes numerous summaries for each article, as well as articles sourced from a variety of different news outlets. Researchers wanted to increase the amount of variety in the summary jobs, and one way to do this was to provide numerous summaries for each article. To do this, they merged the Newsroom dataset with XSum. The XSum-Hallucination dataset is a subset of the XSum dataset that was developed specifically for the purpose of assessing the hallucination capabilities of various summarization models. It is made up of articles that were part of the initial XSum dataset, but the summaries have been tampered with in a way that causes them to include false information or hallucinations. [8]

4.3.3 Split

Language	Train	Dev	Test	Total
Almost all other languages	90%	05%	05%	100%

4.3.4 Usage

Summarization Model Training: The XSum dataset is frequently utilized in the process of training and fine-tuning text summarization models, particularly those models that are focused on extreme summarization.
[16] **Model Evaluation and Benchmarking:** The XSum dataset is used as a standard against which the effectiveness of various text summarization techniques is measured. The researchers evaluate the quality and efficacy of their techniques by contrasting the generated summaries from their models with the reference summaries included inside the dataset.
[6] **Transfer Learning and Pre-training:** The XSum dataset has been put to use in activities involving transfer learning and pre-training. Researchers make use of the dataset to pre-train large-scale language models like BERT, GPT, and RoBERTa, which may subsequently be fine-tuned for particular summarization tasks.
[10] **Multimodal Summarization:** Using the XSum dataset, researchers have also explored multimodal summarization, which involves integrating textual information with either visual or audio elements.
Model Development and Improvement: The XSum dataset is utilized by researchers in order to create and enhance text summarization models.
[11]

4.3.5 Challenges

Length Constraint: This limitation makes summarizing harder since it entails condensing material while keeping coherence and essential ideas.
Single Reference Summaries: Having only one reference summary may limit models' ability to learn from diverse viewpoints and limit summary coverage of crucial information.
Abstraction: Both human annotators and machine learning models sometimes struggle with summarizing complicated and nuanced information into a single statement.
Limited Domain Coverage: When confronted with texts on topics that are not well represented in the XSum dataset, models that were trained on that dataset may face difficulty.

Generalization to Other Languages: The XSum dataset focuses mostly on summaries of English news articles. Models trained on XSum need to be translated or fresh datasets built in the target language before they can be applied. This creates difficulties in ensuring that summaries across languages are of the same high quality, style, and scope.

4.4 C4

4.4.1 Introduction

Being able to obtain reliable, high-quality datasets is essential for the development and evaluation of machine learning algorithms in the field of natural language processing (NLP). One such priceless resource that has drawn a lot of interest in the NLP field is the C4 dataset, which stands for "Colossal Clean Crawled Corpus." This massive dataset was produced by Google and is designed to encompass a variety of language patterns, making it a gold mine for scholars and practitioners.

The C4 dataset's diversity in language structures enables significant improvements in tasks involving language understanding. It is an essential tool for creating and perfecting models aimed at bettering our understanding of human language because of its quality and wide-ranging application. In conclusion, the C4 dataset fosters innovation in language understanding tasks in addition to acting as a cornerstone for NLP research.

4.4.2 Overview and Construction:

The C4 dataset which is a large-scale, publicly available dataset, is a massive collection of text data obtained by crawling and filtering web pages from various domains and languages which involves removing duplicates, spam, and low-quality content. Furthermore, the dataset undergoes language identification, sentence segmentation, and tokenization to ensure proper organization and uniformity of the text. It is a useful tool for developing and testing machine learning models used in NLP applications like sentiment analysis, machine translation, text categorization, and language modeling.

4.4.3 Scale and Variations:

The enormous scale of the C4 dataset is one of its distinguishing features. It started off as a collection of roughly 750GB of English-language material that was obtained from the open Common Crawl web scrape. As of now, the dataset comes in five variants-
c4/en: 806.87GB in JSON format
c4/en.noclean: 6.21TB in JSON format
c4/realnewslike: 36.91GB in JSON format
c4/webtextlike: 17.93GB in JSON format
c4/multilingual: 38.49 TB in JSON format

4.4.4 Split and feature

Most of the variants of the C4 dataset has two splits, training and validation. Each split contains a huge number of rows/examples. Features are the variables or attributes that provide information about each instance or sample in the dataset. They represent the different dimensions or aspects of the data that are relevant to the problem at hand. All versions of the C4 dataset has these five features: content-length, content-type, text, timestamp, url.[14]

