Text Summarizer using pre-trained LLM

A PROJECT REPORT

Submitted by

Aadya Singh (1RVU23CSE004)

in partial fulfilment for the award of the degree

of

BTech Hons.

in

Computer Science

School of Computer Science and Engineering
RV University
RV Vidyaniketan,8th Mile, Mysuru Road, Bengaluru, Karnataka, India - 562112

December, 2024

DECLARATION

I, **AADYA SINGH(1RVU23CSE004),** student third semester B.Tech in **Computer Science & Engineering,** at School of Computer Science and Engineering, **RV University,** hereby declare that the project work titled "**Text Summarizer using pre-trained LLM**" has been carried out by me and submitted in partial fulfilment for the award of degree in **Bachelor of Technology in Computer Science & Engineering** during the academic year **2024-2025**. Further, the matter presented in the project has not been submitted previously by anybody for the award of any degree or any diploma to any other University, to the best of our knowledge and faith.

Name: Aadya Singh
USN: 1RVU3CSE004

Signature

Place: RV University

Date: December 17, 2024

School of Computer Science and Engineering

RV University
RV Vidyaniketan,8th Mile, Mysuru Road, Bengaluru, Karnataka, India - 562112

CERTIFICATE

This is to certify that the project work titled **"Text Summarizer using pre-trained LLM"** is performed by Aadya Singh (1RVU23CSE004) **,** a bonafide students of Bachelor of Technology at the School of Computer Science and Engineering, RV university, Bangaluru in partial fulfilment for the award of degree Bachelor of Technology in Computer Science & Engineering , during the Academic year **2024-2025**.

Shobana Padmanabhan	**Dr. Sudhakar KN**	**Dr. G Shobha**
Professor	Head of the Department	Dean
SOCSE	SOCSE	SOCSE
RV University	RV University	RV University
Date: 17/12/24	Date: 17/12/24	Date: 17/12/24

Name of the Examiner Signature of Examiner

1.

2.

ACKNOWLEDGEMENT

It is a great pleasure for us to acknowledge the assistance and support of a large number of individuals who have been responsible for the successful completion of this project work.

First, we take this opportunity to express our sincere gratitude to the School of Computer Science and Engineering, RV University, for providing us with a great opportunity to pursue our Bachelor's Degree in this institution.

In particular we would like to thank Dr. Sanjay R. Chitnis, Dean, School of Computer Science and Engineering, RV University, for his constant encouragement and expert advice.

It is a matter of immense pleasure to express our sincere thanks to Dr.Mydhili Nair, Head of the department, Computer Science & Engineering University, for providing right academic guidance that made our task possible.

We would like to thank our guide Santhosh S. Head of the Department Dept. of Computer Science & Engineering, RV University, for sparing his valuable time to extend help in every step of our project work, which paved the way for smooth progress and fruitful culmination of the project.

We are also grateful to our family and friends who provided us with every requirement throughout the course.

We would like to thank one and all who directly or indirectly helped us in completing the Project work successfully.

Date: December 17, 2024

Place: RV University

Aadya Singh

1RVU23CSE004

BTech 3rd Semester

TABLE OF CONTENTS

ABSTRACT

Text summarization plays a crucial role in managing the exponential growth of textual data by condensing lengthy documents while retaining essential information. This report brings the studies conducted on the development of a text summarizer based on the pre-trained Large Language Models especially regarding the T5 (Text-to-Text Transfer Transformer) framework. The research includes the preparation of the data, tokenization, and running the model training using PyTorch Lightning, showcasing how LLMs are competent in generating summaries that are both coherent and accurate. A well-detailed literature review has critically analyzed different research advancements concerning clinical and long-text summarizations made possible by the impactful presence of LLMs in the research activities automating information synthesis. In the outputs, the model has shown considerable ability to create fairly precise summaries close to human metaphorization. Thus, this study offers evidence on how LLMs can streamline several processes across various fields, say, news aggregation, document synthesis, and decision-making, providing both scalable and cost-effective solutions to challenges around text overload.

