SINGLE DOCUMENT HYBRID TEXT SUMMARIZATION WITH TRANSFORMERS FOR TELUGU LANGUAGE

Report submitted to the SASTRA Deemed to be University
as the requirement for the course

CSE300 - MINI PROJECT

Submitted by

PAMARTHI SAI NARASIMHAM
(Reg. No.: 226003167, B.Tech., CSE)
THUPAKULA PE PRUDHVI NARAYANA
(Reg. No.: 226003139, B.Tech., CSE)
MUSANI HARI SRAVANTH REDDY
(Reg. No.: 226003164, B.Tech., CSE)

April 2025

Department of Computer Science and Engineering

SRINIVASA RAMANUJAN CENTRE

KUMBAKONAM, TAMIL NADU, INDIA – 612001

SASTRA

ENGINEERING · MANAGEMENT · LAW · SCIENCES · HUMANITIES · EDUCATION

DEEMED TO BE UNIVERSITY

(U/S 3 of the UGC Act,1956)

THINK MERIT | THINK TRANSPARENCY | THINK SASTRA

THANJAVUR | KUMBAKONAM | CHENNAI

Department of Computer Science and Engineering

SRINIVASA RAMANUJAN CENTRE

KUMBAKONAM, TAMIL NADU, INDIA – 612001

Bonafide Certificate

This is to certify that the report titled "**Single Document Hybrid Text Summarization with Transformers for Telugu Language**" submitted as a requirement for the course, **CSE300 : MINI PROJECT** for B.Tech. is a Bonafide record of the work done by **Mr. Pamarthi Sai Narasimham (Reg. No. 226003167), Mr. Thupakula Prudhvi Narayana (Reg. No. 226003139)** and **Mr. Musani Hari Sravanth Reddy (Reg. No. 226003164)** during the academic year 2024-25, in the Srinivasa Ramanujan Centre, under my supervision.

Signature of Project Supervisor : S.Hemamalini

Name with Affiliation : **Smt. S. Hemamalini, AP-II, Department of CSE**

Date : 25-04-25

Mini Project *Viva voce* held on 26-04-2025

Examiner 1

(Dr. A. Neraja
AP-I/CSE (SRC)

Examiner 2

Department of Computer Science and Engineering

SRINIVASA RAMANUJAN CENTRE

KUMBAKONAM, TAMIL NADU, INDIA – 612001

Bonafide Certificate

This is to certify that the report titled "**Single Document Hybrid Text Summarization with Transformers for Telugu Language**" submitted as a requirement for the course, **CSE300 : MINI PROJECT** for B.Tech. is a Bonafide record of the work done by **Mr. Pamarthi Sai Narasimham (Reg. No. 226003167), Mr. Thupakula Prudhvi Narayana (Reg. No. 226003139)** and **Mr. Musani Hari Sravanth Reddy (Reg. No. 226003164)** during the academic year 2024-25, in the Srinivasa Ramanujan Centre, under my supervision.

Signature of Project Supervisor :
Name with Affiliation : Smt. S. Hemamalini, AP-II, Department of CSE
Date :

Mini Project *Viva voce* held on _____

Examiner 1 **Examiner 2**

ACKNOWLEDGEMENT

We would like to thank our Honorable Chancellor **Prof. R. Sethuraman** for providing us with an opportunity and the necessary infrastructure for carrying out this project as a part of our curriculum.

We would like to thank our Honorable Vice-Chancellor **Dr. S. Vaidhyasubramaniam** and **Dr. S. Swaminathan**, Dean, Planning & Development, for the encouragement and strategic support at every step of our college life.

We extend our sincere thanks to **Dr. R. Chandramouli**, Registrar, SASTRA Deemed to be University for providing the opportunity to pursue this project.

We extend our heartfelt thanks to **Dr. V. Ramaswamy**, Dean and **Dr. A. Alli Rani**, Associate Dean, Srinivasa Ramanujam Centre for their constant support and suggestions when required without any reservations.

We sincerely extend our gratitude to our guide **Smt. S. Hemamalini, AP-II, Department of CSE,** who was the driving force behind this whole idea from the start. Her deep insight in the field and invaluable suggestions helped us in making progress throughout our project work.

We would like to thank our project review panel members **Dr. Nagamuthu Krishnan SS**, Sr. AP, Department of CSE and **Dr. Menaga A,** AP-I, Department of CSE for their valuable comments and meticulous support to complete this project successfully.

We would like to place on record, our gratitude to **Dr. Vanitha M,** AP-II, Department of CSE, our Project Coordinator for her benevolent approach and to all the teaching and non-teaching faculty of the Department of CSE who have either directly or indirectly helped us in the completion of the project.

We gratefully acknowledge all the contributions and encouragement from our family and friends resulting in the successful completion of this project. We thank you all for providing us an opportunity to showcase our skills through project.

LIST OF FIGURES

Figure No.	Title	Page No.
1	Architecture Diagram	4
2	Model Architecture for Transformer	5
3	ROUGE Scores comparison graph	18

LIST OF TABLES

Figure No.	Title	Page No.
1	ROUGE Scores comparison table	17

ABBREVIATIONS

ILSUM	Indian Language Summarization
BERT	Bidirectional Encoder Representations from Transformers
Bi-LSTM	Bidirectional Long Short-Term Memory
ELMo	Embeddings from Language Models
Epochs	Epochs (No of Iterations)
FSS	Final Sentence Score
GloVe	Global Vectors for Word Representation
GPU	Graphical Processing Unit
LCS	Longest Common Subsequence
LSTM	Long Short-Term Memory
MLP	Multi-Layer Perceptron
NLP	Natural Language Processing
OOV	Out Of Vocabulary
POS	Part of Speech
RNN	Recurrent Neural Network
ROUGE	Recall-Oriented Understudy for Gisting Evaluation
T2T	Text-to-Text
TF-IDF	Term Frequency-Inverse Document Frequency

NOTATIONS

English Symbols

h_t	Hidden state at time step t in RNN/LSTM
s_t	Decoder state at time step t
a_t	Attention distribution at time step t
c_t	Coverage vector at time step t
P_vocab	Vocabulary distribution (probability of each word)
w	Predicted word
t	Time step in sequence
L	Number of layers or length of sequence
n	Number of sentences or summary size
E	Embedding function
v	Learnable parameter vector in attention mechanism
W_h, W_s	Weight matrices in attention mechanism
b_{attn}	Bias term in attention mechanism

Greek Symbols

α	Initial learning rate
β	learning rate (Adam Optimization)
θ	initialized model parameters
θ'	modified model parameters
Σ	Summation
∇	Gradient

ABSTRACT

Telugu, one of the most widely spoken South Indian languages, is seeing a rapid increase in digital content, making it difficult to manually process and summarize large text datasets. The query we address in this project is: How can we develop an efficient and high-quality automatic summarization system for Telugu text.

Existing summarization models rely on either extractive methods, which simply select key sentences, or abstractive methods, which rephrase content but require extensive training time and struggle with redundancy. Neither approach alone fully satisfies the need for concise, accurate, and readable summaries.

As a solution, we propose a hybrid text summarization model that combines the strengths of both approaches. First, we use the TextRank algorithm to identify the most important sentences in the input text using extractive summarization. Then, we apply a transformer with attention mechanisms to generate an abstractive summary from the extract. This is achieved by utilizing coverage property, redundancy reduction, coherence and relevance properties to reduce repetition and improve the coherence of the final summary.

Keywords: Automatic summarization, TextRank algorithm, Transformer,Bi-LSTM neural network, Attention mechanisms, Coverage property, Coherence property, relevance and redundancy reductions.

Base Paper Details: Babu, G. L. Anand, Badugu, Srinivasu. *Multi-Document Hybrid Text Summarization with Bi-LSTM RNN for Telugu Language*. Sādhanā (Indian Academy of Sciences), 49, 155, 2024. (SCI / SCI-E).

Table of Contents

Title	Page No.
Bonafide Certificate	ii
Acknowledgement	iii
List of figures	iv
List of tables	iv
Abbreviations	v
Notations	vi
Abstract	vii
1. Summary of the base paper	1
2. Merits and Demerits of the base paper	9
3. Source Code	11
4. Snapshots	16
5. Conclusion and Future Plans	19
6. References	20
7. Appendix – Base paper	22

CHAPTER 1

SUMMARY OF THE BASE PAPER

TITLE: Multi-document hybrid text summarization with bi-LSTM RNN for Telugu language

Journal Name: Sådhanå (2024) 49:155

Publisher: Springer (on behalf of the Indian Academy of Sciences)

Year: 2024

Indexing: As a Springer journal, *Sådhanå* is indexed in major databases including SCI-E, Scopus, Web of Science, DBLP, Google Scholar, and INSPEC.

1. SUMMARY

The paper presents a new way to summarize Telugu text documents using a **hybrid approach** that combines both extractive and abstractive summarization techniques. Telugu is a widely spoken South Indian language with increasing digital content, which makes it important to develop summarization systems to process and shorten large texts efficiently.

In this method, the **TextRank algorithm** is first used to pick out the most important sentences from the document (extractive method). Then, these selected sentences are passed into a **Text-to-Text(T5) Transformer model** to produce a more natural and readable summary (abstractive method).

The attention mechanism helps the model focus on the most relevant parts of the input while generating each word of the summary. The coverage mechanism is added to reduce repeated words and improve summary quality.