4.4.5 Significance and Applications:

The availability of the C4 dataset has significantly impacted the NLP research community. It has served as a valuable resource for developing and benchmarking state-of-the-art models. Large-scale language models, such as GPT (Generative Pre-trained Transformer) versions, have been trained using the C4 dataset and have displayed astounding performance on a variety of language understanding tasks. Furthermore, the C4 dataset has facilitated advancements in transfer learning, where models pre-trained on the dataset can be fine-tuned on specific downstream tasks, leading to improved performance with less labeled data. The dataset's broad coverage of topics and languages enables models to generalize better and adapt to different domains and languages[18]

4.4.6 Usage

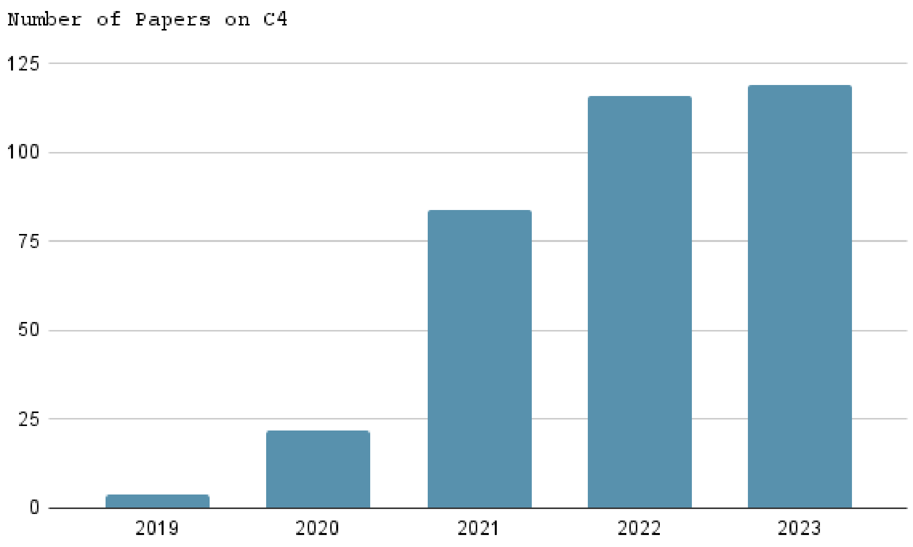


Figure 4.5: Number of Papers on C4.

4.4.7 Conclusion

The C4 dataset's variation and scale make it a valuable resource in the field of NLP. Its diverse linguistic coverage allows for cross-lingual applications and a better understanding of language patterns across different cultures and domains. Additionally, the colossal size of the dataset provides ample training examples, enabling the development of high-performance language models. As researchers continue to explore the potential of the C4 dataset, its variation and scale will remain essential for advancing language understanding capabilities and driving progress in NLP research.

4.5 HugeNews Dataset

4.5.1 Introduction

PEGASUS is a state-of-the-art model for abstractive text summarization tasks. Multiple datasets have been used and yielded significant results on the PEGASUS model. The HugeNews dataset is one of them. It's a fairly newly used dataset. The HugeNews dataset is mainly used in the pre-training of the data. It is a huge text corpus consisting of more than 1.5 billion text articles collected from newspapers and news-like websites. The size of the dataset crosses 3.8TB.

4.5.2 Extraction Process

Over a six-year period (2013-2019), the news pieces were extracted. First off, a whitelist of domains that ranged from high-quality news publishers to low-quality sites like high school newspapers and blogs was curated to produce the dataset. Then a web crawler was utilized to seed the domains, and heuristics were applied to discover articles that resembled newspapers to identify them. The primary article text was then extracted as plain text and used for summarization.

4.5.3 Mixture with C4

C4 is another huge scaled dataset used in PEGASUS. To improve the performance of the datasets a new dataset was introduced. It was a mixture of the C4 dataset and the HugeNews dataset. It is called the "Mixed and Stochastic" model. It is trained on 1.5M articles instead of 500k. The dataset uses a 20% uniform noise to sample key statements and then calculate importance scores. Also, the gap sentence ratio used in the model was consistently between 15% and 45%.