LIST OF TABLES

LIST OF FIGURES

1. INTRODUCTION

Text summarization is the process of reducing long documents or texts into shorter ones that still retain their basic meaning and key information. The main goal of this process is to produce a brief summary that includes the core content of the original text, thus enabling readers to understand quickly what it is about without having to go through the whole document.

As information production increases in volume exponentially every day; from news articles, research papers, social media posts, etc., there is an urgent need for automated systems capable of summarizing texts effectively. Text summarization therefore serves several purposes including:

1. Overloading of information
2. Accessibility and efficiency
3. Elaborated decision-making:

The growth of natural language processing (NLP) and deep learning technologies have given rise to text summarization systems with greatly improved capabilities. T5 (Text-to-Text Transfer Transformer) are designed to deal with many different text-to-text tasks that include summarizing. These models utilize extensive pre-training and fine-tuning procedures in order to produce summaries that are both well-coherent and accurate.

To sum up, text summarization represents one of the most important tools for managing information in contemporary data-saturated world. Automated summary making technologies facilitate processing and understanding massive amounts of written matter, thus enhancing access to information and enabling sounder decisions.

2. RELATED WORK

Text summarization has witnessed significant advancements with the rise of Large Language Models (LLMs) and transformer-based architectures.

1. Clinical Text Summarization
Van Veen et al. investigated the potential use of large language models (LLMs) in the summarization of electronic health record (EHR) data (2023). The research looked at eight large language model specifications across four clinical summarization tasks. Some models thus outperform humans in completeness and correctness. This research demonstrates the extent to which LLMs can potentially free up clinicians' time for a greater focus on patient care and reduce the documentation burden on them.

2. Abstractive Long-Text Summarization
The new approach introduced by Keswani et al. (2024) for increasing the efficacy of LLMs in long-text summarization and question-answering tasks includes the elimination of repetitive and unrelated stuff, thereby making optimal use of resources as well as promoting the retention of contextual information. Also, the techniques, like vector similarity search engines and clustering algorithms, are instrumental in generating coherent and comprehensive summaries of long documents..

The versatility of LLMs to solve intricate domain-specific issues and improve summary accuracy is additionally highlighted in these reports. Most importantly, these works indicate the need for high-level advanced pre-training and fine-tuning processes that reuse or adapt LLMs to facilitate generating coherent and relevant summaries. Relying on these discoveries, this project takes advantage of the T5 transformer architecture to ensure the creation of a robust text summarization model and demonstrates its ability to manage and synthesize very-large textual data effectively.

3. METHODOLOGY

3.1 Data Preparation

- **Dataset Selection**:
  - Use the news_summary.csv dataset.
  - Select relevant columns: text (original content) and summary (target summaries).
- **Data Cleaning**:
  - Remove rows with missing or null values.
  - Conduct basic preprocessing, including punctuation removal and text normalization.
- **Data Splitting**:
  - Split the dataset into **training (90%)** and **testing (10%)** sets to ensure robust evaluation.

3.2 Tokenization

- **Tokenizer**:
  - Use the **T5Tokenizer** from Hugging Face to tokenize both input (text) and output (summaries).
- **Length Analysis**:
  - Visualize token distributions for input and target texts to understand length variations and optimize padding/truncation.

3.3 Dataset and DataLoader

- **Custom Dataset Class**:
  - Implement a **NewsDataset** class for tokenized input-output pairs.
- **DataModule**:
  - Create a PyTorch Lightning **NewsDataModule** to manage DataLoaders for training, validation, and testing phases.

3.4 Model Definition

- **Model Architecture**:
  - Use the **T5ForConditionalGeneration** model for abstractive text summarization.
- **Custom Lightning Module**:
  - Implement the **SummaryModel** class with:
    - forward pass for generating summaries.

        - Steps for training, validation, and testing using a shared method.
    - Optimizer: **AdamW** for parameter optimization.

