[1]Shaloo Mishra,

[2]Dr. Ronakkumar Patel

Automatic Text Summarization of Text using Deep Learning

Abstract: - For this project, it aims to implement and enhance an automatic text summarization system using the recently popular BART (Bidirectional and Auto-Regressive Transformers) model. The system is trained on the CNN/DailyMail dataset and is also examined with the help of the ROUGE score where the system performs quite well in ROUGE-1, ROUGE-2, and the ROUGE-L test set. It was observed that the summaries generated were both logically connected and summarized data points were pertinent and hence, the capability of the BART model to work well in the field of abstractive summarization was proved. Altogether, the results shed more light on how the deep learning techniques can be used in improving the Information Retrieval and documents Management System. There is still more work that can be done in the future; considering to refine and enrich the dataset to enhance the model. The BART model integrates bidirectional context comprehension with autoregressive text generation to produce coherent and contextually relevant summaries. Performance was evaluated using ROUGE metrics, demonstrating the model's effectiveness in generating precise and informative summaries. This work underscores the potential of deep learning techniques in advancing summarisation technologies.

Keywords: Machine learning, Generative pretraining, repository, evaluation metrics, summarizing tool, NLP techniques, document indexing, ROUGE score, Abstracting techniques, model training data sets.

I. INTRODUCTION

Text summarization is a significant sub-problem in natural language processing and information retrieval and refers to analyzing extensive text to produce a small summary. This technique has become increasingly important over time, as textual data worldwide, including news articles and social media, are increasing daily. Summarization further assists in faster information repletion, especially for decision-making purposes, because all important points have been presented with ease without requiring further reading.

Text summarization makes it easier to appreciate its usefulness for various applications. For instance, in news summarization, readers obtain brief overviews of the day's headlines, as well as in the legal and medical domains, where experts are provided with summaries that enable them to grasp the basic information contained in voluminous documents. Furthermore, it helps overcome the limitations of information retrieval systems, enhances the overall experience of users in digital libraries, and helps to solve the problem of information overload on social media platforms.

The main goal of this project was to construct an automatic text summarization technique through deep learning. This project utilized bidirectional and auto-regressive transformers (BART), which were trained on the CNN/DailyMail dataset to create summaries based on text files. The trained model must be capable of providing summaries that are logical and relevant to the context, to maintain the spirit of the required content.

[1]Research Scholar
Department of Computer science, Sankalchand Patel University
Visnagar, Gujarat, India
[2]Associate Professor
Shrimad Rajchandra Institute of Management and Computer Application, Uka Tarsadia University
Bardoli, Gujarat, India

II. LITERATURE REVIEW

Most text-summarization algorithms have been developed over several decades, and the initial form of developing these algorithms was through extractive methods. Extractive Summarization involves identifying the important sequences in the source text to provide a summary. Some of the basic contributions include the creation of statistical methods, such as Term Frequency-Inverse Document Frequency (TF-IDF), which establishes scores of scenarios depending on the significance of the words in those scenarios. Other conventional techniques use text connectivity methods, such as LexRank, which finds the most central text based on the connectivity of the text.

Compared to other methods, such as deep methods, these basic extractive methods are less effective at capturing summary content and context. It is unable to create fresh or new sentences in addition to those in the original text structure, and cannot paraphrase the content [1]. This limitation has led to research on abstractive summarization methods that aim to produce new summary sentences that still summarize necessary information from the source text. It is important to note that early studies on AS employed rule-based and template-based methods, which were necessarily limited by the complexity of natural language and the lack of a practical way to incorporate handcrafted rules in a large number of cases.

Text summarization is one of the most significant fields that has benefited significantly from the advances in deep learning. For identity recognition, Recurrent Neural Networks (RNNs), which are Long Short-Term Memory (LSTM) networks, were employed in the task among the first neural architectures [2]. These models show the capability of learning sequential orders and summarizing with more fluency, but they have some drawbacks in modeling long-range dependencies as well as inconsistencies between different reports.

