

# **ABSTRACTIVE TEXT SUMMARIZATION**

---

## **A MINI PROJECT REPORT**

**18CSC305J – ARTIFICIAL INTELLIGENCE**

**(2018 Regulation)**

**III Year/VI Semester**

*Submitted by*

**SAYAK DAS (RA2011026010101)**

**ROOPAL SOOD (RA2011026010103)**

**KESHAV HANDA (RA2011026010088)**

*Under the guidance of*

**Dr. M.S. ABIRAMI**

**Assistant Professor, Department of Computer Science and Engineering**

*In Partial fulfilment for the award of the degree of*

**BACHERLOR OF TECHNOLOGY**

in

**COMPUTER SCIENCE & ENGINEERING**

of

**FACULTY OF ENGINEERING AND TECHNOLOGY**



**SRM INSTITUTE OF SCIENCE AND TECHNOLOGY**

**Kattankulathur, Chengalpattu District**


**MAY 2023**

# SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Under Section 3 of UGC Act,1956)

## BONAFIDE CERTIFICATE

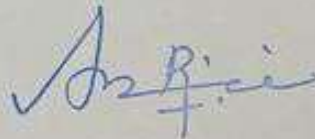
Certified that Mini project report titled "ABSTRACTIVE TEXT SUMMARIZATION" is the bonafide work of SAYAK DAS (RA2011026010101), ROOPAL SOOD (RA2011026010103) & KESHAV HANDA (RA2011026010088) who carried out the minor project under my supervision. Certified further, that to the best of my knowledge, the work reported herein does not form any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

 9/5/23

SIGNATURE

**Dr. M.S. ABIRAMI**  
GUIDE

Assistant Professor  
Department of Computing  
Technologies



SIGNATURE

**Dr. R. ANNIE UTHRA**  
HEAD OF THE DEPARTMENT

Professor & Head  
Department of Computational  
Intelligence

## TABLE OF CONTENTS

<b>S. No</b>	<b>Contents</b>	<b>Page No</b>
<b>1</b>	Abstract	4
<b>2</b>	Problem Statement	5
<b>3</b>	Objective	6
<b>4</b>	Proposed Methodology	7
<b>5</b>	Architecture	8
<b>6</b>	Code & Explanation	9-12
<b>7</b>	Result	13
<b>8</b>	Conclusion	14
<b>9</b>	Reference	15

## **ABSTRACT**

Abstractive text summarization is a Natural Language Processing (NLP) task that involves generating a summary of a given text that captures the most important and relevant information. Unlike extractive summarization, which selects and rephrases key sentences from the original text, abstractive summarization aims to generate new sentences that convey the meaning of the original text.

Abstractive summarization can be applied in various contexts, including news articles, scientific papers, legal documents, and social media posts. It requires advanced NLP techniques, such as natural language generation, machine learning, and deep learning, to generate coherent and informative summaries.

## **PROBLEM STATEMENT**

The increasing amount of textual data available on the internet has created a need for automatic text summarization systems. While extractive summarization methods exist, they often lack coherence and do not capture the main ideas of the original text. Abstractive summarization methods aim to generate more coherent and informative summaries by rephrasing and paraphrasing the original text. The goal of this project is to develop an abstractive text summarization system using NLP techniques that can generate high-quality summaries of input text.

## OBJECTIVES

The main objective of abstractive text summarization using NLP is to generate a concise and coherent summary of a given text, while preserving its most important information and meaning. This is achieved by using natural language processing techniques to identify and extract the key concepts and ideas from the text, and then rephrasing and condensing them in a way that captures their essence.

More specifically, the objectives of abstractive text summarization using NLP can include:

1. **Comprehensiveness:** The summary should cover all the main points and ideas in the original text, while excluding irrelevant or redundant information.
2. **Coherence:** The summary should be well-structured and easy to understand, with clear connections between different ideas.
3. **Conciseness:** The summary should be short and to the point, without sacrificing the essential meaning of the original text.
4. **Fluency:** The summary should be written in natural and fluent language, with correct grammar and syntax.

Overall, the goal of abstractive text summarization using NLP is to provide a useful and effective tool for quickly summarizing large amounts of text.

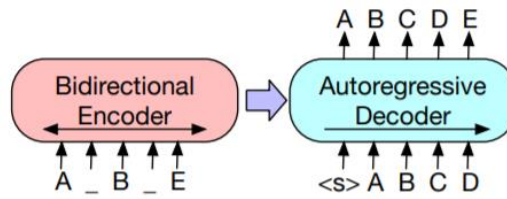
## PROPOSED METHODOLOGY

BART (Bidirectional and Auto-Regressive Transformers) is a state-of-the-art sequence-to-sequence model that has been successfully used for abstractive text summarization.

