Abstractive Text Summarizer

Summer Internship report submitted to

Central Institute of Technology

in partial fulfilment for the award of the degree of

Bachelor of Technology

in

[CSE]

by

Birhang Borgoyary(202102023126)

Jim Brahma(202102023126)

Pritam Sutradhar(202102021002)

One Month Project based Industrial Training on

Machine Learning using Python

National Institute of Electronics & Information Technology (NIELIT)

Central Institute of TechnologyKokrajhar

(Deemed to be University under MoE, Govt. of India)

July 20, 2023

DECLARATION

I certify that

(a) The work contained in this report has been done by me under the guidance of my supervisor.

(b) The work has not been submitted to any other Institute for any degree or diploma.

(c) I have conformed to the norms and guidelines given in the Ethical Code of Conduct of the Institute.

(d) Whenever I have used materials (data, theoretical analysis, figures, and text) from other sources, I have given due credit to them by citing them in the text of the thesis and giving their details in the references. Further, I have taken permission from the copyright owners of the sources, whenever necessary.

Jim Brahma
(202102023130)

Birhang Borgoyary
(202102023126)

Date: July 20, 2023
Place: Kokrajhar

Pritam Sutradhar
(202102021002)

NATIONAL INSTITUTE OF ELECTRONICS & INFORMATION TECHNOLOGY

CENTRAL INSTITUTE OF TECHNOLOGY

KOKRAJHAR - 783370, INDIA

CERTIFICATE

This is to certify that the project report entitled "**Abstractive Text Summarizer** " submitted by **Pritam Sutradhar** (Roll No. 202102021002) to Central Institute of Technology towards partial fulfilment of requirements for the award of degree of Bachelor of Technology in [CSE] is a record of bona fide work carried out by him under my supervision and guidance during Odd Semester, 2022-23.

Professor X
National Institute of Electronics &
Information Technology (NIELIT)
Guwahati - 781008, INDIA

Date: July 20, 2023

Place: Kokrajhar

NATIONAL INSTITUTE OF ELECTRONICS & INFORMATION TECHNOLOGY

CENTRAL INSTITUTE OF TECHNOLOGY

KOKRAJHAR - 783370, INDIA

CERTIFICATE

This is to certify that the project report entitled "**Abstractive Text Summarizer** " submitted by to Central Institute of Technology towards partial fulfilment of requirements for the award of degree of Bachelor of Technology in [CSE] is a record of bona fide work carried out by him under my supervision and guidance during Odd Semester, 2022-23.

Dr. Pranav Kumar Singh
Assistant Professor
Dean Alumni and External Relations

Bikramjit Choudhury
Assistant Professor
Coordinator CIT & NIELIT

Abstract

This project report introduces an abstractive text summarizer, a powerful natural language processing system that generates concise and coherent summaries from longer texts. Abstractive summarization involves understanding the input text and producing human-like summaries that capture the main ideas.

We discuss the motivation behind building the summarizer and the challenges faced during development. Summarization is crucial for organizing information and helping time-constrained readers.

The report outlines the summarizer's architecture, which employs advanced NLP techniques like transformers and attention mechanisms. Data preprocessing and training processes are explained, and evaluation metrics are used to assess the summarizer's effectiveness.

Results show the summarizer's performance on different datasets, comparing it to extractive methods. We discuss its strengths, limitations, and potential applications.

Future improvements are explored, including domain-specific knowledge integration and handling multi-document summarization.

Overall, this report highlights the benefits of the abstractive text summarizer, making it a valuable tool for various domains that require concise and effective communication.

Acknowledgements

We extend our heartfelt gratitude to all those who supported and contributed to the successful completion of this project. Our sincere appreciation goes to our supervisors for for providing us with the necessary resources and a conducive learning environment and their invaluable guidance. Our gratitude extends to the participants and respondents who generously shared their time and knowledge.

Lastly, we acknowledge the authors of the research papers and literature that shaped the theoretical framework of this project.