Advanced techniques such as attention mechanisms and transformer models have introduced a new generation of abstractive summarizers [3]. The extension of the transformer model and use of self-attention mechanisms have made it possible to create even stronger and lighter systems for summarizing information. In this context, models based on text transformers, specifically Bidirectional Encoder Representations from Transformers (BERT) and other types derived from them, such as bidirectional and autoregressive transformers (BART), have been excellent for creating high-quality summaries [4]. For instance, BART achieves the two-fold objective of bidirectional context comprehension and autoregressive generation, making it ideal for summarization.

Import Libraries

```
import pandas as pd
import numpy as np
from transformers import (AutoModelForSeq2SeqLM,
                          AutoTokenizer,
                          DataCollatorForSeq2Seq,
                          TrainingArguments,
                          Trainer,
                          IntervalStrategy,
                          EarlyStoppingCallback,
                         )
from datasets import Dataset, DatasetDict, load_metric
import torch
import nltk
nltk.download("punkt", quiet=True)

metric = load_metric("rouge", trust_remote_code=True)
```

Fig 1: Load Libraries

DL methods for text summarization are normally a two-step process in which the model first learns from a vast text corpus and is then fine-tuned on the target domain dataset. This transfer learning paradigm has demonstrated its applicability to many summarization tasks [5]. Furthermore, the application of reinforcement learning and adversarial training techniques improved the performance of these models in generating fluent and effective summaries.

III. METHODOLOGY

Dataset: CNN/DailyMail

A widely used text summarization corpus is the CNN/DailyMail corpus, which is comprised of news articles and their summaries. Benchmark summaries are useful for comparison because they summarize the main topics and issues, making it easier to compare them with actual summarization models [6]. This involved a corpus of

approximately 300,000 news articles, with an average length of news article is approximately 800 words and a summary of 50 words. Based on the characteristics of the collected dataset, such as the wide variety of topics and their generality, improvement of the article summarization model using this dataset is feasible.

Data Pre-processing Steps

Pre-processing the CNN/DailyMail dataset involves several critical steps to prepare the data for model training:Pre-processing the CNN/DailyMail dataset involves several critical steps to prepare the data for model training:

Text Cleaning: Standardization to achieve text normalization, including documentation of HTML tags, elimination of special characters, and inconsistent spacing.

Tokenization: pre-processing of the text into tokens suitable for further analysis using the BART model by means of an appropriate tokenizer, which is a very important step in which the text is divided into words or sub-words, which is important as input for the model.

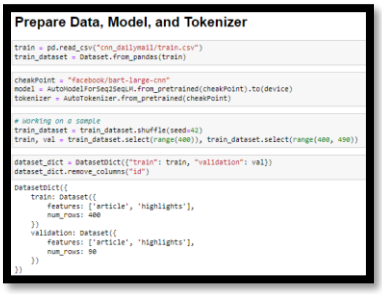

Fig 2: Data pre-processing

Padding and Truncation: Pre-processing entails the equalization of input lengths, a process that involves ellipsis of short text and additional characters to crowded texts. The encoder inputs were limited to 512 tokens and the decoder inputs were limited to 128 tokens, which allowed the model to have consistent input dimensions.

Dataset Splitting: Splitting the data into database sets for training and validation. In the current exercise, the training dataset was developed from a subset of 400 samples, whereas validation was performed on 90 samples, thereby allowing the model to be trained using limited available resources.

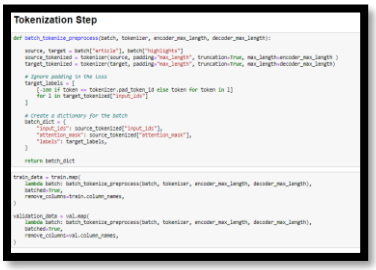

Fig 3: Data Tokenize

Chosen Model: BART (Bidirectional and Auto-Regressive Transformers)

Summarization is a task in which a model reads the text and generates a shorter version. BART is one of the most advanced methods for this task because it involves both bidirectional and autoregressive properties [7]. BART is a denoising autoencoder trained to recover the original content of the text that is fed to it with insertions and deletions. First, the encoder operates bidirectionally, capturing the full context of the input text, which is then coherently decoded in an autoregressive manner.