The authors tested this model on Telugu news articles collected from popular websites like Sakshi and Eenadu. They used the **ROUGE toolkit** to measure the model's performance, comparing it to other existing methods. The results showed that the proposed hybrid model performed better in terms of F1 score, precision, and recall.

Overall, this model is effective in summarizing Telugu texts faster and more accurately by taking the best of both extractive and abstractive summarization methods.

1.1 INTRODUCTION

Telugu is one of India's most spoken languages, with over 84 million speakers. With the rise in Telugu digital content, there is a growing need for effective automatic text summarization systems.

Summarization methods fall into two main categories: extractive (which selects significant sentences directly from the text) and abstractive (which generates new sentences).

Extractive methods are efficient but limited in natural language fluency, while abstractive methods are fluent but computationally expensive.

This paper proposes a hybrid summarization model that combines both approaches. It uses the TextRank algorithm for extractive summarization and a Bi-directional LSTM (Bi-LSTM) based sequence-to-sequence model with attention and coverage mechanisms for abstractive summarization.

This model is designed to reduce training time, enhance performance, and minimize redundancy in the generated summaries.

1.2 PROBLEM STATEMENT

Extractive models, which select key sentences, often lack coherence and natural flow. Abstractive models, though more fluent, require extensive training time and data, and they struggle with issues like repetition and semantic errors.

The query we address in this project is: How can we develop an efficient and high-quality automatic summarization system for Telugu text.

Existing summarization models rely on either extractive methods, which simply select key sentences, or abstractive methods, which rephrase content but require extensive training time and struggle with redundancy. Neither approach alone fully satisfies the need for concise, accurate, and readable summaries.

1.3 LITERATURE SURVEY

In 2004, Mihalcea and Tarau introduced the TextRank algorithm, a graph-based method for ranking sentences based on their connectivity and importance within the text. This approach became a foundational technique in extractive summarization, allowing significant sentences to be identified without supervised learning. TextRank is particularly effective for summarizing structured documents and is computationally efficient, making it widely used in many summarization systems.

Later in 2011, Kallimani et al. explored an abstractive summarization method tailored for Indian regional languages, including Telugu. Their approach involved using TF-IDF to extract keywords and generate summaries by rephrasing sentences. Although it marked an early effort toward Telugu abstractive summarization, the method was largely rule-based and lacked deep learning techniques, which limited its ability to generate semantically rich summaries.

In 2014, Bahdanau et al. made a major breakthrough in neural machine translation and summarization by proposing the attention mechanism. This model allowed the decoder to focus on relevant parts of the input sequence while generating each word, significantly improving performance, especially for longer texts. It addressed the limitations of earlier encoder-decoder models, which compressed all information into a single vector.

See et al., in 2017, further improved the attention mechanism by introducing a coverage model. Their work used a pointer-generator network that allowed copying words from the input text while keeping track of what parts had already been summarized. This helped in reducing repetition and producing more coherent summaries, making the approach highly effective in neural abstractive summarization.

In 2019, Shashikanth and Sanghavi proposed an extractive summarization model for the Telugu language using clustering methods such as k-means. The model grouped similar sentences and selected representative ones based on frequency and position. While useful, the model did not address semantic redundancy or sentence fluency.

Khanam's work in 2016 proposed another extractive method for Telugu summarization using frequency-based sentence ranking. The process included tokenization, stop-word removal, and calculating word frequencies to rank sentences. Though simple, the model lacked the abstraction needed for natural summaries and struggled with text coherence.

Sudha and Latha in 2020 presented a multi-document abstractive summarization approach for Telugu using semantic similarity and Recurrent Neural Networks (RNNs). Their model calculated sentence similarity using semantic features and applied an RNN for generating summaries. While it incorporated deep learning, it was limited by dataset size and did not integrate attention or coverage mechanisms.

Rush et al. in 2015 introduced a neural attention model for sentence-level abstractive summarization. Their model generated fluent summaries by focusing dynamically on different parts of the input sentence. It handled rare and out-of-vocabulary words better than previous models and set the stage for future attention-based models in NLP tasks.

