Dr. Vishwanath Karad
MIT WORLD PEACE UNIVERSITY | PUNE
TECHNOLOGY, RESEARCH, SOCIAL INNOVATION & PARTNERSHIPS

Department of Computer Engineering and Technology

Third Year

Course: Cognitive Computing and Natural Language Processing (CET4005B)
CCA3- BOT development/Case study Implementation
Panel - E

Student Name- Pranjal Vishwakarma

PRN- 1032222756 Roll No- 24

Faculty Name- Dr.Amruta Aphale

LEVERAGING TEXT TO TEXT TRANSFER TRANSFORMERS (T5):

FOR CONTEXT AWARE NEWS ARTICLE SUMMARIZATION

DR.AMRUTA APHALE

PRANJAL VISHWAKARMA
1032222756
PANEL E- 24

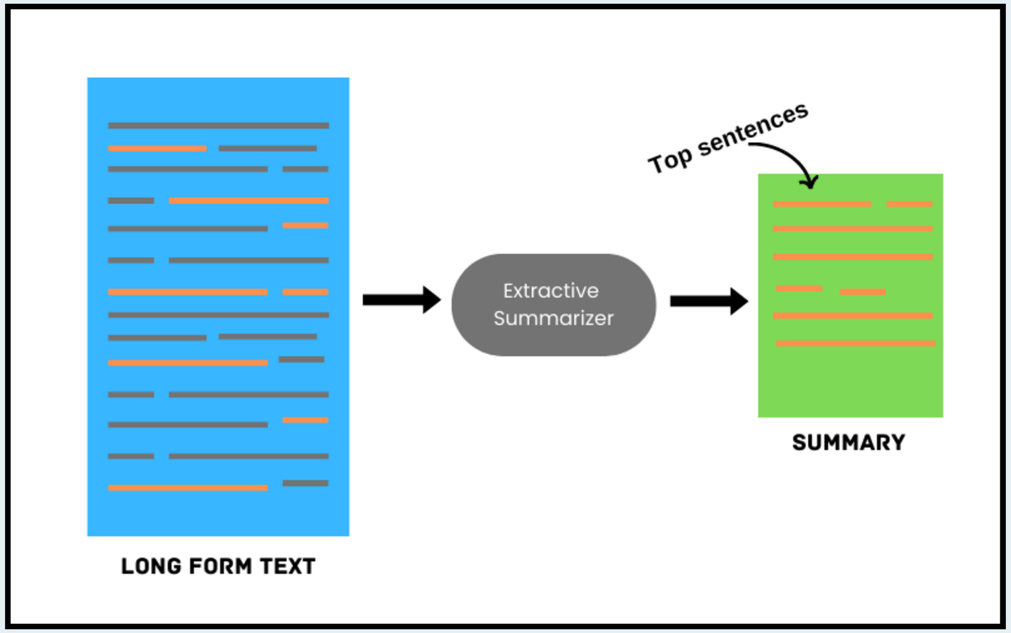
INTRODUCTION

The internet is flooded with news articles daily, making it difficult for readers to consume and process long pieces of information.

Summarization helps compress these articles into concise summaries without losing the core message.

This project explores how state-of-the-art NLP models, particularly T5, can be used to automatically generate high-quality summaries of Indian news articles.

The solution aims to be language-aware, context-sensitive, and scalable for real-world news summarization applications.