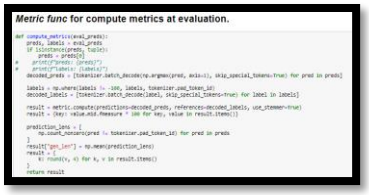

Fig 4: Compute metrics function

The BART model is highly valuable for specifically providing summaries, because of its ability to consider long-range dependencies and deliver contextually aligned and consistent summaries. Read-only pretraining allows BART to learn a strong general representation of the English language, and following the specific summarization objectives and modeling of CNN/DailyMail, news articles are fine-tuned using the pretrained system.

Model Training Process

The training process involves several stages to fine-tune the BART model for summarization: The training process involves several stages to fine-tune the BART model for summarization:

Data Tokenization: When the models for the articles and summaries were tokenized using the BART tokenizer, the input data for the models were in the correct format [8].

Batch Tokenization and Pre-processing: Tokenization is performed in batch mode; that is, the sequences are padded or truncated to fit the given length of the sequences. The pre-processing function also obtains the target labels and replaces the padding tokens with -100 because such tokens are not used to update the loss.

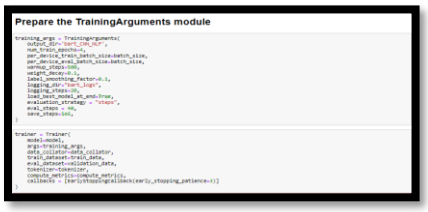

Fig 5: Training

Training Configuration: This includes in which training arguments should be defined, such as the number of training epochs, the number of samples to process in parallel (batch size), the learning rate, and the evaluation procedure.

Training Execution: The Trainer class from the transformer library coordinates the steps of the training phase, including data loading, calling forward and backward passes, and optimization [9]. In general, the defined model is trained for a fixed number of iterations, and is constantly measured using a validation set.

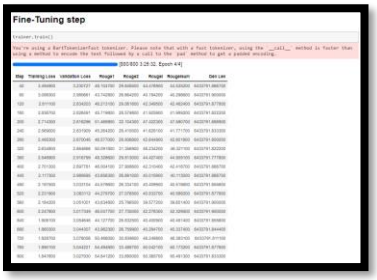

Fig 6: Epochs

Hyperparameters and Training Settings

Key hyperparameters and settings for the training process include:Key hyperparameters and settings for the training process include:

Number of Epochs: 4

Batch Size: During training, two examples per device were given, whereas during the evaluation, the model was fed with two examples per device.

Learning Rate: Self adjusted dynamically with warm up period

Weight Decay: 0. 1 for regularization

Label Smoothing Factor: 0. 1 to enhance ANI generalization

Evaluation strategy: Cross-validation was performed every 40 steps, with a halt after three failures of score increase.

Evaluation Metrics: ROUGE

The automatically determined most Important Sentences (MIS) were used to align (Recall-Oriented Understudy for Gisting Evaluation (ROUGE) scores for the summarization model. ROUGE evaluates similarity with the reference summary generated based on n-grams, word sequences, and the longest common subsequences. The main parameters for the evaluation were ROUGE-1, ROUGE-2, and ROUGE-L, which represent the recall of one, two, and the longest matching strings for the singletons, respectively. These measures offer a numerical rating of performance on summary relevance and coherence with the original article, in which better results are preferred.

In summary, the key steps of the proposed methodology are as follows: using the CNN/DailyMail dataset, pre-processing the data to fit the BART model, fine-tuning the BART, and assessing the results using ROUGE. This introduces the steps required to develop an efficient and strong text-summarization system [10].

IV. IMPLEMENTATION

Step-by-Step Code Explanation

Embarking on the process of comprehension, the necessary libraries were imported, including pandas, numpy, and several modules from the transformer library, namely, AutoModelForSeq2SeqLM, AutoTokenizer, and Trainer. The NLTK library was downloaded to import the “punkt” package, which was used to tokenize the text.