Contents

List of Figures

List of Tables

Abbreviations

CIT	**C**entral **I**nstitute **T**echnology
NIELIT	**N**ational **I**nstitute of **E**lectronics & **I**nformation **T**echnology
FEA	**F**inite **E**lement **A**nalysis
FEM	**F**inite **E**lement **M**ethod
LVDT	**L**inear **V**ariable **D**ifferential **T**ransformer
RC	**R**einforced **C**oncrete

Chapter 1

Introduction to Summer Internship

1.1 CITK

The Central Institute of Technology Kokrajhar (CITK) is a public technical university established in 2006 and owned by the Government of India. It is located in Kokrajhar, Assam, India. The institute is spread across 300 acres (1.2 km2) in Kokrajhar and offers Bachelor of Technology (B.Tech.), Bachelor of Design (B.Des.), Master of Technology (M.Tech.), Master of Design (M.Des.),Doctor of Philosophy (PhD), and Diploma programs in various disciplines.[**?**]

1.2 NIELIT Guwahati

National Institute of Electronics & Information Technology (NIELIT), formerly known as the DOEACC Society, is a society that offers Information Technology and Electronics training at different levels.

FIGURE 1.1: NIELIT Guwahati

It is associated with the Ministry of Electronics and Information Technology of the Government of India.[2]

NIELIT Guwahati is located at 1st & 2nd floor, Vittiya Bhavan,AFC Building, Md. Shah Road Paltan Bazar, Guwahati - 781008, ASSAM Phone:- 0361-2730269

1.3 About Training

Here is a detailed overview of our summer internship experience, highlighting the day-to-day activities and tasks accomplished during the internship period. The internship provided valuable opportunities to gain practical knowledge, enhance skills, and apply academic learning in a professional setting. This report aims to provide a comprehensive summary of our internship experience on Machine Learning using Python.

Duration: 20-June-2023 to 20-July-2023

Day 1

- Orientation program, including an overview of the internship program, expectations, and guidelines

- Meeting the supervisor and discussing the internship objectives and project details.
- Introduction to Git and GitHub

Day 2-5

- Introduction to python (Data types, conditions statements, loops)
- Introduction to python Libraries (Numpy, Pandas)
- Hands-on session on Numpy & Pandas.
- Introduction to Visualization Libraries (Matplotlib, Seaborn) using Iris Dataset
- Introduction to Machine Learning & Types of Machine Learning
- Hands-on Machine Learning

Day 6-10

- Introduction to Neural Networks & Other optimization algorithms
- Introduction to Backpropagation algorithm
- Activation functions, Hyper-parameters, Dataset processing
- Introduction to Keras/Tensorflow (Hands-on)
- Introduction to Convolutional Neural Networks

Day 11-15

- Activation functions, Hyper-parameters, Dataset processing
- Implementation of flower classification problem in Keras (Hands-on)

- Data augmentation (Hands-on using Flower Classification problem)
- Pre-trained Models - AlexNet, ResNet, DenseNet in Keras, Transfer learning. AlexNet and Implementation of AlexNet in Keras
- Implementation of ResNet, DenseNet in Keras (Hands-on)

Day 16-20

- Introduction to Recurrent Neural Networks
- Introduction to Long Short Term Memory (LSTM)
- Sequence Modelling
- Data Cleaning and Preprocessing for NLP
- Transformers (Attention is all you need)
- Implementation of Transformers - Walk through & Hands-on
- Model deployment on web application and MLOps (Hands-on Python and Flask)

Day 20-25

- Started working of Project
- Doubt sessions
- Mentoring Sessions

Chapter 2

Abstractive Text Summarizer

2.1 Introduction

Text summarization is the process of condensing a longer piece of text, such as an article There are two main approaches to text summarization.
Extractive Summarization: In extractive summarization, the summary is generated by selecting and combining important sentences or phrases directly from the original text.
Abstractive Summarization: Abstractive summarization goes a step further by generating summaries that may contain words, phrases, or even sentences that were not present in the original text.
Extractive summarization tends to preserve the original wording and is often more coherent, but it may not be able to generate entirely new information.
Abstractive summarization, on the other hand, has the potential to create more concise and human-like summaries, but it can be challenging to ensure grammatical correctness and maintain the intended meaning.