Dataset	**C4**	**HugeNews**	**Mixed and Stochastic**
xsum	45.20/22.06/36.99	47.21/24.56/39.25	47.60/24.83/39.64
cnndailymail	43.90/21.20/40.76	44.17/21.47/41.11	44.16/21.56/41.30
gigaword	38.75/19.96/36.14	39.12/19.86/36.24	39.65/20.47/36.76
newsroom	45.07/33.39/41.28	45.15/33.51/41.33	45.98/34.20/42.18
multinews	46.74/17.95/24.26	47.52/18.72/24.91	47.65/18.75/24.95
wikihow	43.07/19.70/34.79	41.35/18.51/33.42	46.39/22.12/38.41

Train/Dev/Test data

4.5.4 Usage

The HugeNews is a relatively new dataset. So its usage is really limited till now. The usage of HugeNews is mostly in the PEGASUS model only. And with the mixture of C4 and the increased performances, now HugeNews dataset's usage is increasing now.

Chapter 5

Preliminary analysis

5.1 PEGASUS

CNN/DailyMail dataset was tested using the pre-trained pegasus. The pre-trained google/pegasus-cnn_dailymail model was loaded from Hugging Face. Rouge-1, Rouge-2, Rouge-L, and Rouge-Lsum are the assessment metrics used to evaluate the model's performance. The obtained results are:

	Rouge-1	Rouge-2	Rouge-L	Rouge-Lsum
PEGASUS	0.47519	0.278492	0.373879	0.42883

Table 5.1: Results from the PEGASUS

The Rouge-1 score of 0.47519 shows that the PEGASUS model does a respectable job of capturing the overlap of unigram sequences between the generated summaries and the reference summaries. A higher Rouge-1 score indicates greater unigram recall and precision. The Rouge-2 score of 0.278492 indicates the model's ability to recognize bigram sequence overlap. Given that this score is lower than Rouge-1's, the model has difficulty delivering accurate summaries that are both precise and bigram-recallable. The Rouge-L score of 0.373879 determines how similar the generated summaries and the reference summaries are by using the longest common subsequence. This score demonstrates how well the model did at identifying longer word sequences. A higher Rouge-L score indicates better performance in terms of catching longer, integrated words. By taking into account both bigram and unigram sequences, the Rouge-Lsum score of 0.428834 assesses the overall quality of the generated summaries. This rating offers a thorough assessment of the model's effectiveness in producing accurate and illuminating summaries. Based on the results, it can be seen that the model does rather well in terms of unigram recall and precision (Rouge-1), but needs help capturing the overlap of bigram sequences (Rouge-2). The model does rather well at capturing longer, cohesive phrases (Rouge-L) and producing overall high-quality summaries (Rouge-Lsum).

5.2 BART

The pre-trained BART model from the Hugging Face library named "bart-base-finetuned-cnn_dailymail" was used for CNN/DailyMail dataset testing. The produced results from this model:

	Rouge-1	Rouge-2	Rouge-L	Rouge-Lsum
BART	0.311395	0.155579	0.249109	0.293648

Table 5.2: Results from the BART

This score is lower when compared to the Rouge-1 of the prior model, indicating that the recall and precision of unigrams may be worse in the present model. Similar to the Rouge-1 score, this model's Rouge- 2 score is lower than the score of the prior model, indicating that the current model has more difficulty producing accurate summaries in terms of bigram recall and precision. Once more, the Rouge-L score is lower than that of the prior model, suggesting that the present model would struggle to capture lengthier word sequences and keep the summaries coherent.
By taking into account both bigram and unigram sequences, the Rouge-Lsum score of 0.293648 assesses the overall quality of the generated summaries. This score, like the other metrics, is lower than the score of the previous model, indicating that the present model might produce summaries of lower overall quality.

Here, a learning rate of 5.6e-05, a batch size of 8 for training and evaluation, and a seed of 42 were used, according to an analysis of the hyperparameters. The optimization algorithm and its parameters were selected using the Adam optimizer, which has betas=(0.9,0.999) and epsilon=1e-08. The learning rate scheduler maintained a linear schedule. Four training epochs were used to train the model.
Based on the results, it can be seen that the "bart-base-finetuned-cnn_dailymail" model performs worse than the prior model. The current model appears to have problems collecting unigram and bigram sequences, preserving coherence in longer sequences, and producing high-quality summaries generally, as evidenced by the lower scores across all evaluation metrics. It may be worthwhile to think about modifying the hyperparameters to enhance the model's performance, such as testing various learning rates, batch sizes, or optimization strategies. Additionally, more training epochs or a different method of fine-tuning might produce superior outcomes. Finding the ideal setup for increasing the model's performance on the CNN/DailyMail dataset requires repeated testing of various configurations.