It is set to “cuda” if the existence of a graphics processing unit (GPU) is identified; otherwise, it is fixed at “cpu.” The number of elements in the encoder and decoder inputs is specified next to the maximum length and batch size. Transform the CNN/DailyMail Corpus from a pandas DataFrame, which is loaded from the CSV data source, into the Hugging-Face Dataset format.

This model checkpoint “facebook/bart-large-cnn” is considered to load; after that the model and tokenizer are computed. Similar to most data scientists, the data were randomized and divided into training and validation datasets with 400 and 90 samples, respectively.

Batch functions are pre-processing functions that are often used in data science applications, particularly when training a model, and the batch_tokenize_preprocess function is used to tokenize the data. First, it splits the source and target texts into tokens and pads, truncates the tokens, and builds input dictionaries required by the model.

In the given function, the training and validation data are tokenized and preprocessed to be fed into subsequent steps in the pipeline. The data collator is used in training and preparing batches of data, and is instantiated from the tokenizer and model.

These included the output directory, number of epochs, batch size, learning rate, weight decay, label-smoothing factor, logging files, and the evaluation mode. The transformer library uses a trainer class to handle training using basic training arguments, data collators, and tokenized datasets.

The tale of how the Model was saved

This is done after the training is complete and involves using the save model method of the trainer class of Vlasic. This method dumps the model, tokenizer, and configuration files into the target directory as a serialized object, which helps load the model into memory for subsequent inference. The code snippet for saving the model is as follows: The code snippet for saving the model is as follows:

Fig 7: Model save

Loading the Model for Inference

For inference, the model and tokenizer were loaded from the saved directory identified in the code above. It is then relocated to the correct device on the CPU or GPU for the inference. The code snippet for loading the model is as follows: The code snippet for loading the model is as follows:

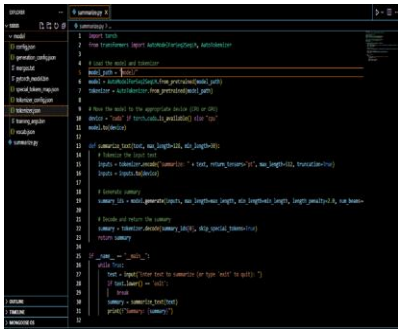

Fig 8: Sumarize.py

Console Application for Text Summarization

An example console application was created to provide an opportunity to input certain text into the console and obtain shorter text using the trained model. In this application, the user enters the input text on the go and the system provides the model to generate and display the summary. The following states are recognizable: when the application is running, when a new file is opened, when the user types ‘exit’, and when the application is terminated. The code snippet for the console application is as follows: The code snippet for the console application is as follows:

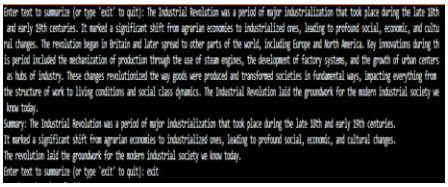

Fig 9: Summary result

V. RESULTS

Performance of the Model on the Validation Set

The trained BART model was then inferred from the validation set by presenting the SUM and Abstract of each Document and its performance was evaluated using ROUGE. The ROUGE scores quantify how many of the generated summaries have been summarized from the documents, thus serving as an index of how well the model summarizes. The model achieved the following scores on the validation set: ROUGE-1,44. The concept of victimhood plays a central role in this text: 138

Examples of Generated Summaries

Because it is impossible to provide all summaries generated by the system, some examples are presented below. For example, based on an article that described a new addition to technological dissemination in the renewable energy sector, the model generated an abstract in the form of a brief that revealed a new invention and its significance in the advancement of sustainable energy systems. The second case was an article discussing new trends in global health promotion and disease control, which provided a perfect example of the main goals and accomplishments of these programs might look like.