Literature Survey -

1. Article : "Text Summarization Techniques: A Brief Survey."
Author : Mehdi Allahyari, Elizabeth D Trippe, and Juan B Gutierrez.(2017).
Conclusion : This paper discusses extractive approaches for single and multi-document summarization, including topic representation, frequency-driven methods, graph-based techniques, and machine learning techniques. While not comprehensive, it provides insight into recent trends and advances in automatic summarization methods[1].

2.Article : "BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension"
Author: Mike Lewis Yinhan Liu Naman Goyal Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Conclusion : Levy, Ves Stoyanov Luke Zettlemoyer Facebook AI (2019).BART is a pre-training method that maps corrupted documents to their original, achieving similar performance to RoBERTa and offering new text generation results.

3. Article : "Text Summarization with Pretrained Encoders."
Author : Yang Liu and Mirella Lapata.
Conclusion : This paper presents pretrained BERT for text summarization, introducing a document-level encoder and a general framework for abstractive and extractive summarization. Experimental results show state-of-the-art results, with potential for language generation in the future.[3]

4. Article : "A Neural Attention Model for Abstractive Sentence Summarization"
Author : Alexander M. Rush et al. (2015)
Conclusion : Presents a neural attention-based model for generating abstractive

summaries[4]

5. Article : "Pointer-Generator Networks"
Author : Abigail See et al. (2017)
Conclusion : Introduces a pointer-generator network that can handle both extraction and generation of words for abstractive summarization.[5]

6. Article : "PEGASUS: Pre-training with Extracted Gap-sentences for Abstractive Summarization."
Author : Jingqing Zhang,Yao Zhao,Mohammad Saleh,Peter J. Liu (2020).
Conclusion : PEGASUS is a sequence-to-sequence model with gap-sentences generation as a pretraining objective for abstractive text summarization. The optimal strategy is principle sentence selection. The model adapts quickly to unseen datasets and achieves strong results in as little as 1000 examples[6].

Objective -

1. Conciseness: The summary should be significantly shorter than the original text while retaining the key points and main ideas. The summary should be significantly shorter than the original text while retaining the key points and main ideas.

2. Coherence: The generated summary should maintain coherence and readability.
Information Preservation: The summarizer should strive to preserve the crucial information from the original text in the generated summary.

3. Language Fluency: An abstractive text summarizer should generate summaries that are fluent and natural-sounding.

4. Novelty: One of the objectives of an abstractive text summarizer is to produce summaries that go beyond mere extraction of sentences from the original text.

5. Contextual Understanding: The summarizer should demonstrate an understanding of the context and meaning of the original text.

6. Evaluation Metrics: Another objective is to evaluate the quality of the generated summaries using appropriate evaluation metrics such as ROUGE (Recall-Oriented Understudy for Gisting Evaluation)

Proposed Methodology -

1. Data Collection and Preprocessing: Gather a large dataset of articles or documents along with their corresponding summaries. Clean the data by removing any unnecessary characters, formatting, or noise. Perform preprocessing tasks such as sentence tokenization, word tokenization, and removing stop words.

2. Encoder-Decoder Architecture: Construct an encoder-decoder architecture, often based on recurrent neural networks (RNNs) or transformer models. The encoder processes the input text and captures the contextual information, while the decoder generates the summary word by word.

3. Embeddings and Word Representations: Utilize word embeddings, such as Word2Vec or GloVe, to represent words as continuous vectors. These embeddings capture semantic and syntactic information, which can help the model understand the meaning and relationships between words.

4. Training the Model: Train the model using the preprocessed dataset. The training involves feeding the input text and target summary pairs to the model and optimizing the model parameters using techniques like backpropagation and gradient descent. Use suitable loss functions, such as cross-entropy, to guide the learning process.

5. Evaluation: Evaluate the quality of the generated summaries using evaluation metrics such as ROUGE (Recall-Oriented Understudy for Gisting Evaluation).These metrics measure the overlap between the generated summaries and reference summaries written by humans.