1.4 ARCHITECTURE DIAGRAM

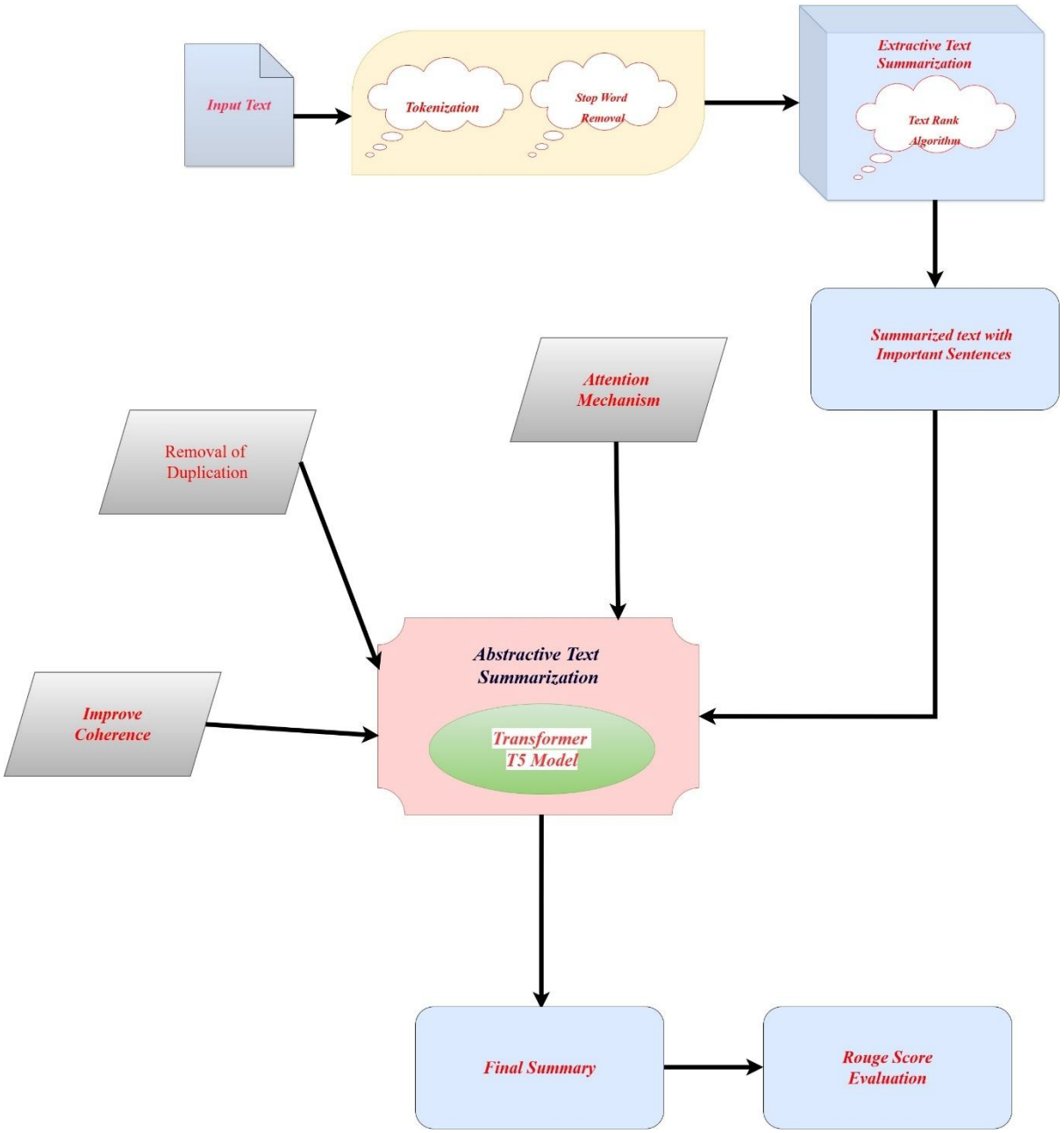


Fig. 1. Architecture Diagram

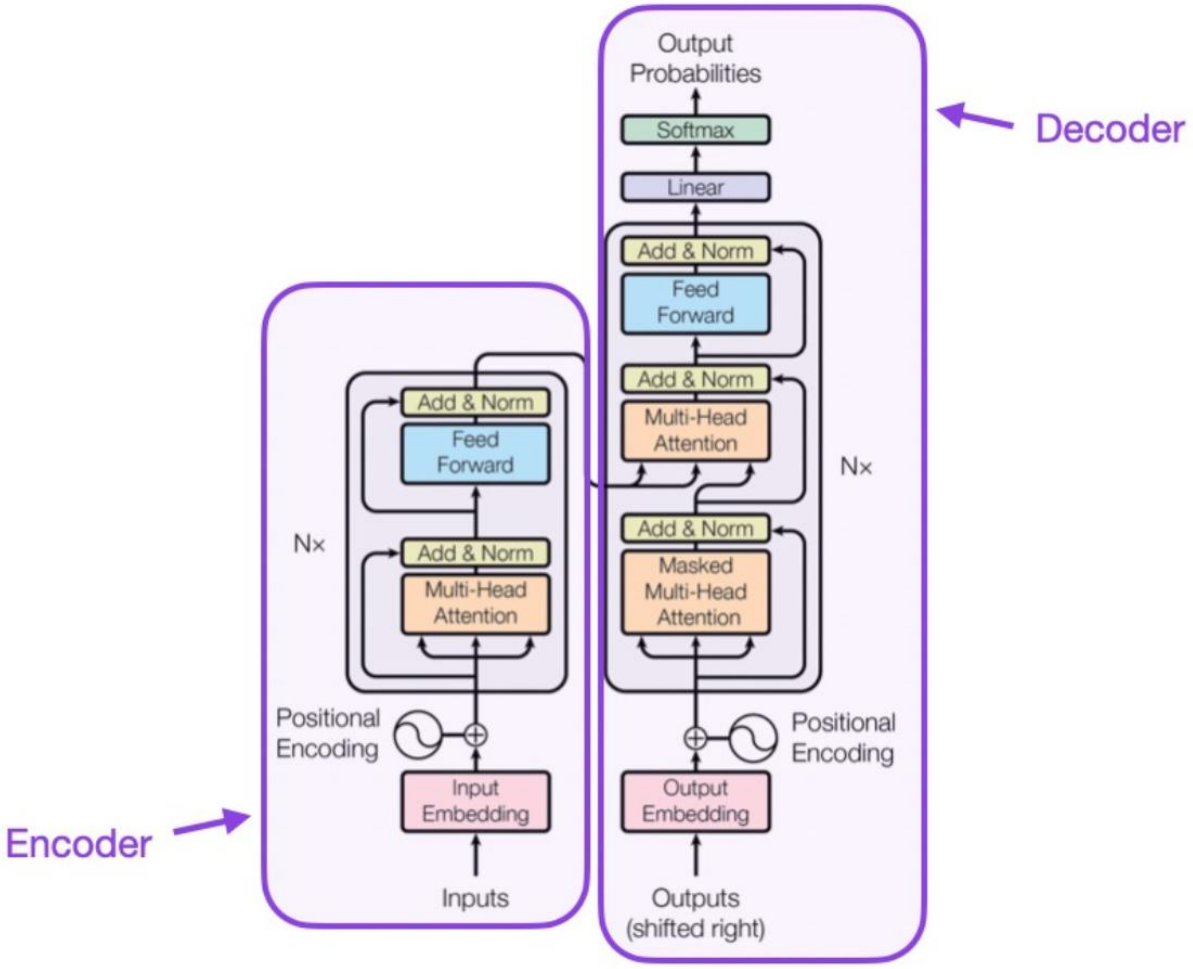


Fig. 2. Model Architecture for Transformer

The architecture for abstractive text summarization starts with the input text, which first goes through a preprocessing stage. In preprocessing, the text is tokenized into smaller units like words or sub words, and common stop words are removed to keep only the meaningful content. This helps in making the summarization process more efficient. After preprocessing, extractive summarization is applied using the TextRank algorithm. This step is about selecting the most important sentences directly from the text based on how important they are.

Once the important sentences are extracted, further cleaning is done by removing duplicate information and improving the coherence among sentences, making the text flow naturally. After that, the refined text is passed into an abstractive summarization model. This is where the Transformer-based T5 model comes in. The T5 model, using an attention mechanism, understands the meaning of the sentences and generates a completely new, shorter version of the original text in its own words instead of just picking sentences. This allows the summary to look more like how a human would explain it.

Finally, the generated summary is evaluated using ROUGE scores, which measure how close the machine-generated summary is to a human-written reference summary. High ROUGE scores mean the summary captures the main ideas well.

The Transformer model used here is built with two main parts: an encoder and a decoder. The encoder processes the input by first embedding the words and adding positional information so the model knows the order of the words. It uses multi-head attention mechanisms to focus on different parts of the text at the same time and applies feed-forward layers to transform the information.

The decoder is responsible for generating the output summary. It also uses embeddings and positional encoding, but it includes masked multi-head attention to make sure it predicts the next word correctly without cheating by looking ahead. It then attends to the encoder outputs to understand the input text better and finally generates the output word by word until the summary is complete.

Overall, this architecture allows the model to produce high-quality abstractive summaries that are fluent, concise, and relevant to the original text.

1.5 HARDWARE AND SOFTWARE REQUIREMENTS

HARDWARE REQUIREMENTS

- Processor: Intel i5 or above
- RAM: Minimum 8 GB
- Storage: 100 GB HDD or SSD
- GPU (Optional but useful): NVIDIA GPU with at least 4 GB VRAM

SOFTWARE REQUIREMENTS

- Operating System: Windows/Linux
- Programming Language: Python 3.8 or above
- Libraries and Tools:
    - TensorFlow / PyTorch
    - NLTK
    - NumPy, Pandas
    - Matplotlib
    - NetworkX (for TextRank)
    - ROUGE Toolkit for evaluation
- IDE: Jupyter Notebook / VS Code

1.6 MODULES AND DESCRIPTION

The proposed system is divided into the following main modules:

1.6.1 DATA COLLECTION

To train and test the summarization model, Telugu news articles were collected from various online sources such as Sakshi, Eenadu, and Andhrajyothy. These articles were manually downloaded or scraped and then saved in a structured format like Excel or CSV files. This helped in easy access and use during processing and training.

1.6.2 PREPROCESSING

The collected Telugu text was cleaned and prepared before summarization. It was first tokenized into sentences and words, then cleaned by removing stop words, punctuation marks, and unnecessary symbols. The clean data was then converted into a proper format that could be used in the summarization model.

1.6.3 EXTRACTIVE SUMMARIZATION USING TEXTRANK

In this stage, each sentence from the text was converted into a vector using ELMo embeddings to capture its meaning. Then, a similarity matrix was built using cosine similarity to understand the relation between sentences. The TextRank algorithm was applied to rank the sentences, and the most important ones were selected as the extractive summary.

1.6.4 ABSTRACTIVE SUMMARIZATION WITH TR

An abstractive summary is generated by using an extractive summary as input, where the Transformer-based T5 model performs abstractive summarization. This step involves using BiLSTM networks for sentence encoding and feature extraction, followed by a Transformer-based T5 model to generate the final summary. T5, a pre-trained transformer model, efficiently handles the task by leveraging its ability to generate fluent and contextually relevant summaries from extractive inputs.

1.6.5 COVERAGE MECHANISM

To avoid repeating the same content in the summary, a coverage mechanism was used. It keeps track of which parts of the text have already been attended to by maintaining a coverage vector.

This helps the model stay focused and avoid redundant information while generating the summary.

1.6.6 EVALUATION

The final module in the system is Evaluation, where the quality and performance of the generated summaries are analyzed. This is done by comparing the system-generated summaries with reference (human-written) summaries using a well-known evaluation toolkit called ROUGE (Recall-Oriented Understudy for Gisting Evaluation).

Evaluation Metrics Used:

- ROUGE-1: Measures the overlap of unigrams (single words) between the system summary and reference summary.
- ROUGE-2: Measures the overlap of bigrams (two-word sequences).
- ROUGE-L: Measures the Longest Common Subsequence (LCS) between the generated and reference summaries.

CHAPTER 2

MERITS AND DEMERITS OF THE BASE PAPER

MERITS:

Hybrid Approach:
Combines both extractive and abstractive methods, which improves the quality of summaries and reduces training time.

LANGUAGE-SPECIFIC MODEL:
Focuses on Telugu, a language that has very limited research and resources in the field of text summarization.

REDUCED REPETITION:
The use of a coverage mechanism helps avoid repeating the same content in the generated summaries**.**

IMPROVED EVALUATION SCORES:
The model achieves better ROUGE-1, ROUGE-2, and ROUGE-L scores compared to existing methods, showing that it is effective.

FASTER TRAINING:
Since only top-ranked sentences are passed into the abstractive model, the training time is significantly reduced.

USE OF ADVANCED NLP TECHNIQUES:
Uses modern deep learning tools like Bi-LSTM, Attention,LLMs, and ELMo embeddings, making the model technically strong and up-to-date.

DEMERITS:

LIMITED DATASET:
The dataset used is relatively small (only 140 Telugu news articles), which may affect the model's performance on a large scale.

SHORT SUMMARY OUTPUT:
The model performs better for short summaries, but struggles with generating longer, more detailed summaries.

LANGUAGE DEPENDENCY:
The model is designed specifically for Telugu, so it might not directly work for other languages without major changes.

REQUIRES GPU FOR TRAINING:
Deep learning models like transformer and attention mechanisms require high computational power, especially for training.

DEPENDENCY ON PREPROCESSING:
The quality of the final summary highly depends on how well the input text is cleaned and pre-processed.

CHAPTER 3

SOURCE CODE

Samples from Source Code

- Loading the dataset and splitting it into training, validation, and testing and Model Setup.