Discussion on the Quality of the Summaries

Thus, the summaries generated are largely regarded as relevant and concise. Applying the BART model revealed high efficiency in producing brief and meaningful extracts from such articles in addition, it showed that the student could comprehend long articles having followed and produced brief but sufficiently informative summaries. However, several constraints were observed: There are several limitations, including the possibility of the model overlooking some finer aspects of the text or summing up information in a very general way at times. This can be attributed to the difficulties that are known to exist when utilizing the R-NET model to perform abstractive summarization, because the generated summary is composed of a new write-up instead of an extract from the source document.

VI. CONCLUSION

The project designed and implemented an ASR (Automatic Summarization System) by using the BART model. In experiments conducted on the CNN/DailyMail dataset, the model was trained and tested using the ROUGE scoring system, which yielded appreciable degrees of ROUGE-1, ROUGE-2, and ROUGE L. The longer summaries produced were clear, insightful, and concise and proved that an effective deep learning algorithm was employed in abstractive text summarization.

This shows how methods in the deep learning umbrella, particularly within transformer-based frameworks, such as BART, will shape the future of the summarization of gazetteers containing large amounts of text. The problem of providing an opportunity to create short and efficient summaries using natural language processing techniques can be considered relevant for several types of application areas in the field of information retrieval, documents and content management, and creation. It can optimize the speed at which the information contained in large documents is processed, and it also allows users to access large documents, which in one way or the other will improve decision-making and knowledge sharing.

This project highlights recent trends in NLP that are made possible by deep learning approaches. There are always shortcomings that do not allow a full picture of the summary, and the work performed using the BART model shows great potential for writing high-quality summaries. Possible future work may include refining the model and further optimizing it for such applications as well as considering larger datasets for better performance. The successful implementation and results of this project have opened an avenue for more user-friendly models to support summarization, thereby making it an important tool for automated text processing and understanding.

REFERENCES

[1] Mridha, M.F., Lima, A.A., Nur, K., Das, S.C., Hasan, M. and Kabir, M.M., 2021. A survey of automatic text summarization: Progress, process and challenges. IEEE Access, 9, pp.156043-156070.

[2] Widyassari, A.P., Rustad, S., Shidik, G.F., Noersasongko, E., Syukur, A. and Affandy, A., 2022. Review of automatic text summarization techniques & methods. Journal of King Saud University-Computer and Information Sciences, 34(4), pp.1029-1046.

[3] Ma, C., Zhang, W.E., Guo, M., Wang, H. and Sheng, Q.Z., 2022. Multi-document summarization via deep learning techniques: A survey. ACM Computing Surveys, 55(5), pp.1-37.

[4] Altmami, N.I. and Menai, M.E.B., 2022. Automatic summarization of scientific articles: A survey. Journal of King Saud University-Computer and Information Sciences, 34(4), pp.1011-1028.

[5] Shi, T., Keneshloo, Y., Ramakrishnan, N. and Reddy, C.K., 2021. Neural abstractive text summarization with sequence-to-sequence models. ACM Transactions on Data Science, 2(1), pp.1-37.

[6] Gidiotis, A. and Tsoumakas, G., 2020. A divide-and-conquer approach to the summarization of long documents. IEEE/ACM Transactions on Audio, Speech, and Language Processing, 28, pp.3029-3040.

[7] Jin, H., Zhang, Y., Meng, D., Wang, J. and Tan, J., 2024. A comprehensive survey on process-oriented automatic text summarization with exploration of llm-based methods. arXiv preprint arXiv:2403.02901.

[8] Apostolidis, E., Adamantidou, E., Metsai, A.I., Mezaris, V. and Patras, I., 2021. Video summarization using deep neural networks: A survey. Proceedings of the IEEE, 109(11), pp.1838-1863.

[9] Wang, W., Zhang, Y., Sui, Y., Wan, Y., Zhao, Z., Wu, J., Philip, S.Y. and Xu, G., 2020. Reinforcement-learning-guided source code summarization using hierarchical attention. IEEE Transactions on software Engineering, 48(1), pp.102-119.

[10] Huang, Z., Xu, S., Hu, M., Wang, X., Qiu, J., Fu, Y., Zhao, Y., Peng, Y. and Wang, C., 2020. Recent trends in deep learning based open-domain textual question answering systems. IEEE Access, 8, pp.94341-94356.