BACKGROUND INFORMATION

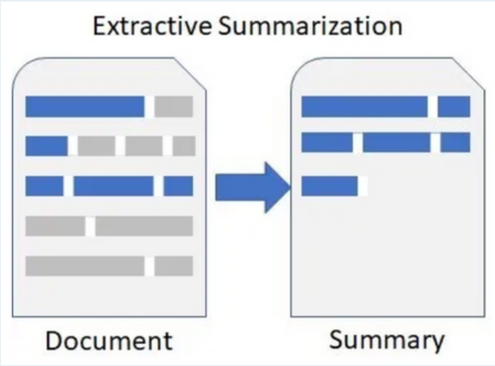
Text summarization is a well-studied task in Natural Language Processing, divided into:

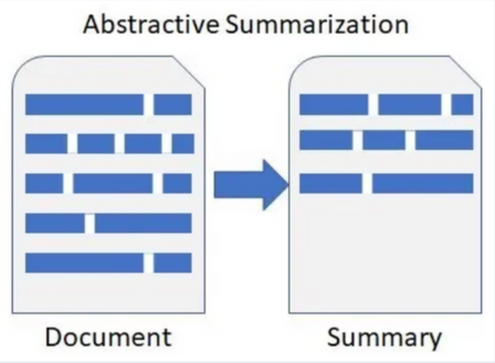
- **Extractive summarization** (picks important sentences)
- **Abstractive summarization** (generates new sentences based on meaning)

Traditional approaches struggle with semantic understanding and context relevance.

Google's T5 (Text-To-Text Transfer Transformer) treats every NLP task as a text generation problem, making it ideal for abstractive summarization.

Indian news often contains culturally-specific or domain-specific entities—requiring a context-aware model trained on local datasets.





OBJECTIVES

The primary objective of this project is to:

01 Build and fine-tune a T5 model for automatic abstractive summarization of Indian news articles.

02 Evaluate the performance using both quantitative metrics (e.g., ROUGE) and qualitative analysis (e.g., side-by-side comparisons).

03 Demonstrate the model's ability to reduce long news articles into clear, factual, and readable summaries.

METHODOLOGY

01

Dataset Preparation:

- Source: Custom CSV file with 4500+ Indian news articles and human-written summaries.
- Preprocessing:
  - Lowercasing, cleaning special characters.
  - Tokenization using T5Tokenizer.
  - Appended "summarize: " prefix to each article as required by the T5 model.

02

Model and Training:

- Base Model: t5-small from HuggingFace Transformers.
- Fine-Tuning Environment: Google Colab (GPU), PyTorch.
- Key Libraries: HuggingFace Transformers, W&B for experiment tracking.