[1]:
```python
import pandas as pd
from sklearn.model_selection import train_test_split

# Load the full dataset (make sure it has "inputs" and "targets" columns)
df = pd.read_csv(r"C:\Users\prudh\OneDrive\Desktop\news_articles_dataset_cleaned.csv")
df_cleaned = df[df['template_id'] != 1]

# Save the cleaned dataset
df_cleaned.to_csv("news_articles_dataset_cleaned_v2.csv", index=False)

print("✅ Cleaned dataset saved as news_articles_dataset_cleaned_v2.csv (without template_id 1)")

# Take only the first 10,000 rows
df_10k = df.iloc[:10000].copy()

# Shuffle and split: 80% train, 10% validation, 10% test
train_df, temp_df = train_test_split(df_10k, test_size=0.2, random_state=42)
valid_df, test_df = train_test_split(temp_df, test_size=0.5, random_state=42)  # split remaining 20% equally

# Save the splits
train_df.to_csv("train_data.csv", index=False)
valid_df.to_csv("valid_data.csv", index=False)
test_df.to_csv("test_data.csv", index=False)

print("✅ Split completed: train_data.csv, valid_data.csv, test_data.csv")
```

```
✅ Cleaned dataset saved as news_articles_dataset_cleaned_v2.csv (without template_id 1)
✅ Split completed: train_data.csv, valid_data.csv, test_data.csv
```

```python
# Load CSV file
train_df = pd.read_csv("train_data.csv")
test_df = pd.read_csv("test_data.csv")
valid_df = pd.read_csv("valid_data.csv")
```

```python
# Model setup
model_name = "ai4bharat/IndicBERTv2-MLM-only"
tokenizer = AutoTokenizer.from_pretrained(model_name)
model = AutoModel.from_pretrained(model_name).eval()
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
model = model.to(device)
```

- Performing Extractive Text Summarization for generating Summary using Text Rank Algorithm.

[5]:
```python
def get_sentence_embeddings(sentences, batch_size=32, max_len=128):
    embeddings = []
    for i in range(0, len(sentences), batch_size):
        batch = sentences[i:i+batch_size]
        encoded = tokenizer(batch, return_tensors="pt", padding=True, truncation=True, max_length=max_len)
        encoded = {k: v.to(device) for k, v in encoded.items()}
        with torch.no_grad():
            output = model(**encoded)
        batch_embeds = output.last_hidden_state.mean(dim=1).cpu().numpy()
        embeddings.extend(batch_embeds)
    return np.array(embeddings)
```

[6]:
```python
def textrank_summary(sentences, embeddings):
    if not sentences:
        return []
    top_k = max(1, int(len(sentences) * 0.70))
    if len(sentences) <= top_k:
        return sentences
    sim_matrix = cosine_similarity(embeddings)
    scores = sim_matrix.sum(axis=1)
    ranked_ids = np.argsort(scores)[::-1]
    selected = sorted(ranked_ids[:top_k])
    return [sentences[i] for i in selected]
```

- Generating extractive summary for the original article.

```python
 #Ensure the summarize_batch function returns summaries correctly
def summarize_batch(df_batch, text_col="Article"):
    summaries = []
    for sentences in df_batch[text_col]:
        if not sentences:
            summaries.append("")
            continue
        if isinstance(sentences, str):
            sentences = [sentences]
        embeddings = get_sentence_embeddings(sentences)
        summary = textrank_summary(sentences, embeddings)
        summaries.append(" ".join(summary))
    return summaries

    #, embeddings)
     #    summaries.append(" ".join(summary))
    return summaries


# Main loop to process DataFrame in batches
def process_dataframe_in_batches(df, text_col="Article", batch_size=100):
    summaries = []
    for i in tqdm(range(0, len(df), batch_size), desc="Batch processing"):
        df_batch = df.iloc[i:i + batch_size]
        batch_summaries = summarize_batch(df_batch, text_col=text_col)
        summaries.extend(batch_summaries)
    df["extractive_summary"] = summaries
    return df
```

```python
def prepare_data(dataframe, tokenizer, batch_size=4):
    # Split data into train and validation
    train_df, val_df = train_test_split(dataframe, test_size=0.2, random_state=42)

    # Create datasets with fixed lengths
    train_dataset = TeluguSummaryDataset(train_df, tokenizer)
    val_dataset = TeluguSummaryDataset(val_df, tokenizer)

    # Create dataloaders
    train_loader = DataLoader(train_dataset, batch_size=batch_size, shuffle=True)
    val_loader = DataLoader(val_dataset, batch_size=batch_size, shuffle=False)

    return train_loader, val_loader
```

Chat (CTRL + I) / Share (CTRL + L)

✓ 0.0s

```python
# 2. Model Training Functions
def initialize_model(model_name="google/mt5-small"):
    tokenizer = T5Tokenizer.from_pretrained(model_name)
    model = T5ForConditionalGeneration.from_pretrained(model_name).to(device)
    return tokenizer, model
```

✓ 0.0s

Text2Text(T5) MODEL

```
Training: 100%|██████████| 1000/1000 [29:31<00:00,  1.77s/it]
Evaluating: 100%|██████████| 250/250 [03:09<00:00,  1.32it/s]
Train Loss: 0.8834, Val Loss: 1.4362
Validation ROUGE Scores: {'rouge1': 0.20259039898010486, 'rouge2': 0.13448029748029744, 'rougeL': 0.20160706564677153}
Epoch 18/25
Training: 100%|██████████| 1000/1000 [29:29<00:00,  1.77s/it]
Evaluating: 100%|██████████| 250/250 [03:11<00:00,  1.31it/s]
Train Loss: 0.8457, Val Loss: 1.4493
Validation ROUGE Scores: {'rouge1': 0.21814561876032462, 'rouge2': 0.138583904983905, 'rougeL': 0.2175622854269913}
Epoch 19/25
Training: 100%|██████████| 1000/1000 [29:30<00:00,  1.77s/it]
Evaluating: 100%|██████████| 250/250 [03:12<00:00,  1.30it/s]
Train Loss: 0.8110, Val Loss: 1.4576
Validation ROUGE Scores: {'rouge1': 0.21332572199042787, 'rouge2': 0.1366340492840493, 'rougeL': 0.21274238865709452}
Epoch 20/25
Training: 100%|██████████| 1000/1000 [29:28<00:00,  1.77s/it]
Evaluating: 100%|██████████| 250/250 [03:11<00:00,  1.31it/s]
Train Loss: 0.7730, Val Loss: 1.4851
Validation ROUGE Scores: {'rouge1': 0.22219085636585634, 'rouge2': 0.14230237540237542, 'rougeL': 0.22160752303252304}
Epoch 21/25
Training: 100%|██████████| 1000/1000 [29:29<00:00,  1.77s/it]
Evaluating: 100%|██████████| 250/250 [03:10<00:00,  1.31it/s]
Train Loss: 0.7422, Val Loss: 1.4895
Validation ROUGE Scores: {'rouge1': 0.21036081141081142, 'rouge2': 0.13639204684204687, 'rougeL': 0.20977747807747807}
Epoch 22/25
Training: 100%|██████████| 1000/1000 [29:29<00:00,  1.77s/it]
Evaluating: 100%|██████████| 250/250 [03:11<00:00,  1.30it/s]
Train Loss: 0.7161, Val Loss: 1.5038
Validation ROUGE Scores: {'rouge1': 0.2151927072927073, 'rouge2': 0.13865142635142633, 'rougeL': 0.214609373959374}
Epoch 23/25
Training: 100%|██████████| 1000/1000 [29:33<00:00,  1.77s/it]
Evaluating: 100%|██████████| 250/250 [03:10<00:00,  1.31it/s]
Train Loss: 0.6980, Val Loss: 1.5088
Validation ROUGE Scores: {'rouge1': 0.21390064380064383, 'rouge2': 0.13661406371406368, 'rougeL': 0.21331731046731048}
Epoch 24/25
Training: 100%|██████████| 1000/1000 [29:30<00:00,  1.77s/it]
Evaluating: 100%|██████████| 250/250 [03:11<00:00,  1.31it/s]
Train Loss: 0.6774, Val Loss: 1.5410
Validation ROUGE Scores: {'rouge1': 0.21561514041514038, 'rouge2': 0.13813325563325562, 'rougeL': 0.21503180708180708}
Epoch 25/25
Training: 100%|██████████| 1000/1000 [29:29<00:00,  1.77s/it]
Evaluating: 100%|██████████| 250/250 [03:12<00:00,  1.30it/s]
Train Loss: 0.6703, Val Loss: 1.5362
Validation ROUGE Scores: {'rouge1': 0.21715958485958486, 'rouge2': 0.13865230325230327, 'rougeL': 0.2165762515262515}
```

TRANING THE MODEL

- Evaluation of model Parameters.

```
model.eval
✓ 0.0s
<bound method Module.eval of T5ForConditionalGeneration(
  (shared): Embedding(250112, 512)
  (encoder): T5Stack(
    (embed_tokens): Embedding(250112, 512)
    (block): ModuleList(
      (0): T5Block(
        (layer): ModuleList(
          (0): T5LayerSelfAttention(
            (SelfAttention): T5Attention(
              (q): Linear(in_features=512, out_features=384, bias=False)
              (k): Linear(in_features=512, out_features=384, bias=False)
              (v): Linear(in_features=512, out_features=384, bias=False)
              (o): Linear(in_features=384, out_features=512, bias=False)
              (relative_attention_bias): Embedding(32, 6)
            )
            (layer_norm): T5LayerNorm()
            (dropout): Dropout(p=0.1, inplace=False)
          )
          (1): T5LayerFF(
            (DenseReluDense): T5DenseGatedActDense(
              (wi_0): Linear(in_features=512, out_features=1024, bias=False)
              (wi_1): Linear(in_features=512, out_features=1024, bias=False)
              (wo): Linear(in_features=1024, out_features=512, bias=False)
              (dropout): Dropout(p=0.1, inplace=False)
              (act): NewGELUActivation()
...
    (final_layer_norm): T5LayerNorm()
    (dropout): Dropout(p=0.1, inplace=False)
  )
  (lm_head): Linear(in_features=512, out_features=250112, bias=False)
)>
```