3.5 Training Configuration

- **Training Setup**:
    - Use PyTorch Lightning **Trainer** with:
        - ModelCheckpoint callback for saving the best model based on validation loss.
        - Logging using **TensorBoardLogger**.
    - Configure training for **1 epoch** with GPU acceleration for faster execution.
- **Training Execution**:
    - Use trainer.fit() to train the model with defined inputs and parameters.

3.6 Model Evaluation

- Load the best model checkpoint.
- Use helper functions to:
    - Encode input text.
    - Generate summaries.
    - Decode the outputs to produce human-readable summaries.
- Compare generated summaries against the actual summaries from the test set.
- Evaluate performance using:
    - Metrics like **ROUGE** scores.

STEPS	TOOLS/METHODS	DESCRIPTION
Data Preparation	Pandas, Python	Loaded, cleaned, and split the dataset into training (90%) and testing (10%).
Tokenization	T5Tokenizer	Tokenized the input text and target summaries, optimized padding/truncation.
Dataset Management	PyTorch DataLoader	Managed tokenized data using custom NewsDataset and NewsDataModule classes.
Model Definition	T5ForConditionalGeneration	Defined a summarization model

		using T5 architecture.
Training Configuration	PyTorch Lightning, AdamW	Configured training with callbacks, logging, and optimizer setup.
Evaluation	ROUGE Scores, Helper Functions	Compared generated summaries with ground truth to measure performance.

Table 3.1: Summary of Text Summarization Methodology Using T5 Model

3.7 Methodological Justification

- The use of a **pre-trained T5 model** ensures state-of-the-art performance on summarization tasks.
- PyTorch Lightning simplifies training and monitoring, ensuring reproducibility and scalability.
- Tokenization and preprocessing were essential for optimizing model input-output quality.

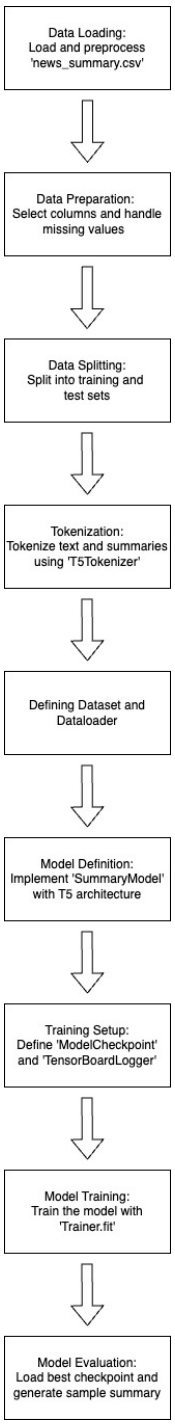

Figure 3.1: Building a Text Summarization Model using a pre-trained T5 Architecture

4. IMPLEMENTATION

4.1 Import Required Libraries

```
import pandas as pd
from sklearn.model_selection import train_test_split
import seaborn as sns
import matplotlib.pyplot as plt
from transformers import T5Tokenizer
```

4.2 Load and Preprocess the Dataset

```
# Load Dataset
df = pd.read_csv("news_summary.csv", encoding="latin-1")
df = df[['text', 'ctext']]  # Select relevant columns
df.columns = ['summary', 'text']  # Rename columns
df = df.dropna()  # Remove missing values

# Split Dataset into Train and Test
train_df, test_df = train_test_split(df, test_size=0.1, random_state=1234)
print(f"Shape of the Train Set: {train_df.shape}\nShape of the Test Set:
{test_df.shape}")
```