5.3 T5-small

The "small" variant of T5, often known as T5-small, is a scaled-down version of the original model that retains much of the original's capability while being significantly computationally efficient. Compared to T5-base and T5-large, it has fewer parameters. It is suitable in situations where either time or computing power are limited. T5-small is a great choice for many NLP tasks, including text creation and summarizing issues: produce some output text given some input text. T5-small is pre-trained on a sizable corpus of text data and may be tailored for a variety of specific applications, just like the more thorough T5 models.
Rouge-1, Rouge-2, Rouge-L, and Rouge-Lsum are the evaluation measures used to gauge the model's effectiveness. Here are the results:

	Rouge-1	Rouge-2	Rouge-L	Rouge-Lsum
T5-small	0.354386	0.151331	0.243898	0.303233

Table 5.3: Results from the T5-small

This score is somewhat higher than BART but lower than PEGASUS in comparison to the previous model, indicating that the present model has a modest unigram recall and precision.

5.4 Results Comparison

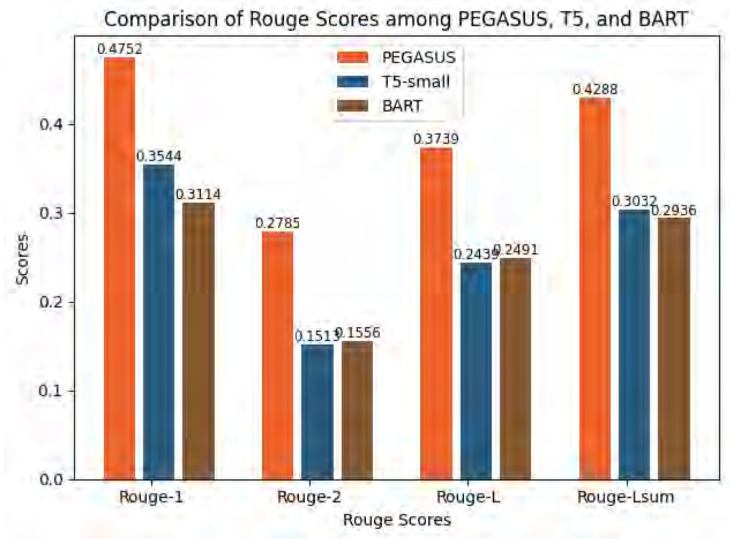


Figure 5.1: Rouge scores of the pre-trained models

The result shows PEGASUS performing the best, T5 performing well and BART performing satisfactorily. Finally, after comparing these models' generated summaries with the human written ones, it clearly shows that these models are quite capable of doing their tasks.

Chapter 6

Fine-tuning Flan-T5 on XLSum

6.1 Introduction

In the NLP discipline, it is common practice to modify previously trained models for certain tasks. This chapter describes an experiment where 20% of the XLsum English dataset was used to fine-tune the Flan-T5 model. A pre-trained mT5 baseline was used to compare its performance to that. Fascinatingly, Flan-T5 did not outperform mT5 in ROUGE scores for the assigned job despite the specific fine-tuning.

6.2 Dataset Selection

The XLsum English dataset was chosen for its comprehensive and balanced corpus specifically designed for text summarization tasks. For this experiment, 20% of the dataset was employed to train the Flan-T5 model.

6.3 Model Selection

Flan-T5 was selected for its potential in state-of-the-art text summarization, Flan-T5 was subjected to fine-tuning in order to adapt its capabilities to the characteristics of the XLsum English dataset

6.4 Hyperparameter

- Learning rate: 2e-05
- Training batch-size: 8
- Evaluation batch-size: 8
- Seed: 42
- Optimizer: Adam with betas=(0.9,0.999) and epsilon=1e-08
- LR scheduler type: linear
- Number of epochs: 5

6.5 Results

The results were evaluated using ROUGE metrics. The following data represents the recorded scores.:

	Rouge-1	Rouge-2	Rouge-L	Rouge-LSUM
mT5_multilingual_XLSum	**37.601**	**15.153**	**29.88**	-
flan-t5-xlsum	35.4133	14.4516	28.2197	28.2456

Table 6.1: Rouge scores from the pre-trained and fine-tuned models

While the FLAN-T5 model did not manage to surpass the baseline MT5 model in terms of performance, it is important to note that this does not represent the end of the line for improving text summarization models. As part of our ongoing efforts, we plan to fine-tune the PEGASUS model on the same 20% subset of the XLsum English dataset to see if it offers any improvements over the baseline and fine-tuned FLAN-T5 models. PEGASUS is another state-of-the-art model with a different architecture, and there is reasonable potential for it to yield better results on the text summarization task.