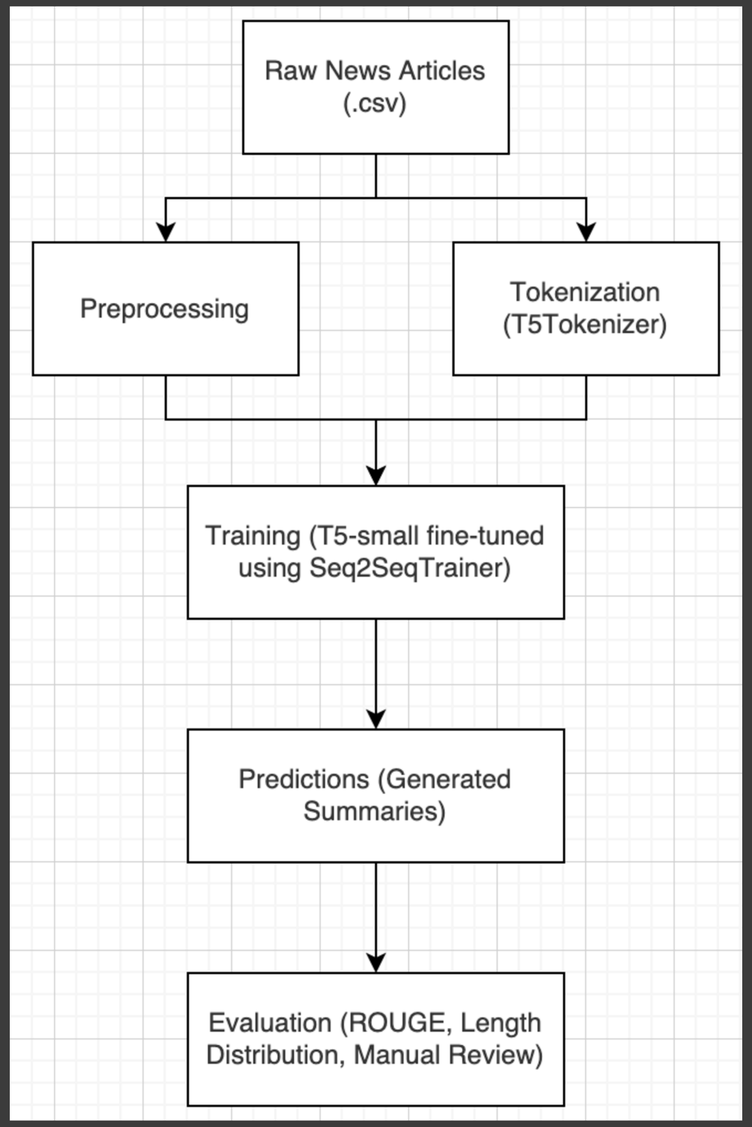
03

Training Strategy:

- Used Seq2SeqTrainer from HuggingFace.
- Monitored loss, validation performance.
- Optimized with AdamW and a learning rate scheduler.
- Logging with WandB for transparency and analysis.

IMPLEMENTATION

Steps:



IMPLEMENTATION

Technical Stack:

- Language: Python
- Libraries: PyTorch, Transformers (HuggingFace), Pandas, Matplotlib
- Tracking: Weights & Biases (wandb.ai)
- Runtime: Google Colab with CUDA support

Supporting Components:

- DataCollator to handle padding and tensor alignment
- Custom DataLoader for handling batched sequences
- Visualization scripts for length and keyword analysis

FiNDiNGS/ANALYSiS

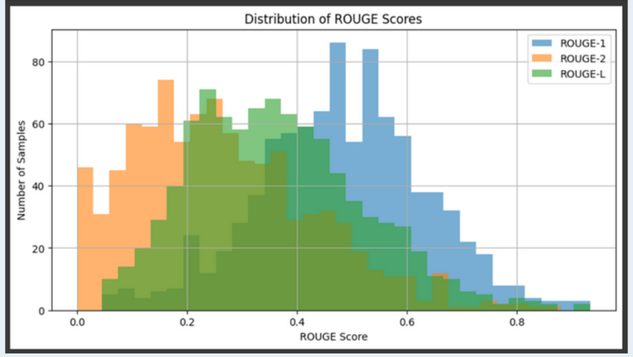
Quantitative:

ROUGE scores showed satisfactory overlap with reference summaries.

```python
import pandas as pd
from rouge_score import rouge_scorer
from tqdm import tqdm

# Load predictions
df = pd.read_csv('predictions.csv')

# Initialize ROUGE scorer
scorer = rouge_scorer.RougeScorer(['rouge1', 'rouge2', 'rougeL'], use_stemmer=True)

# Compute ROUGE scores
rouge1_scores, rouge2_scores, rougeL_scores = [], [], []

for i in tqdm(range(len(df))):
    reference = str(df.loc[i, 'Actual Text'])
    prediction = str(df.loc[i, 'Generated Text'])
    scores = scorer.score(reference, prediction)

    rouge1_scores.append(scores['rouge1'].fmeasure)
    rouge2_scores.append(scores['rouge2'].fmeasure)
    rougeL_scores.append(scores['rougeL'].fmeasure)

# Add to dataframe
df['ROUGE-1'] = rouge1_scores
df['ROUGE-2'] = rouge2_scores
df['ROUGE-L'] = rougeL_scores

# Print average scores
print("\nAverage ROUGE-1:", round(df['ROUGE-1'].mean(), 4))
print("Average ROUGE-2:", round(df['ROUGE-2'].mean(), 4))
print("Average ROUGE-L:", round(df['ROUGE-L'].mean(), 4))
```