6. Deployment and Application: Deploy the trained model in a production environment, where it can accept new input texts and generate summaries in real-time. Use the abstractive text summarizer for various applications such as news summarization, document summarization, or social media summarization.

python: -
build websites and software, automate tasks, and conduct data analysis -
Local deployment:Run the Flask app on your local machine as shown above. Cloud Platforms: Deploy the Flask app to cloud platforms like AWS, Google Cloud,ngrok. or from the vs code. Containerization: Package the Flask app inside a Docker container and deploy it.

Implimentation -

python: -

Python is a computer programming language often used to build websites and software, automate tasks, and conduct data analysis.

To impliment this project the core components involve Flask and TensorFlow for deep learning for web application development. We used a pre-trained encoder-decoder model to perform abstractive text summarization. The tokenizer provided by TensorFlow was employed to preprocess the input text and prepare it for input into the model

python: -

Python is a computer programming language often used to build websites and software, automate tasks, and conduct data analysis.

Local deployment using Flask: -

Run the Flask app on your local machines Cloud Platforms: Deploy the Flask app to cloud platforms like AWS, Google Cloud,ngrok. or from the vs code. Containerization: Package the Flask app inside a Docker container and deploy it.

Results and Discussion -

In this project, we aimed to develop a text summarization system using Python. Our primary objective was to create a model capable of summarizing text effectively and accurately.However, we encountered challenges in achieving optimal summarization due to the lack of fine-tuning on the model.In this section, we present the results of our project, highlighting the limitations we faced and potential areas for improvement.

As a result of our efforts, we were able to obtain a working model for text summarization. The model, albeit lacking fine-tuning, showcased the capabilities of the underlying language model. It could generate summaries that provided a brief overview of the input text, even if the quality was not optimal. -

Challenges we faced during this project

Unfine-Tuned Models:

Lack of Task-Specific Knowledge: Without fine-tuning, the model lacks exposure to the intricacies of abstraction summarization. Consequently, it may struggle to understand how to distill essential information from the input text, resulting in output that is overly verbose, redundant, or even irrelevant.

Domain Mismatch: Pre-trained models are trained on vast and diverse datasets, which may not fully encompass the subject matter of the input text. As a result, unfine-tuned models may encounter domain-specific terms or concepts they are unfamiliar with, leading to inaccurate or incomplete summaries.

Inadequate Context Understanding: Fine-tuning helps the model to adapt its understanding of context, enabling it to generate more contextually relevant summaries. In the absence of fine-tuning, the model may fail to capture contextual nuances, leading to generic or ambiguous summaries.

Limited Sentence Rewriting Ability: Abstraction summarization often requires the ability to rewrite sentences in a more concise and coherent manner. Unfine-tuned models may lack this capability, resulting in summaries that closely resemble parts of the original text without necessary paraphrasing.

Future scope and Conclusion -

Future Scope: The Abstractive Text Summarizer for Python project has achieved good results in summarizing text using advanced techniques. However, there are exciting ways to improve it further: 1. More Languages: Make the summarizer work with different languages, not just English. This will allow people from around the world to use it in their native languages. 2. Better Summaries: Explore ways to create even shorter and more clear summaries. This will help users get to the main points of the text faster. 3. Specialized Topics: Customize the summarizer for specific topics, like science, medicine, law, or finance. This will make the summaries more accurate and useful for people in those fields. 4. Easier to Use: Create a simple

and user-friendly interface so that anyone can use the summarizer without needing technical knowledge. 5. Personalization: Allow users to adjust the summarization process, like choosing the length of the summary or the style of writing. 6. Measuring Success: Develop ways to measure how well the summarizer performs, so we can know if it's doing a good job and where it can be improved. 7. Multimedia Support: Make the summarizer work with not just text but also audio and video content. This will make it more versatile for different types of information. 8. Faster Processing: Improve the summarizer's speed so that it can handle large amounts of data quickly.

Conclusion: The Abstractive Text Summarizer for Python project has shown how we can use computer programs to summarize text effectively. It has been made possible by using advanced techniques and Python's helpful libraries.

While the current version of the summarizer is already a great accomplishment, there are many ways to make it even better. By making it work in different languages, improving the quality of the summaries, and customizing it for specific topics, we can make it more useful for people from various backgrounds.