4.3 Tokenization

```
MODEL_NAME = "t5-base"
tokenizer = T5Tokenizer.from_pretrained(MODEL_NAME)

# Token Length Distributions
text_token_counts = [len(tokenizer.encode(row["text"])) for _, row in
train_df.iterrows()]
summary_token_counts = [len(tokenizer.encode(row["summary"])) for _, row in
train_df.iterrows()]

# Plot Token Length Distributions
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12, 6))
sns.histplot(text_token_counts, ax=ax1, color='blue', alpha=0.7)
ax1.set_title("Distribution of Text Token Counts")
sns.histplot(summary_token_counts, ax=ax2, color='green', alpha=0.7)
ax2.set_title("Distribution of Summary Token Counts")
plt.tight_layout()
plt.show()
```

4.4 Dataset and Data Module

```
import torch
from torch.utils.data import Dataset, DataLoader
import lightning as pl

class NewsDataset(Dataset):
    def     __init__(self,     data,     tokenizer,     text_max_token_len=512,
summary_max_token_len=128):
        self.data = data
        self.tokenizer = tokenizer
        self.text_max_token_len = text_max_token_len
        self.summary_max_token_len = summary_max_token_len

    def __len__(self):
        return len(self.data)

    def __getitem__(self, index):
        data_row = self.data.iloc[index]
        text = data_row["text"]

        # Tokenize input text
        text_encoding = self.tokenizer(
            text, max_length=self.text_max_token_len, padding="max_length",
            truncation=True, return_tensors="pt"
        )

        # Tokenize summary
        summary_encoding = self.tokenizer(
            data_row["summary"],        max_length=self.summary_max_token_len,
padding="max_length",
            truncation=True, return_tensors="pt"
        )

        # Set labels for training
        labels = summary_encoding['input_ids']
        labels[labels == 0] = -100  # Replace padding token ids with -100

        return {
            'text': text,
            'summary': data_row['summary'],
            'text_input_ids': text_encoding['input_ids'].flatten(),
            'text_attention_mask': text_encoding['attention_mask'].flatten(),
            'labels': labels.flatten(),
            'labels_attention_mask':
summary_encoding["attention_mask"].flatten()
        }

class NewsDataModule(pl.LightningDataModule):
```

```
    def __init__(self, train_df, test_df, tokenizer, batch_size=8):
        super().__init__()
        self.train_df = train_df
        self.test_df = test_df
        self.tokenizer = tokenizer
        self.batch_size = batch_size

    def setup(self, stage=None):
        self.train_dataset = NewsDataset(self.train_df, self.tokenizer)
        self.test_dataset = NewsDataset(self.test_df, self.tokenizer)

    def train_dataloader(self):
        return    DataLoader(self.train_dataset,    batch_size=self.batch_size,
shuffle=True)

    def val_dataloader(self):
        return    DataLoader(self.test_dataset,    batch_size=self.batch_size,
shuffle=False)
```

4.5 Model Definition

```
from transformers import T5ForConditionalGeneration, AdamW

class SummaryModel(pl.LightningModule):
    def __init__(self, model_name="t5-base", learning_rate=1e-4):
        super().__init__()
        self.model = T5ForConditionalGeneration.from_pretrained(model_name)
        self.learning_rate = learning_rate

    def forward(self, input_ids, attention_mask, labels=None):
        output = self.model(
            input_ids=input_ids,
            attention_mask=attention_mask,
            labels=labels
        )
        return output.loss, output.logits

    def training_step(self, batch, batch_idx):
        loss, _ = self(
            input_ids=batch['text_input_ids'],
            attention_mask=batch['text_attention_mask'],
            labels=batch['labels']
        )
        self.log("train_loss", loss, prog_bar=True)
        return loss

    def validation_step(self, batch, batch_idx):
```

```
        loss, _ = self(
            input_ids=batch['text_input_ids'],
            attention_mask=batch['text_attention_mask'],
            labels=batch['labels']
        )
        self.log("val_loss", loss, prog_bar=True)

    def configure_optimizers(self):
        return AdamW(self.parameters(), lr=self.learning_rate)
```