Chapter 7

Fine-tuning PEGASUS on XLSum

The PEGASUS transformer model, which was initially created by Google AI Research for abstractive summarization, is the subject of our empirical study in this chapter. Our main focus is on the improvements made possible by perfecting this model on the XLSum dataset. Through our efforts, we were able to create a model that, when tested on the same dataset, performs better than the mT5 baseline. Particularly, our improved PEGASUS model demonstrated significantly enhanced performance in Rouge-1, Rouge-2, and Rouge-L measures.

7.1 PEGASUS Transformer Model: A Recap

The PEGASUS transformer model is renowned for using self-supervised objectives to pretrain from a sizable corpus of text, consequently displaying appreciable advancements in the field of abstractive text summarization. The primary way that this model differs from others is by pre-training it with an aim that prefers significant sentences over randomly chosen ones, namely the sentences that are used in human-generated summaries.

7.2 Fine-Tuning the PEGASUS Transformer Model

A large percentage of our study and investigation is focused on fine-tuning. For this procedure, we used 20% of the English training corpus from XLSum and the following hyperparameters:

- Learning rate: 2e-05
- Training batch-size: 8
- Evaluation batch-size: 8
- Seed: 42
- Optimizer: Adam with betas=(0.9,0.999) and epsilon=1e-08
- LR scheduler type: linear
- Number of epochs: 4

It is crucial to remember that these hyperparameters were chosen following a number of tests and parameter sweeps to enhance performance. These decisions aided in our model's convergence and led to the production of better results.

7.3 Comparative Evaluation: PEGASUS Vs. mT5

As a baseline model for comparison, we used the mT5 model, an unsupervised multilingual variant of T5. The mT5 model was also improved using the XLSum dataset, and its Rouge scores for Rouge-1, Rouge-2, and Rouge-L were 37.60, 15.15, and 29.88, respectively [23]

	Rouge-1	Rouge-2	Rouge-L	Rouge-LSUM
mT5_multilingual_XLSum	37.601	15.153	29.88	-
pegasus_xlsum	**39.121**	**17.467**	**30.894**	30.892

Table 7.1: Rouge scores from the mT5 and fine-tuned PEGASUS

These results show a considerable performance gain when compared to our improved PEGASUS model. Our model achieved Rouge scores for Rouge-1, Rouge-2, and Rouge-L of 39.12, 17.46, and 30.89, respectively. The improvement in these measures illustrates the efficacy of our process of fine-tuning and confirms the potency of the PEGASUS transformer model when appropriately suited for certain workloads.

7.4 In-depth Analysis of Model Performance

The observed rise in Rouge scores signifies a notable advancement in the field of abstractive text summarization. The Rouge-1 measure is employed for evaluating the extent of unigram overlap between the system-generated summary and the reference summary. A higher Rouge-1 score signifies a greater degree of lexical similarity between the reference summary and the generated summary.
The Rouge-2 methodology is employed to compute the level of bigram overlap between the system-generated summaries and the reference summaries. This criterion emphasizes the coherence and fluidity of the resulting summary. The updated PEGASUS model demonstrates superior performance in generating summaries that exhibit coherency and logical organization, as shown by a higher Rouge-2 score.
In contrast, Rouge-L takes into account the sequential order of the text and examines the longest common subsequence shared by the system-generated summaries and the reference summaries. This enhancement underscores the model's capacity to generate summaries that nearly replicate the semantic organization of the source text, while also maintaining its logical coherence.

7.5 Model Dissemination

Given the significant improvements made possible by our method of fine-tuning, the improved PEGASUS transformer model was subsequently released on the Hugging

Face Model Hub. The Hugging Face platform was chosen because it is well-liked in the research community and has an approachable infrastructure that makes it simple to share and include models.

With this release, we want to provide other scholars and professionals the chance to use this improved model, put it to use in their own situations, and further advance the art of abstractive summarization.

Chapter 8

Interpretation of fine-tuned PEGASUS with SHAP

Understanding why a model generates a specific prediction is as crucial as the prediction itself in the increasingly complicated world of machine learning. This understanding is crucial for establishing trust in any systems, validating models, and making informed decisions. SHAP (SHapley Additive exPlanations), an interpreter library designed to bring transparency and interpretability to machine learning models.