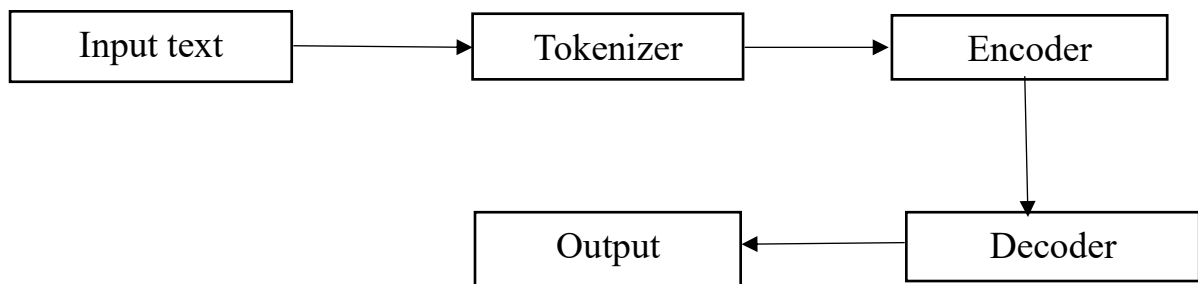
- The BART model is based on a transformer architecture, which uses self-attention mechanisms to capture the dependencies between the input and output sequences. The model consists of two components: an encoder and a decoder. The encoder reads the input text and produces a set of hidden representations, while the decoder uses these representations to generate the summary.
- BART uses a pretraining method that combines denoising autoencoding and sequence-to-sequence pretraining objectives. During pretraining, the model is trained to reconstruct a corrupted version of the input text, and then to generate a target sequence given a source sequence. This approach has been shown to improve the quality of the model's representations and enable it to better capture long-term dependencies in the input text.
- During fine-tuning for abstractive text summarization, the BART model is trained to generate a summary that captures the key information in the input text. The model is trained to minimize a loss function that measures the similarity between the generated summary and the reference summary.

Overall, BART has been shown to achieve state-of-the-art results on several benchmark datasets for abstractive text summarization, demonstrating the effectiveness of the model's pretraining method and transformer architecture.

## ARCHITECTURE



(c) BART: Inputs to the encoder need not be aligned with decoder outputs, allowing arbitrary noise transformations. Here, a document has been corrupted by replacing spans of text with mask symbols. The corrupted document (left) is encoded with a bidirectional model, and then the likelihood of the original document (right) is calculated with an autoregressive decoder. For fine-tuning, an uncorrupted document is input to both the encoder and decoder, and we use representations from the final hidden state of the decoder.





## Import Packages

NLTK (Natural Language Toolkit) is the go-to API for NLP (Natural Language Processing) with Python.

It is a powerful tool to preprocess text data for further analysis.

Re is a built-in python package which is used to work with regular expression.

The Punkt tokenizer is a pre-trained sentence tokenizer, which can be used to split text into individual sentences.

The stopwords corpus contains a list of common words that are often removed from text during preprocessing, as they do not add much meaning to the text.

```
#NLTK (Natural Language Toolkit) is the go-to API for NLP (Natural Language Processing) with Python.
#powerful tool to preprocess text data for further analysis

import nltk
nltk.download(['punkt'])
nltk.download('stopwords')

[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data] Unzipping tokenizers/punkt.zip.
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data] Unzipping corpora/stopwords.zip.
True
```

```
#re is a built-in Python package, which can be used to work with Regular Expressions.
import re

DOCUMENT = re.sub(r'\n|\r', ' ', DOCUMENT)
DOCUMENT = re.sub(r' +', ' ', DOCUMENT)
DOCUMENT = DOCUMENT.strip()
```

This code snippet performs text preprocessing on the variable DOCUMENT. Specifically, it replaces newline and carriage return characters with spaces, replaces multiple consecutive spaces with a single space, and removes leading and trailing spaces. These steps are often used to clean up text data and make it more consistent for downstream processing.

## Import Model

BART: Bidirectional and Auto-Regressive Transformers

BART, a denoising autoencoder for pretraining sequence-to-sequence models. BART is trained by

- (1) corrupting text with an arbitrary noising function
- (2) learning a model to reconstruct the original text.

It uses a standard Transformer-based neural machine translation architecture which, despite its simplicity, can be seen as generalizing BERT (due to the bidirectional encoder)

```
[ ] #load BART Model
    from transformers import BartTokenizer, BartForConditionalGeneration, BartConfig
    BART_PATH = 'facebook/bart-large-cnn'
```

## Build Function to process the texts

The `nest_sentences()` function takes a string of text and returns a nested list of sentences, where each sub-list contains up to 1024 characters. This is useful for processing large documents in smaller chunks, which can be more efficient for certain NLP tasks.

The function first initializes an empty list `nested` and an empty list `sent`, which will store the sentences that are grouped together in each sub-list. It also initializes a variable `length` to keep track of the total number of characters in the sentences that have been added to `sent`.

The function then uses the `nlk.sent_tokenize()` function to tokenize the input document into a list of individual sentences. It iterates over each sentence and checks whether adding it to `sent` would exceed the maximum length of 1024 characters. If not, it appends the sentence to `sent` and updates `length`. If adding the sentence would exceed the maximum length, it appends `sent` to `nested`, initializes a new empty list for `sent`, and sets `length` back to 0.