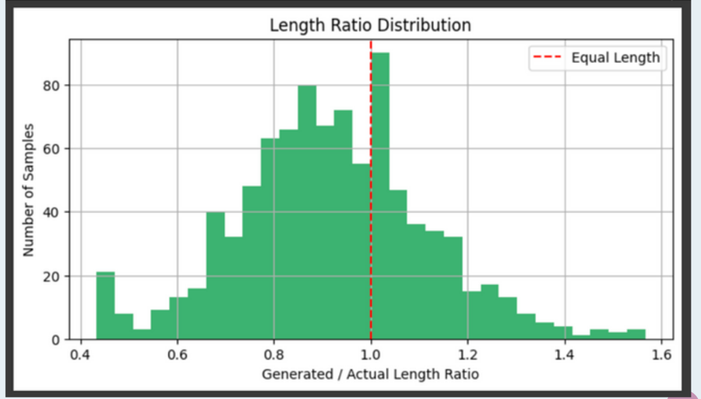
```
100%|██████████| 903/903 [00:02<00:00, 389.19it/s]
Average ROUGE-1: 0.4827
Average ROUGE-2: 0.2641
Average ROUGE-L: 0.364
```

```python
import matplotlib.pyplot as plt

plt.figure(figsize=(10, 5))
plt.hist(df['ROUGE-1'], bins=30, alpha=0.6, label='ROUGE-1')
plt.hist(df['ROUGE-2'], bins=30, alpha=0.6, label='ROUGE-2')
plt.hist(df['ROUGE-L'], bins=30, alpha=0.6, label='ROUGE-L')
plt.xlabel("ROUGE Score")
plt.ylabel("Number of Samples")
plt.title("Distribution of ROUGE Scores")
plt.legend()
plt.grid(True)
plt.show()
```



Most generated summaries fell between 20-40 words, matching target length.

```python
#To quantitatively show how concise the model's output is compared to the input

df['len_ratio'] = df['gen_len'] / df['act_len']

plt.figure(figsize=(8, 4))
plt.hist(df['len_ratio'], bins=30, color='mediumseagreen')
plt.axvline(1.0, color='red', linestyle='--', label='Equal Length')
plt.xlabel("Generated / Actual Length Ratio")
plt.ylabel("Number of Samples")
plt.title("Length Ratio Distribution")
plt.legend()
plt.grid(True)
plt.show()
```




```python
print('Now generating summaries on our fine tuned model for the validation dataset and saving it in a dataframe')
for epoch in range(config.VAL_EPOCHS):
    predictions, actuals = validate(epoch, tokenizer, model, device, val_loader)
    final_df = pd.DataFrame({'Generated Text':predictions,'Actual Text':actuals})
    final_df.to_csv('predictions.csv')
    print('Output Files generated for review')
```

```
                                                text  \
0  The Administration of Union Territory Daman an...
1  Malaika Arora slammed an Instagram user who tr...
2  The Indira Gandhi Institute of Medical Science...
3  Lashkar-e-Taiba's Kashmir commander Abu Dujana...
4  Hotels in Maharashtra will train their staff t...

                                               ctext
0  summarize: The Daman and Diu administration on...
1  summarize: From her special numbers to TV?appe...
2  summarize: The Indira Gandhi Institute of Medi...
3  summarize: Lashkar-e-Taiba's Kashmir commander...
4  summarize: Hotels in Mumbai and other Indian c...
FULL Dataset: (4514, 2)
TRAIN Dataset: (3611, 2)
TEST Dataset: (903, 2)
```

Fine-tuned T5 produced readable and informative summaries across 900+ test articles.