8.1 Shapley Values and Their Mathematical Formulation

SHAP hinges on the concept of Shapley values, a mathematical framework borrowed from cooperative game theory. They are used to fairly allocate a specific value among multiple contributing players in a game. In the context of machine learning, these "players" are the features used by the model, and the "value" being allocated is the contribution of each feature towards making a particular prediction. The Shapley value ϕ_i of a feature i in a model with N features is computed as follows:

$$\phi_i = \sum_{S \subseteq N \setminus \{i\}} \frac{|S|! \cdot (|N| - |S| - 1)!}{|N|!} \left(f(S \cup \{i\}) - f(S)\right)$$

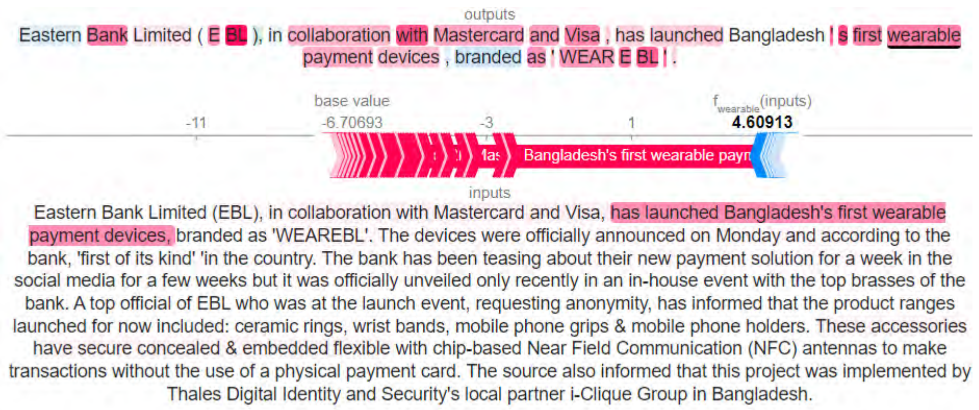
Here, $f(S)$ is the prediction of the model for a given subset of features S, and $|S|$ denotes the cardinality of S. N is the set of all features. The term $f(S \cup \{i\}) - f(S)$ calculates the marginal contribution of feature i when added to a set S. The Shapley value ϕ_i is calculated by taking an average over all possible combinations of features, evaluating how much feature i contributes to every possible subset S [7].

8.2 SHAP for summarization models

The pre-trained or fine-tuned abstractive summarization model is first prepared for analysis. This often entails tokenizing the input text and structuring it in a format compatible with the specific transformer model. After initialization, the SHAP library uses the summarization model to compute the Shapley values. These values aim to quantify the contribution of each feature—in this context, each input token or group of tokens—to the output. Once the text sample is tokenized and fed into the model along with its corresponding SHAP explainer, Shapley values for each token are generated. These values serve as indicators of the level of influence each token had in shaping the final summary output.

8.3 Predicting pegasus_xlsum output

In the context of abstractive summarization models, SHAP values help in understanding the contributions of each feature (word or token in this case) towards making a particular prediction. In many SHAP visualizations, blue and red colors are used to represent the impact of a particular feature on the model's output. In general, blue typically indicates a feature that decreases the prediction value. On the other hand, red signifies a feature that increases the prediction value. In the context of abstractive summarization, a red token may be one that the model firmly feels is necessary for the summary to include in order for it to be coherent or useful, whereas a blue token may be one that the model feels is less significant. Once more, a positive SHAP value for a word or token denotes that the possibility of the term appearing in the summary is increased by its presence. A negative SHAP score, on the other hand, decreases the likelihood of inclusion in the summary. The model is affected in a "negative" way by it.



Eastern Bank Limited (E BL), in collaboration with Mastercard and Visa , has launched Bangladesh ' s first wearable payment devices , branded as ' WEAR E BL ' .

Eastern Bank Limited (EBL), in collaboration with Mastercard and Visa, has launched Bangladesh's first wearable payment devices, branded as 'WEAREBL'. The devices were officially announced on Monday and according to the bank, 'first of its kind' 'in the country. The bank has been teasing about their new payment solution for a week in the social media for a few weeks but it was officially unveiled only recently in an in-house event with the top brasses of the bank. A top official of EBL who was at the launch event, requesting anonymity, has informed that the product ranges launched for now included: ceramic rings, wrist bands, mobile phone grips & mobile phone holders. These accessories have secure concealed & embedded flexible with chip-based Near Field Communication (NFC) antennas to make transactions without the use of a physical payment card. The source also informed that this project was implemented by Thales Digital Identity and Security's local partner i-Clique Group in Bangladesh.