- A robust evaluation function was implemented to compare predicted and reference summaries using precision, recall, and F1-score.

```python
from indicnlp.tokenize.indic_tokenize import trivial_tokenize
from collections import Counter

def compute_metrics(reference, predicted, lang='te'):
    # Tokenize summaries (use indicnlp for Telugu, or split for English)
    if lang == 'te':
        ref_tokens = trivial_tokenize(reference, lang='te')
        pred_tokens = trivial_tokenize(predicted, lang='te')
    else:
        ref_tokens = reference.lower().split()
        pred_tokens = predicted.lower().split()

    # Convert to sets for overlap
    ref_counter = Counter(ref_tokens)
    pred_counter = Counter(pred_tokens)

    # Compute TP, FP, FN
    tp = sum((ref_counter & pred_counter).values())  # Intersection
    fp = sum((pred_counter - ref_counter).values())  # Predicted but not in reference
    fn = sum((ref_counter - pred_counter).values())  # Reference but not in predicted

    # Handle edge cases
    precision = tp / (tp + fp) if (tp + fp) > 0 else 0.0
    recall = tp / (tp + fn) if (tp + fn) > 0 else 0.0
    f1 = (2 * precision * recall) / (precision + recall) if (precision + recall) > 0 else 0.0

    return {
        'precision': precision,
        'recall': recall,
        'f1_score': f1,
        'tp': tp,
        'fp': fp,
        'fn': fn
    }
```

- Evaluating the quality of generated summaries using ROUGE-1, ROUGE-2, and ROUGE-L metrics by comparing them with the reference (gold) summaries.

```python
import evaluate

# Load ROUGE metric
rouge = evaluate.load("rouge")

# Convert Series to lists
predictions = test['generated'].tolist()
references= test['summary'].tolist()

# Compute ROUGE
results = rouge.compute(predictions=predictions, references=references)

# Print results
for key, value in results.items():
    print(f"{key}: {value*num:.4f}")
```

CHAPTER 4

SNAPSHOTS

Providing with a sample document and producing a summary

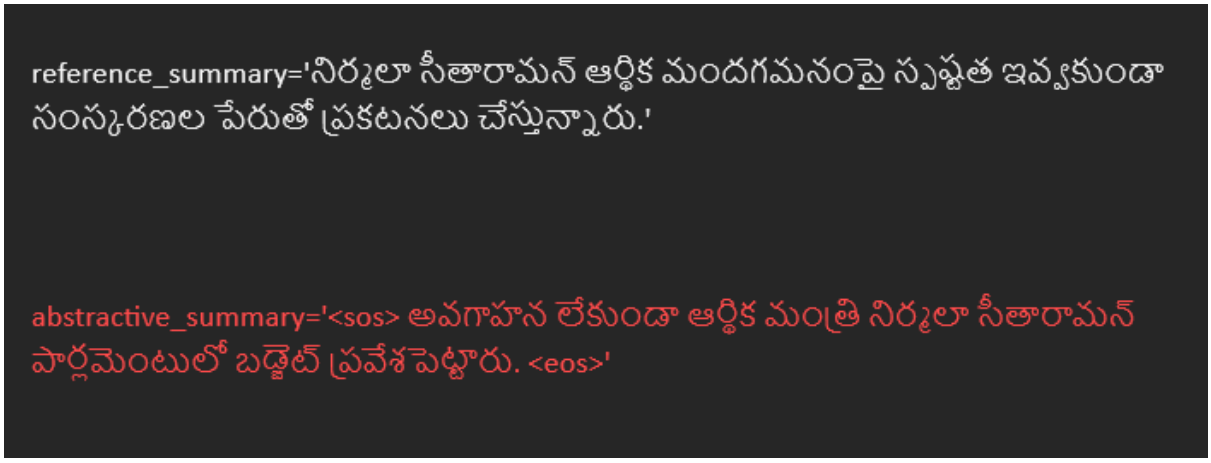
Sample document chosen from the dataset as an input article

```
original_article='నిర్మలా సీతారామన్ గృహ వాహన రుణాలపై వడ్డీ భారం తగ్గించేందుకు చర్యలు తీసుకుంటున్నాం', 'శుక్రవారం సాయంత్రం దిల్లీలో విలేకరుల సమావేశంలో మాట్లాడిన ఆమె దేశ ఆర్థిక పరిస్థితికి సంబంధించి అనేక అంశాలను వివరించారు', 'గత అయిదేళ్లుగా సంస్కరణలను అమలు చేస్తున్నామని', 'సంస్కరణలనేవి నిరంతర ప్రక్రియని చెప్పారు', 'ఇప్పటికే వాణిజ్యంలో పన్ను విధానాల్లో ఎన్నో సంస్కరణలు తీసుకొచ్చినట్లు చెప్పారు', 'ఇంకా ఏమేం చెప్పారు జీఎస్టీ మరింత సులభతరం చేస్తాం', 'దీనిపై ఆగస్టు 25న అధికారులతో సమావేశం నిర్వహిస్తున్నాం', 'పన్నుల వసూళ్లలో ఎవరికీ ఇబ్బందులు ఉండవు', 'వాణిజ్య యుద్ధం ప్రభావం ప్రపంచ దేశాలపైనా ఉంది ప్రపంచ జీడీపీ 3', '2శాతం నుంచి మరింత పతనమవుతోంది', '2014 నుంచి మేం తీసుకొచ్చిన సంస్కరణలతో భారత్ సురక్షిత స్థితి ఉంది', '2014 నుంచి సంస్కరణలే అజెండాగా పనిచేస్తున్నాం', 'రెపో రేట్లకు అనుగుణంగానే గృహ వాహన రుణాలపై భారం తగ్గనుంది', 'మార్కెట్లో రూ', '5 లక్షల కోట్ల ద్రవ్య లభ్యతకు ఏర్పాట్లు చేస్తున్నాం', 'ప్రభుత్వ రంగ బ్యాంకులకు రూ', '70 వేల కోట్లు ఆర్థిక సర్దుబాటు చేస్తాం', 'వడ్డీ రేట్లు తగ్గించేందుకు చర్యలు చేపడతాం', 'ఆ తగ్గింపు రుణ గ్రహీతలకు చేరేలా చర్యలు తీసుకుంటాం', 'ఎంఎస్ఈలను బలోపేతం చేయాలన్నది మా ప్రభుత్వ లక్ష్యం', 'వ్యాపారులు పారిశ్రామికవేత్తలను విచారించాలనేది ప్రభుత్వ ఉద్దేశం కాదు', 'ఆర్థిక అవకతవకలను సహించం', 'భారీ జరిమానాలు విధిస్తాం', 'సీఎస్ఆర్ ఉల్లంఘనలను క్రిమినల్ నేరాలుగా పరిగణించం', 'అక్టోబర్ 1 నుంచి కేంద్రీకృత విధానంలో ఆదాయ పన్ను నోటీసులు ఇస్తాం', 'నోటీసులు అందిన మూడు నెలల్లోనే అన్ని కేసులు పరిష్కారమవుతాయి', 'డీఎన్ఐ లేని నోటీసులకు సమాధానాలు చెప్పాల్సిన అవసరం లేదు', 'దేశీయ విదేశీ ప్రత్యక్ష ఈక్విటీ పెట్టుబడులపై బడ్జెట్ ముందునాటి విధానం పునరుద్ధరిస్తాం', '2020 మార్చి వరకు కొనుగోలు చేసిన బీఎస్4 రకం వాహనాల జీవిత కాలం ఎంతవరకు ఉందో అంతవరకు తిప్పే అవకాశం ఉంది', 'అన్ని శాఖల్లో పాత వాహనాల స్థానంలో కొత్తవి తీసుకోమని కోరుతాం', 'పాత వాహనాల విషయంలో త్వరలో విధానాన్ని ప్రకటిస్తాం', 'కాంగ్రెస్ విమర్శలు ఆర్థిక మంత్రి ప్రసంగం అనంతరం విపక్ష కాంగ్రెస్ విమర్శలు సంధించింది', 'బ్యాంకులకు రూ', '70 వేల కోట్ల రీక్యాపిటలైజేషన్పై సందేహాలు లేవనెత్తింది', 'ఇది మధ్యతరగతి పన్ను చెల్లింపుదారులపై భారం కాదా గత రీక్యాపిటలైజేషన్ ఎలాంటి ఫలితాలనిచ్చింది', 'ద్రవ్య స్థితిని ఇది చక్కదిద్దుతుందా అంటూ ట్వీటర్ వేదికగా ప్రశ్నలు వేసింది', 'అంతేకాదు', 'ప్రపంచవ్యాప్తంగా ఆర్థిక మందగమనం ఉందని ఆర్థిక మంత్రి చెబుతున్నారు కానీ నోట్లరద్దు జీఎస్టీ చేసిన నష్టాల గురించి కావాలనే మాట్లాడడం లేదని కాంగ్రెస్ పార్టీ విమర్శలు చేసింది', 'కాంగ్రెస్ అధికార ప్రతినిధి రణదీప్ సుర్జేవాలా జీఎస్టీ అమల్లోకొచ్చిన మూడేళ్ల తరువాత కూడా రాబడిలో ఇంకా భారీ లోటు తప్పడం లేదని ట్వీట్ చేశారు'
```