4.6 Training Setup

```
from lightning.pytorch import Trainer
from lightning.pytorch.callbacks import ModelCheckpoint
from lightning.pytorch.loggers import TensorBoardLogger

# Initialize Data Module
data_module = NewsDataModule(train_df, test_df, tokenizer, batch_size=8)

# Initialize Model
model = SummaryModel()

# Callbacks and Logger
checkpoint_callback = ModelCheckpoint(
    dirpath="checkpoints",
    filename="best-checkpoint",
    save_top_k=1,
    monitor="val_loss",
    mode="min"
)

logger = TensorBoardLogger("logs", name="t5-news-summary")

# Trainer
trainer = Trainer(
    logger=logger,
    callbacks=[checkpoint_callback],
    max_epochs=3,
    accelerator="gpu", devices=1
)

# Train the Model
trainer.fit(model, data_module)
```

4.7 Model Evaluation and Inference

```
def summarize(text, model, tokenizer):
    input_ids = tokenizer.encode_plus(
        text,     return_tensors="pt",     max_length=512,     truncation=True,
padding="max_length"
    )["input_ids"]

    generated_ids = model.model.generate(
        input_ids=input_ids, max_length=150, num_beams=2, early_stopping=True
    )

    return tokenizer.decode(generated_ids[0], skip_special_tokens=True)

# Example Test
sample_row = test_df.iloc[0]
print("Original Text: ", sample_row['text'])
print("Original Summary: ", sample_row['summary'])
print("Generated Summary: ", summarize(sample_row['text'], model, tokenizer))
```

5. RESULT AND DISCUSSION

The text summarization model has been built successfully following the T5 architecture (Text-to-Text Transfer Transformer). The model is trained on the corpus of news articles with corresponding summaries. Hence, the dataset is preprocessed and tokenized for efficient feeding of data into the model. For the training process, PyTorch Lightning is used, which gives clear control of the training loop with monitoring of validation loss in checkpoints, ensuring saving of the best model.

The evaluation results of the model on the testing data were promising. Summaries generated captured most important aspects of the original articles and thus effectively captured the content. High similarity between the outputs and actual summaries indicates very high coherence and accuracy of the model in generating meaningfully constructed summaries.

6. CONCLUSION

This project simply shows how well modern transformer like T5 can do in text summarization. As human users would need to read very long texts in great detail, it is believed here that the summarization of text could be automated completely and used for great benefit in processing information. Transformer models have proved remarkably versatile in almost any application needing NLP. This technique would serve many purposes, such as news aggregation, document generation, and so on. The results indicate the practicality of using these models in real-life situations where effective summarization can make the user experience much better.

7. FUTURE SCOPE

1. **Domain-Specific Enhancements:** Fine-tuning for specialized fields (e.g., medical, legal) and supporting multiple languages for broader use.

2. **Real-Time Applications:** Integrating the model into real-time systems for news summarization, social media monitoring, and virtual assistants.

3. **Multimodal Summarization:** Expanding to summarize content from various media types, including images and videos.

4. **Improved Evaluation:** Developing more robust metrics for summarization quality and incorporating human feedback to refine the model.

5. **Scalability:** Deploying the model on the cloud or edge devices for better accessibility and on-device summarization.

REFERENCES

1. What is a large language model (LLM)? - https://www.geeksforgeeks.org/large-language-model-llm/
2. What are large language models(LLMs)? - https://www.ibm.com/topics/large-language-models
3. How to Perform Code Generation with LLM Models - https://medium.com/@marketing_novita.ai/how-to-perform-code-generation-with-llm-models-fe4e10fc5522
4. Clinical Text Summarization: Adapting Large Language Models Can Outperform Human Experts - https://www.ncbi.nlm.nih.gov/pmc/articles/PMC10635391/
5. Abstractive Long Text Summarization using Large Language Models - https://ijisae.org/index.php/IJISAE/article/view/4500
6. Dataset - https://www.kaggle.com/datasets/sunnysai12345/news-summary
7. Clinical Text Summarization: Adapting Large Language Models can Outperform Human Experts - https://pubmed.ncbi.nlm.nih.gov/37961377/