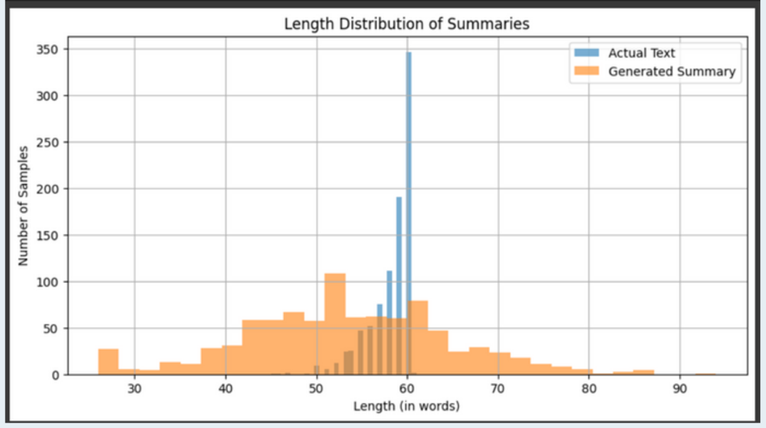
FiNDiNGS/ANALYSiS

Visualizations:

Word Cloud: Most frequent tokens in both generated summaries and original text indicated topical preservation.

Length Distribution Histogram: Compared actual vs. generated summary lengths showed meaningful reduction.

```python
#Shows how consistent the model is compared to the original summaries

import pandas as pd
import matplotlib.pyplot as plt

df = pd.read_csv('predictions.csv')

df['act_len'] = df['Actual Text'].str.split().apply(len)
df['gen_len'] = df['Generated Text'].str.split().apply(len)

plt.figure(figsize=(10, 5))
plt.hist(df['act_len'], bins=30, alpha=0.6, label='Actual Text')
plt.hist(df['gen_len'], bins=30, alpha=0.6, label='Generated Summary')
plt.xlabel("Length (in words)")
plt.ylabel("Number of Samples")
plt.title("Length Distribution of Summaries")
plt.legend()
plt.grid(True)
plt.show()
```




```python
#Quickly shows what kind of words dominate input vs. generated summaries

from wordcloud import WordCloud

actual_words = ' '.join(df['Actual Text'])
generated_words = ' '.join(df['Generated Text'])

plt.figure(figsize=(14,6))

plt.subplot(1,2,1)
plt.title("Actual Text WordCloud")
plt.imshow(WordCloud(width=800, height=400, background_color='white').generate(actual_words), interpolation='bilinear')
plt.axis('off')

plt.subplot(1,2,2)
plt.title("Generated Summary WordCloud")
plt.imshow(WordCloud(width=800, height=400, background_color='white').generate(generated_words), interpolation='bilinear')
plt.axis('off')

plt.tight_layout()
plt.show()
```

Sample Predictions Table: Showed side-by-side comparison of news articles and their generated summaries.

Key Insights:

- Model preserved named entities and event descriptions well.
- Shorter articles had better summaries; longer ones sometimes missed context (due to token limit).

CHALLENGES

Token Size Mismatch

Transformer models require input and output tensors to be of the same shape within a batch, but natural text varies in length.

Handled using HuggingFace's DataCollatorForSeq2Seq

Training Instability

Training a large model like T5 on limited resources can cause high loss variance and slow convergence.

Used gradient accumulation and learning rate warmup

Hallucination (Invented Info)

Transformer models sometimes generate plausible-sounding but incorrect or unrelated facts (a known issue in text generation called "hallucination").

Controlled via early stopping and beam search decoding

Small Dataset

With limited training samples, there's a risk of overfitting and poor generalization.

Mitigated overfitting with dropout, weight decay

FUTURE SCOPE

- **Scale to Larger T5 Variants:** *Try t5-base, t5-large for better comprehension.*
- **Integrate Named Entity Protection:** *Ensure factual accuracy using NER tagging.*
- **Multilingual Support:** *Extend to regional languages (e.g., Marathi, Hindi).*
- **Deploy as Web Tool:** *Build a React/Flask app for live summarization.*
- **Use Real-Time News Feeds:** *Scrape news APIs (like NewsAPI.org) for continuous learning.*

THANK YOU