Figure 8.1: pegasus_xlsum output analysis using SHAP

From the above pictures, it is seen that tokens like "mastercard" (3.2417), "wearable" (4.60913), and "payment" (2.92806) have positive SHAP values, meaning these words strongly contribute to the model's decision to include them in the summary. They may be considered key aspects of the original text that the model believes are necessary for a good, informative summary. Tokens like "field" (-0.329) and "accessories" (-0.114) have negative SHAP values, implying that these words are less critical to the model's generation of the summary. The negative values suggest that they might be omitted or given less emphasis to make the summary more coherent, concise, or informative. Again, despite being negative SHAP value words, words like "branded" (-1.8354) and "eastern" (-1.3011) are added to the summary. It is because these words hold contextual importance. They may have negative SHAP values, but their inclusion could be contextually relevant to other parts of the summary. Again, the decision to include a word is influenced by the combined effects of all the words in the input. So, even if a word has a negative SHAP value, the total contribution of all the features may still result in that word being included. Moreover, sometimes models include words for readability or coherence, even if those words don't add significant information as quantified by SHAP values.

Understanding the SHAP values can help in diagnosing the model's behavior, allowing for possible refinements. Overall, the use of SHAP values in interpreting abstractive summarization models adds an essential layer of transparency for understanding the decision-making process of these advanced systems.

Chapter 9

Conclusion

In this work, we experimented with various transformer architectures to fine-tune abstractive text summarization models on the csebuetnlp/xlsum/eng dataset. Initially, we fine-tuned the pre-trained google/flan-t5-base model, resulting in the flan-t5-xlsum model. Despite the effort, the ROUGE scores did not meet expectations. After finetuning, the flan-t5-xlsum model achieved ROUGE-1, ROUGE-2, ROUGE-L, and ROUGE-LSUM scores of 35.4133, 14.4516, 28.2197, and 28.2456, respectively. In contrast, the mT5_multilingual_XLSum model by BUET presents ROUGE scores of 37.601 for ROUGE-1, 15.153 for ROUGE-2, and 29.88 for ROUGE-L. However, significant improvements were observed when we used the pre-trained google/pegasus-cnn_dailymail model for fine-tuning. We named our developed model pegasus_xlsum, which outperformed the csebuetnlp/mT5_multilingual_XLSum model in all ROUGE metrics, achieving ROUGE-1, ROUGE-2, ROUGE-L, and ROUGE-LSUM scores of 39.121, 17.467, 30.894, and 30.892, respectively. To quantify the improvement, we calculated the percentage change in each ROUGE score for the pegasus_xlsum model as compared to the mT5 model. We could see a 4.04% improvement in the ROUGE-1 score and a 3.39% improvement in the ROUGE-L score. Our model exhibited a notable improvement, showing a 15.25% increase in the ROUGE-2 score, which indicates a substantial enhancement in capturing more complex sentence structures and content relationships. The table below shows the results altogether:

	Rouge-1	Rouge-2	Rouge-L	Rouge-LSUM
mT5_multilingual_XLSum	37.601	15.153	29.88	-
pegasus_xlsum	**39.121**	**17.467**	**30.894**	30.892
flan-t5-xlsum	35.4133	14.4516	28.2197	28.2456

Table 9.1: Rouge scores from the pre-trained and fine-tuned models

These results indicate that the pegasus_xlsum model demonstrates a noticeable performance advantage over the csebuetnlp/mT5_multilingual_XLSum model. Thanks to the architectural advantages of the Pegasus model, originally designed for extractive and abstractive summarization tasks, it appears to align well with the intricacies of the csebuetnlp/xlsum/eng dataset. The advancements in our pegasus_xlsum model make it a viable choice for abstractive summarization tasks.

Bibliography

[1] Y. Gong and X. Liu, "Generic text summarization using relevance measure and latent semantic analysis," in *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, 2001, pp. 19–25.

[2] R. Mihalcea and P. Tarau, "Textrank: Bringing order into text," in *Proceedings of the 2004 conference on empirical methods in natural language processing*, 2004, pp. 404–411.

[3] N. Munot and S. S. Govilkar, "Comparative study of text summarization methods," *International Journal of Computer Applications*, vol. 102, no. 12, 2014.

[4] Y. Sankarasubramaniam, K. Ramanathan, and S. Ghosh, "Text summarization using wikipedia," *Information Processing & Management*, vol. 50, no. 3, pp. 443–461, 2014.

[5] R. Nallapati, B. Zhou, C. Gulcehre, B. Xiang, *et al.*, "Abstractive text summarization using sequence-to-sequence rnns and beyond," *arXiv preprint arXiv:1602.06023*, 2016.