Finally, the function checks if there are any remaining sentences in `sent` and appends them to `nested` if so. The nested list of sentences is then returned as the output of the function.

```
[ ] def nest_sentences(document):  
  
    nested = []  
    sent = []  
    length = 0  
    for sentence in nltk.sent_tokenize(document):  
        length += len(sentence)  
        if length < 1024:  
            sent.append(sentence)  
        else:  
            nested.append(sent)  
            sent = []  
            length = 0  
  
    if sent:  
        nested.append(sent)  
  
    return nested
```

## Tokenization

Words of the paragraph are tokenized for efficient text summarization process.

```
[ ] device = 'cuda'

input_tokenized = bart_tokenizer.encode(' '.join(nested[0]), truncation=True, return_tensors='pt')
input_tokenized = input_tokenized.to(device)
input_tokenized

tensor([[ 0, 133, 928, 8566, 32, 62, 13, 275, 987, 1760,
         8, 275, 2642, 6, 25, 157, 25, 562, 80, 11097,
        11, 5, 275, 2214, 4120, 4, 22, 170, 300, 174,
       101, 42, 662, 44, 27, 7516, 38, 206, 47, 17,
        27, 241, 7076, 17, 27, 1297, 26, 211, 31953, 4,
        22, 2409, 38, 21, 101, 44, 27, 7516, 11380, 6,
        61, 65, 116, 17, 27, 178, 122, 52, 17, 27,
       548, 300, 7076, 13, 237, 4188, 4, 38, 1266, 6,
      26388, 328, 2]], device='cuda:0')

[ ] summary_ids = bart_model.to('cuda').generate(input_tokenized,
                                                  length_penalty=3.0,
                                                  min_length=30,
                                                  max_length=100)

summary_ids

tensor([[ 2, 0, 133, 928, 8566, 32, 62, 13, 275, 987,
        1760, 8, 275, 2642, 6, 25, 157, 25, 562, 80,
       11097, 11, 5, 275, 2214, 4120, 4, 22, 170, 300,
        174, 101, 42, 662, 44, 27, 7516, 38, 206, 47,
        17, 27, 241, 7076, 17, 27, 1297, 26, 211, 31953,
         4, 2]], device='cuda:0')
```

`length_penalty=3.0`: This argument controls how much the length of the generated summary should be penalized. A higher length penalty will result in shorter summaries, while a lower length penalty will result in longer summaries.

`min_length=30`: This argument specifies the minimum length (in tokens) that the summary should be. If the summary is shorter than this, it will be rejected and the model will generate a new summary.

`max_length=100`: This argument specifies the maximum length (in tokens) that the summary should be. If the summary is longer than this, it will be truncated to this length.

## Result

### Input :

```
[ ] DOCUMENT = """
ChatGPT is a large language model (LLM), a machine-learning system that autonomously learns from data and can produce sophisticated and seemingly intelligent writing after training on a massive data set of text. It is the latest in a series of such models released by OpenAI, an AI company in San Francisco, California, and by other firms. ChatGPT has caused excitement and controversy because it is one of the first models that can convincingly converse with its users in English and other languages on a wide range of topics. It is free, easy to use and continues to learn.
This technology has far-reaching consequences for science and society. Researchers and others have already used ChatGPT and other large language models to write essays and talks, summarize literature, draft and improve papers, as well as identify research gaps and write computer code, including statistical analyses. Soon this technology will evolve to the point that it can design experiments, write and complete manuscripts, conduct peer review and support editorial decisions to accept or reject manuscripts.
Conversational AI is likely to revolutionize research practices and publishing, creating both opportunities and concerns. It might accelerate the innovation process, shorten time-to-publication and, by helping people to write fluently, make science more equitable and increase the diversity of scientific perspectives. However, it could also degrade the quality and transparency of research and fundamentally alter our autonomy as human researchers. ChatGPT and other LLMs produce text that is convincing, but often wrong, so their use can distort scientific facts and spread misinformation.
We think that the use of this technology is inevitable, therefore, banning it will not work. It is imperative that the research community engage in a debate about the implications of this potentially disruptive technology. Here, we outline five key issues and suggest where to start.
"""
```

### Output:

#### Text summary

```
[ ] final_summ = generate_summary(nest_summ)
nest_sentences(' '.join(final_summ))

[['ChatGPT is a machine-learning system that autonomously learns from data.',
'It is the latest in a series of such models released by OpenAI, an AI company in San Francisco.']]
```

## CONCLUSION

In this project, we explored the task of Abstractive Text Summarization using the BART model in the Natural Language Processing (NLP) domain. We first preprocessed the text data, then fine-tuned the pre-trained BART model on a dataset of news articles and their corresponding headlines. Finally, we demonstrated how the trained model can be used to generate a summary for a given news article. This project shows the potential of using deep learning models like BART for automatic summarization of large volumes of text data, which can have significant applications in various fields, such as journalism, finance, and healthcare.

## REFERENCES

1. [https://huggingface.co/docs/transformers/model\\_doc/bart](https://huggingface.co/docs/transformers/model_doc/bart)
2. <https://arxiv.org/abs/1910.13461>
3. <https://paperswithcode.com/method/bart>
4. <https://youtu.be/Y2wrtZPrct8>