Model generating Extractive Summary for the above document

```
extractive_summary='నిర్మలా సీతారామన్ గృహ వాహన రుణాలపై వడ్డీ భారం తగ్గించేందుకు చర్యలు తీసుకుంటున్నాం గత అయిదేళ్లుగా సంస్కరణలను అమలు చేస్తున్నామని సంస్కరణలనేవి నిరంతర ప్రక్రియని చెప్పారు ఇప్పటికే వాణిజ్యంలో పన్ను విధానాల్లో ఎన్నో సంస్కరణలు తీసుకొచ్చినట్లు చెప్పారు ఇంకా ఏమేం చెప్పారు జీఎస్టీ మరింత సులభతరం చేస్తాం దీనిపై ఆగస్టు 25న అధికారులతో సమావేశం నిర్వహిస్తున్నాం పన్నుల వసూళ్లలో ఎవరికీ ఇబ్బందులు ఉండవు 2శాతం నుంచి మరింత పతనమవుతోంది 2014 నుంచి మేం తీసుకొచ్చిన సంస్కరణలతో భారత్ సురక్షిత స్థితి ఉంది 2014 నుంచి సంస్కరణలే అజెండాగా పనిచేస్తున్నాం రెపో రేట్లకు అనుగుణంగానే గృహ వాహన రుణాలపై భారం తగ్గనుంది 5 లక్షల కోట్ల ద్రవ్య లభ్యతకు ఏర్పాట్లు చేస్తున్నాం 70 వేల కోట్లు ఆర్థిక సర్దుబాటు చేస్తాం వడ్డీ రేట్లు తగ్గించేందుకు చర్యలు చేపడతాం ఆ తగ్గింపు రుణ గ్రహీతలకు చేరేలా చర్యలు తీసుకుంటాం ఎంఎస్ఈలను బలోపేతం చేయాలన్నది మా ప్రభుత్వ లక్ష్యం ఆర్థిక అవకతవకలను సహించం భారీ జరిమానాలు విధిస్తాం సీఎస్ఆర్ ఉల్లంఘనలను క్రిమినల్ నేరాలుగా పరిగణించం అక్టోబర్ 1 నుంచి కేంద్రీకృత విధానంలో ఆదాయ పన్ను నోటీసులు ఇస్తాం నోటీసులు అందిన మూడు నెలల్లోనే అన్ని కేసులు పరిష్కారమవుతాయి డీఎన్ఐ లేని నోటీసులకు సమాధానాలు చెప్పాల్సిన అవసరం లేదు దేశీయ విదేశీ ప్రత్యక్ష ఈక్విటీ పెట్టుబడులపై బడ్జెట్ ముందునాటి విధానం పునరుద్ధరిస్తాం అన్ని శాఖల్లో పాత వాహనాల స్థానంలో కొత్తవి తీసుకోమని కోరుతాం పాత వాహనాల విషయంలో త్వరలో విధానాన్ని ప్రకటిస్తాం 70 వేల కోట్ల రీక్యాపిటలైజేషన్పై సందేహాలు లేవనెత్తింది ఇది మధ్యతరగతి పన్ను చెల్లింపుదారులపై భారం కాదా గత రీక్యాపిటలైజేషన్ ఎలాంటి ఫలితాలనిచ్చింది'
```

Model’s predicted abstractive summary and generated Abstractive Summary for the above document



When we chose to compare the model’s performance against existing systems, the following results are obtained: -

Evaluation Metric	Model	F1 Score	Precision	Recall
ROUGE-1	Seq2seq with Attention + Coverage	0.382	0.363	0.393
	T5-small	0.500	0.500	0.500
ROUGE-2	Seq2seq with Attention + Coverage	0.162	0.185	0.158
	T5-small	0.000	0.000	0.000
ROUGE-L	Seq2seq with Attention + Coverage	0.371	0.353	0.370
	T5-small	0.500	0.500	0.500

Table1: ROUGE scores Comparison Table

Proposed Model vs Base Paper Model

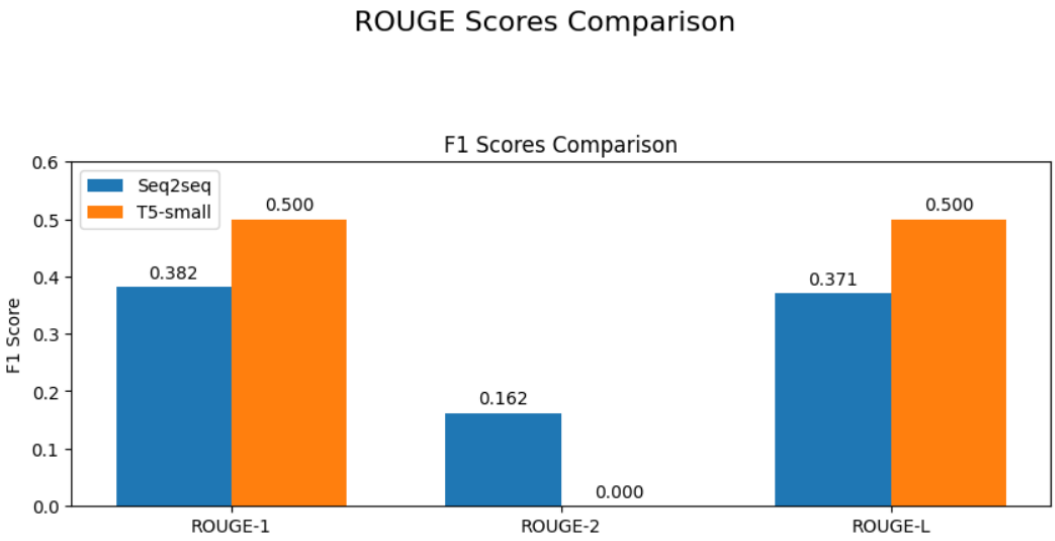
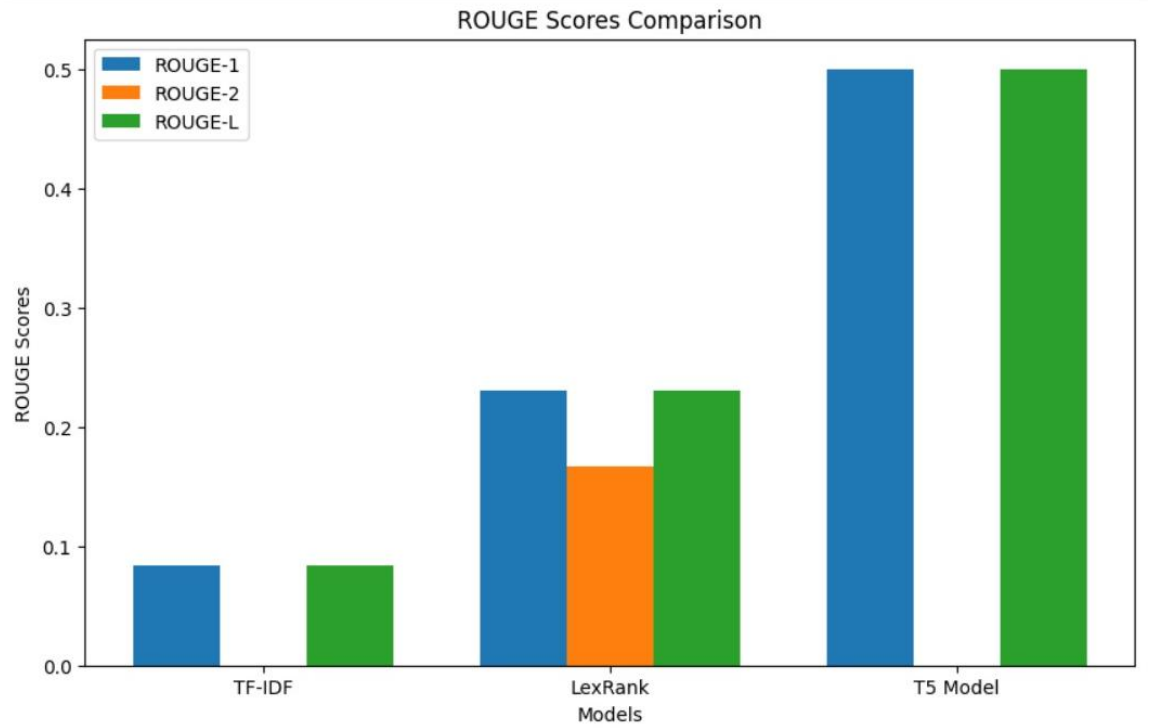


Fig 2: ROUGE Scores Comparison Graph



CHAPTER 5

CONCLUSION AND FUTURE PLANS

CONCLUSION:

In this project, a hybrid text summarization model was developed for the Telugu language using both extractive and abstractive techniques. The extractive part employs the TextRank algorithm to identify the most important sentences, while the abstractive part utilizes a Bi-LSTM-based Transformer T5 model. T5's attention mechanism and pre-trained capabilities enable the generation of more meaningful and coherent summaries.