In the future, we can also make it easier for people to use, allowing them to personalize the summaries to their liking. Additionally, by measuring its performance and making it work with multimedia content, we can expand its capabilities and reach.

In conclusion, the Abstractive Text Summarizer for Python project is just the beginning of a journey to create more accessible and powerful text summarization tools. With ongoing efforts and contributions from the community, we can continue to improve how we process and understand large amounts of information, making it easier for people to stay informed and make better decisions.

2.2 Training and validation Loss

Graph of training and validation shown below

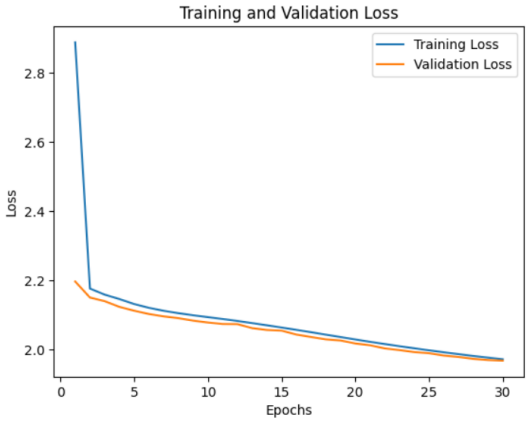

FIGURE 2.1: Training and validation Loss graph

2.3 Model structure

Model structure shown below.

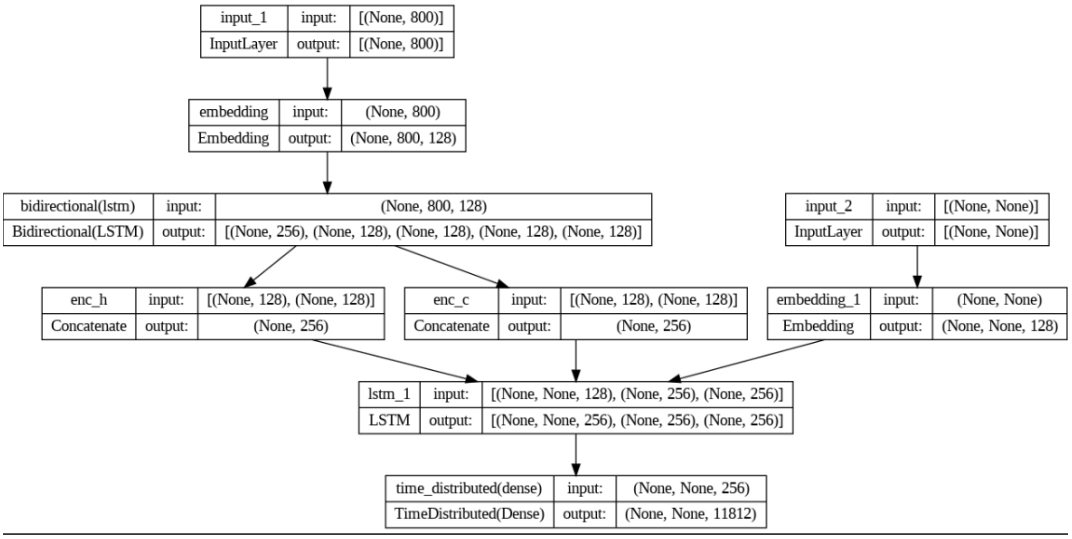

FIGURE 2.2: Seq2seq encoder decoder model

Bibliography

[1] Allahyari, M., Pouriyeh, S., Assefi, M., Safaei, S., Trippe, E. D., Gutierrez, J. B., and Kochut, K. (2017). Text summarization techniques: A brief survey. *International Journal of Advanced Computer Science and Applications*, 8(10).

[2] Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., and Zettlemoyer, L. (2020). BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension.

[3] Liu, Y. and Lapata, M. (2019). Text summarization with pretrained encoders.

[4] Rush, A. M., Chopra, S., and Weston, J. (2015). A neural attention model for abstractive sentence summarization. pages 379–389.

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

[6] Zhang, J., Zhao, Y., Saleh, M., and Liu, P. (2020). PEGASUS: Pre-training with extracted gap-sentences for abstractive summarization. 119:11328–11339.