[6] A. Herbelot and M. Baroni, "High-risk learning: Acquiring new word vectors from tiny data," in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, Copenhagen, Denmark: Association for Computational Linguistics, Sep. 2017, pp. 304–309. DOI: 10.18653/v1/D17-1030. [Online]. Available: https://aclanthology.org/D17-1030.

[7] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," *Advances in neural information processing systems*, vol. 30, 2017.

[8] A. See, P. Liu, and C. Manning, "Get to the point: Summarization with pointer-generator networks," in *Association for Computational Linguistics*, 2017. [Online]. Available: https://arxiv.org/abs/1704.04368.

[9] A. See, P. J. Liu, and C. D. Manning, "Get to the point: Summarization with pointer-generator networks," *arXiv preprint arXiv:1704.04368*, 2017.

[10] S. Narayan, S. B. Cohen, and M. Lapata, "Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, Brussels, Belgium: Association for Computational Linguistics, Oct. 2018, pp. 1797–1807. DOI: 10.18653/v1/D18-1206. [Online]. Available: https://aclanthology.org/D18-1206.

[11] S. Narayan, S. B. Cohen, and M. Lapata, “Don’t give me the details, just the summary! Topic-aware convolutional neural networks for extreme summarization,” in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, Brussels, Belgium, 2018.

[12] F. Boutkan, J. Ranzijn, D. Rau, and E. van der Wel, “Point-less: More abstractive summarization with pointer-generator networks,” *arXiv preprint arXiv:1905.01975*, 2019.

[13] M. Lewis, Y. Liu, N. Goyal, *et al.*, “Bart: Denoising sequence-to-sequence pretraining for natural language generation, translation, and comprehension,” *arXiv preprint arXiv:1910.13461*, 2019.

[14] C. Raffel, N. Shazeer, A. Roberts, *et al.*, “Exploring the limits of transfer learning with a unified text-to-text transformer,” *arXiv e-prints*, 2019. arXiv: 1910.10683.

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

[16] J. Maynez, S. Narayan, B. Bohnet, and R. McDonald, “On faithfulness and factuality in abstractive summarization,” in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Online: Association for Computational Linguistics, Jul. 2020, pp. 1906–1919. DOI: 10.18653/v1/2020.acl-main.173. [Online]. Available: https://aclanthology.org/2020.acl-main.173.

[17] J. Pilault, R. Li, S. Subramanian, and C. Pal, “On extractive and abstractive neural document summarization with transformer language models,” in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2020, pp. 9308–9319.

[18] C. Raffel, N. Shazeer, A. Roberts, *et al.*, “Exploring the limits of transfer learning with a unified text-to-text transformer,” *J. Mach. Learn. Res.*, vol. 21, no. 1, Jan. 2020, ISSN: 1532-4435.

[19] C. Raffel, N. Shazeer, A. Roberts, *et al.*, “Exploring the limits of transfer learning with a unified text-to-text transformer,” *The Journal of Machine Learning Research*, vol. 21, no. 1, pp. 5485–5551, 2020.

[20] S. Sotudeh, A. Cohan, and N. Goharian, “On generating extended summaries of long documents,” *arXiv preprint arXiv:2012.14136*, 2020.

[21] W. Xiao and G. Carenini, “Systematically exploring redundancy reduction in summarizing long documents,” *arXiv preprint arXiv:2012.00052*, 2020.

[22] J. Zhang, Y. Zhao, M. Saleh, and P. Liu, “Pegasus: Pre-training with extracted gap-sentences for abstractive summarization,” in *International Conference on Machine Learning*, PMLR, 2020, pp. 11 328–11 339.

[23] T. Hasan, A. Bhattacharjee, M. S. Islam, *et al.*, “XL-sum: Large-scale multilingual abstractive summarization for 44 languages,” in *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, Online: Association for Computational Linguistics, Aug. 2021, pp. 4693–4703. DOI: 10.18653/v1/2021.findings-acl.413. [Online]. Available: https://aclanthology.org/2021.findings-acl.413.

[24] F. Nan, R. Nallapati, Z. Wang, *et al.*, "Entity-level factual consistency of abstractive text summarization," *arXiv preprint arXiv:2102.09130*, 2021.

[25] S. Abdel-Salam and A. Rafea, "Performance study on extractive text summarization using bert models," *Information*, vol. 13, no. 2, 2022, ISSN: 2078-2489. DOI: 10.3390/info13020067. [Online]. Available: https://www.mdpi.com/2078-2489/13/2/67.

[26] A. Gupta, D. Chugh, R. Katarya, *et al.*, "Automated news summarization using transformers," in *Sustainable Advanced Computing*, Springer, 2022, pp. 249–259.