The proposed method reduces training time while maintaining good summary quality. The results demonstrated improved performance, particularly in terms of ROUGE evaluation scores, compared to existing models, highlighting the effectiveness of combining both summarization techniques for Telugu text summarization.

FUTURE PLANS:

- In the current model, short summaries are generated effectively, but the performance drops for long summaries. In the future, the model can be improved to handle longer summaries more efficiently.
- The dataset used was limited. Future work can involve collecting larger and more diverse datasets from various sources to improve accuracy and generalization.
- The current model is focused on Telugu language only. Future versions can aim to build multilingual summarizers or extend the same model to support other Indian regional languages.
- Optimization techniques such as transformer-based models like BERT could be explored to enhance the abstractive summarization part.

CHAPTER 6

REFERENCES

1. Mihalcea, Rada, and Paul Tarau. "TextRank: Bringing order into texts." *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2004, pp. 404–411.

2. Lin, Chin-Yew. "ROUGE: A package for automatic evaluation of summaries." *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*, 2004.

3. Erkan, Günes, and Dragomir R. Radev. "LexRank: Graph-based lexical centrality as salience in text summarization." *Journal of Artificial Intelligence Research*, vol. 22, 2004, pp. 457–479.

4. Zhang, Peng, and Chunxia Li. "Automatic text summarization based on sentences clustering and extraction." *2009 2nd IEEE International Conference on Computer Science and Information Technology*, Beijing, 2009, pp. 167–170.

5. Wong, Kam-Fai, Ming Wu, and Wenjie Li. "Extractive summarization using supervised and semi-supervised learning." *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, 2008, pp. 985–992.

6. Kallimani, J. S., K. G. Srinivasa, and B. Eswara Reddy. "Information extraction by an abstractive text summarization for an Indian regional language." *7th International Conference on Natural Language Processing and Knowledge Engineering*, Tokushima, Japan, 2011, pp. 319–322.

7. Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio. "Neural machine translation by jointly learning to align and translate." *arXiv preprint arXiv:1409.0473*, 2014.

8. Rush, Alexander M., Sumit Chopra, and Jason Weston. "A neural attention model for abstractive sentence summarization." *arXiv preprint arXiv:1509.00685*, 2015.

9. Khanam, M. H. "Text summarization for Telugu document." *Journal of Computer Engineering*, 2016, pp. 25–28.

10. Shashikanth, S., and S. Sanghavi. "Text summarization techniques survey on Telugu and foreign languages." *International Journal of Research in Engineering, Science and Management* 2.1 (2019): 211–213.

11. Tomer, Mohit, and Manoj Kumar. "Improving text summarization using ensembled approach based on fuzzy with LSTM." *Arabian Journal for Science and Engineering*, vol. 45, no. 12, 2020, pp. 10743–10754.

12. Elayeb, Bilel, et al. "Automatic Arabic text summarization using analogical proportions." *Cognitive Computation*, vol. 12, no. 5, 2020, pp. 1043–1069.

13. Sudha, D. N., and Y. M. Latha. "Multi-document abstractive text summarization through semantic similarity matrix for Telugu language." *International Journal of Advanced Science and Technology*, vol. 29, no. 1, 2020, pp. 513–521.

14. Shashikanth, S., and S. Sanghavi. "Text summarization techniques survey on Telugu and foreign languages." *International Journal of Research in Engineering, Science and Management*, vol. 2, no. 1, 2019, pp. 211–213.

15. Alzuhair, Abdullah, and Mohammad Al-Dhelaan. "An approach for combining multiple weighting schemes and ranking methods in graph-based multi-document summarization." *IEEE Access*, vol. 7, 2019, pp. 120375–120386.

16. Hernandez-Castaneda, A., et al. "Extractive automatic text summarization based on lexical-semantic keywords." *IEEE Access*, vol. 8, 2020, pp. 49896–49907.

17. Anand Babu, G. L., and Srinivasu Badugu. "Multi-document hybrid text summarization with bi-LSTM RNN for Telugu language." *Sādhanā*, vol. 49, 2024, pp. 1–12.

CHAPTER 7

APPENDIX – BASE PAPER

Title

Multi-document Hybrid Text Summarization with Bi-LSTM RNN for Telugu Language

Authors

G. L. Anand Babu and Srinivasu Badugu
Department of Information Technology, Anurag University, Hyderabad
Department of Computer Science and Engineering, Stanley College of Engineering and Technology for Women, Hyderabad

Journal & Publication Info

Journal: Sādhanā – Springer, Indian Academy of Sciences
Accepted: 11 February 2024
DOI: 10.1007/s12046-024-02499-8

https://doi.org/10.1007/s12046-024-02499-8

Multi-document hybrid text summarization with bi-LSTM RNN for Telugu language

G L ANAND BABU[1,*] and SRINIVASU BADUGU[2]

[1] Department of Information Technology, Anurag University, Hyderabad, Telangana 500088, India
[2] Department of Computer Science and Engineering, Stanley College of Engineering and Technology for Women, Abids, Hyderabad, Telangana 500001, India e-mail: anandbabu77@gmail.com; srinivasucse@gmail.com

MS received 13 May 2022; revised 2 August 2023; accepted 11 February 2024

Abstract. One of the most popular south Indian languages in India is the Telugu language which is currently spoken by 84 million native Telugu speakers in Andhra Pradesh and Telangana. With the rapid growth of the Telugu digital content, the need for the automatic text summarizer is arisen to provide short text from huge text documents. Extractive text summarization model generates only significant sentences. Abstractive text summarization method requires more training time. In this paper, a novel hybrid model is proposed for generating text summaries by combining extractive and abstractive approach to reduce the training time. For extractive method TextRank algorithm is utilized and for abstractive method attention-based sequence to sequence model with bidirectional long short-term memory (Bi-LSTM) is utilized. Moreover, coverage mechanism is included into the proposed hybrid approach to reduce the repetition in summaries and to improve the quality of summaries. The performance of the proposed hybrid model is evaluated by the ROUGE toolkit in terms of F-measure, recall and precision. The results of the proposed model are compared with other existing models which shows that the proposed hybrid model outperforms other existing text summarization models for Telugu Language.

Keywords. TextRank algorithm; sequence to sequence model; bidirectional long short-term memory (BiLSTM); attention mechanism; coverage mechanism; ROUGE.

1. Introduction

Nowadays the need for Telugu text summarization is grown tremendously because of the huge increase in the amount of online data in Telugu Language. The size of the document text size is reduced by building a summary that has the most important data in that document to give a better understanding of a lot of data in a very short time using an automatic summarizer [1]. The text summarization approaches can be categorized into two: extractive summarization and abstractive summarization [2]. The extractive models extract the top ranked sentences to summarize the input text. In contrast to extractive summarization, abstractive models generate summaries using new sentences that don't present in the input text [3]. The demerit of extractive model is that it only chooses the significant sentences of the input text. The abstractive summarization is better as it rewords the sentences. The demerit of abstractive model is that for training it wants enormous data and it takes more training time. If the significant sentences selected from extractive model given as the input text for abstractive summarization can eliminate the demerits of the two methodologies. So that we can generate abstractive summary with less training time [4].

For extractive summarization, important sentences are identified with different approaches such as sentences clustering [5], supervised approaches [6] and graph-based methods [7]. Since the graph-based ranking method is one of the important extractive text summarization approaches, that method is utilized in this paper for text summarization. Through the years, many important works are proposed for graph-based ranking method such as LexRank [8] and TextRank [9]. The popular TextRank algorithm is employed for graph-based ranking method in our work. Due to the advancement in technology, the research in abstractive summarization has gained more attraction in recent years. The traditional rule-based approaches give poor performance in the abstractive text summarization and

*For correspondence
Published online: 25 April 2024

advanced deep learning technique with the use of sequenceto-sequence models have improved the performance of text summarization [10]. From the recent studies we can know that sequence to sequence models can suffer from semantic irrelevance, repetition and grammatical errors [11]. In order to address the undesirable behaviour of sequence to sequence model and for improving the quality of text summarization, in this paper some mechanisms such as attention mechanism [12, 17] and coverage mechanism [13] are employed. Though these mechanisms can improve the quality of text summarization, the training time is higher. So that, a hybrid approach which combines the extractive and abstractive model is proposed in our work to improve the efficiency of text summarization. The extractive models generate grammatically correct and coherent summary, while the abstractive models have more abilities like generalization and paraphrasing [4]. When combining these two models, only the top ranked sentences from the input text by extractive model is given as the input to the encoder in the abstractive model. Thus, the hybrid architecture reduces the training time by shortening the encoder input and also this hybrid technique combines the benefits of both extractive and abstractive models.

In this paper, a novel hybrid model is proposed for generating text summaries by combining extractive and abstractive approach to reduce the training time and improve the performance of summarizer. The proposed hybrid model is applied for Telugu text summarization. Text summarization works are very few for the Telugu language compared to other languages and the summarization focused mainly on extractive approaches. In this work, a novel hybrid model is proposed for Telugu text summarization which is a fusion of extractive and abstractive approach. In abstractive model, Bi-LSTM encoder and beam search for inference with a unidirectional decoder is utilized. For updating network weights Bi-LSTM employs Adam optimizer and attention mechanism. In our proposed hybrid approach, first the extractive model with TextRank algorithm is utilized for identify and choose the top ranked sentences then these selected sentences are given as the input to the abstractive model to produce the finalized summary. Thus, the proposed method combines the advantages of both summarization approaches to enhance the speed, quality and stability of the proposed Telugu text summarizer model. The performance of our hybrid text summarization model is evaluated with the ROUGE toolkit [14]. Experimental results show that the proposed hybrid method is effective for the Telugu text summarization.

The main aim of this work is to combine the merits of both extractive and abstractive methods for the Telugu text summarization to reduce the training time. In this model, TextRank algorithm is utilized for the extractive summarization and Bi-LSTM neural network is utilized for abstractive summarization. In order to improve the summarization efficiency and reduce the repetition problem, attention mechanism and coverage mechanism are also integrated. The rest of this paper is arranged as follows. Section 2 details the related works in the area of extractive and abstractive text summarization for Telegu document. Section 3 explains the proposed methodology. Experiments are provided in Section 4. In Section 5, the results of experiments and discussion are given. Section 6 concludes this paper with future work.

2. Related work

When compared to other languages such as English works on automatic Telugu text summarization are relatively few. Many of the works mainly focused on the extractive approach. This section will discuss the extractive and abstractive approaches and the related works in Telugu language.

2.1 Text summarization approaches

The text summarization methods are mainly classified into two types: extractive summarization and abstractive summarization. In the extractive summarization the sentences are selected from the original input text. But in the abstractive method the generated summary contains new sentences which are not present in the original text. For extractive summarization, important sentences have been identified with different approaches such as sentences clustering, supervised approaches and graph-based methods. Since the graph-based ranking method is considered as one of the important extractive text summarization approaches, that method is utilized in our proposed work for text summarization. Through the years, many important works are proposed for graph-based ranking method such as LexRank [8] and TextRank [9]. The popular TextRank algorithm is employed for graph-based ranking method in our work.

For abstractive text summarization, the deep learning model gives better results than the rule-based method. The sequence-to-sequence method have been mainly used as the deep learning model for abstractive text summarization. To solve the sequence to-sequence tasks, the encoder-decoder model is utilized by Sutskever et al [15] where the sequences of input and output are of different lengths. All the necessary information of the input text is need to be compressed into a fixed-length vector by the neural network in the encoder-decoder approach, so that it is difficult to summarize long sentences using neural network and deteriorates the performance of the basic encoder-decoder as the increase in the length of an input sentence. So that, to improve the performance of the basic encoder-decoder attention mechanism is proposed by Bahdanau et al [16], which encodes the source sentences into a fixed-length vector and then decodes it into the target language, have the limitation in handling long sentence. It also improves the extension to the neural mechanism translation model, which allows the model to automatically search for parts of the source sentence that are relevant to predicting the target word, without having to segment the source sentence explicitly.

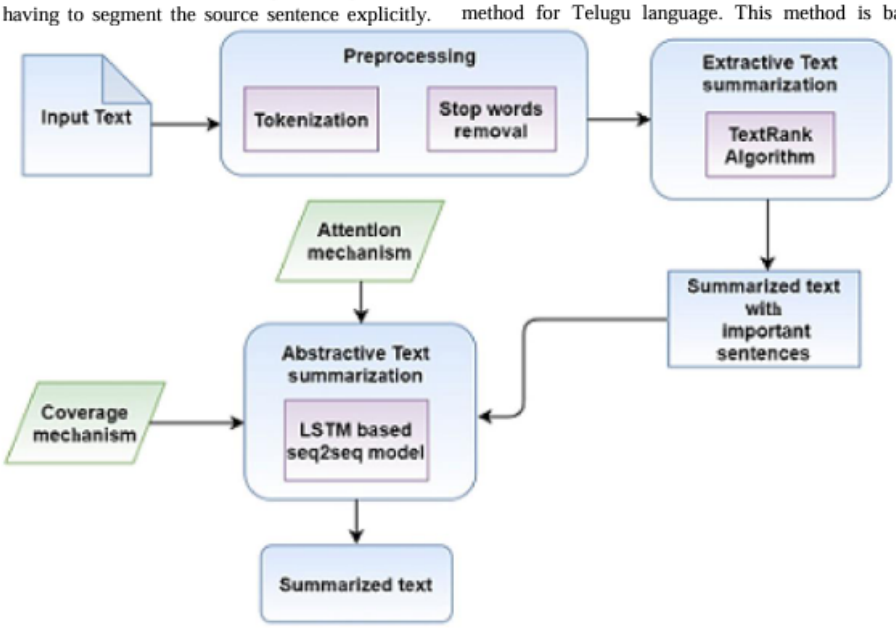


Figure 1. Flowchart for proposed hybrid text summarization process.

This attention-based model was improved by Rush et al [17] to overcome the limitation of the previous paper by proposing an abstractive sentence summarization, which is a task that aims to generate a concise and fluent summary of a longer sentence, preserving its main meaning and information. It dynamically focuses on different parts of the input sentence while generating the summary. It also handles out-of-vocabulary words and rare words. Then the coverage mechanism is integrated into this attention-based sequence-to-sequence model by See et al [18] to avoid the repetition in summaries. It uses a hybrid pointer-generator network that can copy words from the source text via pointing, which aids the accurate reproduction of information while retaining the ability to produce novel words through the generator. Then, it uses coverage to keep track of what has been summarized, which discourages repetition.

Thus, in our work, we have proposed a hybrid method combining extractive and abstractive approaches with LSTM.

2.2 Text summarization approaches in the Telugu language

Many of work in the Telugu text summarization has been performed based on the extraction-based text summarization. Only a few works have been achieved on the abstractive text summarization for Telugu Language. Khanam et al [19] presented the extractive summarization method for Telugu language. This method is based on ranking system. In the pre-processing of Telegu text, the text is tokenized, tagged and the stop words are removed. After that, the frequency of the words is calculated and low frequency sentences are removed. The sentences are ranked based on the frequency and depending on the rank, the summary is generated.

Shashikanth and Sanghavi [20] also proposed an extractive summarization method for Telugu language with k-means clustering. This method is based on ranking system. In the pre-processing of Telegu text, the text is tokenized, tagged and the stop words are removed. After that, the frequency of the words is calculated and clusters are formed with the sentences. The sentences are ranked based on the frequency and depending on the rank, low frequency sentences are removed and the summary is generated by selecting sentences from cluster.

Kallimani et al [21] discussed the different approaches for abstractive text summarization approach in Telugu Language. In the pre-processing of Telegu text, the text is tokenized, part of speech tagged and the stop words are removed. After pre-processing, the keywords are extracted using the TFIDF method. After that the important sentences are selected based on the score. The final summary is generated with the extracted sentences and the text is post processed by summary refinement and rephrasing approaches.

Sudha and Latha [22] proposed a multi document abstractive text summarization approach for Telugu language with semantic similarity matrix. The similarity is calculated with RNN, Jiang similarity and semantic concepts. For measuring the performance of model, the ROUGE score is utilized. From these reviews we can know, most of the works were mainly focused on the extractive text summarization for Telugu language. Only a few works were addressed the abstractive text summarization with Telugu text. Our novel proposed approach combines both the extractive and abstractive model to summarize the Telugu text. To the best of my knowledge no other hybrid text summarization works have been done for the Telugu Language.

3. Proposed Bi-LSTM-based hybrid textsummarization

This section proposed a hybrid text summarization model for Telugu language. The proposed hybrid model combines both the abstractive and extractive model for efficient document summarization with reduced training time. The flowchart of the proposed hybrid text summarization process is shown in figure 1.

In the first step the input text in the Telugu language is pre-processed. Pre-processing is the initial phase for text summarization. The input text is pre-processed to filter out useless data. In the pre-processing step the input text is split into individual sentences and then individual sentences are split into words using tokenization. Then the stop words are removed from the tokenized words. The proposed text summarization approach is a hybrid of two main text summarization methods (i) Extractive summarization using graph-based algorithm and (ii) Abstractive summarization using deep learning model. The pre-processed text is then summarized using the extractive summarization approach. For extractive summarization, graph-based Bi-LSTM model with TextRank algorithm is employed which selects the top-ranked sentences which is further given as the input to the sequence to sequence model with LSTM to produce the final abstractive summary.

3.1 Extractive text summarization using text rank algorithm

Obtaining the most important sentences from a document as an abstract of that document is known as Extractive text summarization. TextRank algorithm is used to find these 'important' sentences in our approach which is a graphbased ranking model. For each and every sentence we will find vector representation and then the mean of those vectors using the Elmo embedding. To obtain a single vector that contains all the information of the token, Elmo embedding uses a method called the Deep Bi-LSTM layer. This method has a higher and lower layer that captures different aspects of the token. The higher layer focuses on the context-dependent aspects, while the lower layer focus on the syntax aspect. The word representations are obtained by adding the weighted network representations. Each network represents in the intermediate layers contributes to the word representations with a different weight as shown in Equation (1)

$$ELMO_k^{task} = \gamma E(R_{k;h}) \Rightarrow \gamma c^{task} \sum_{j=0}^{L} s_j^{task} h_{kj} \quad (1)$$

s^{task}_j is represented as soft-max normalized weight, c^{task} is represented as the scalar parameter. c is of practical importance to aid the optimization process. To find the importance of each sentence these vectors are utilized. The sentence vectors are utilized to find the similarity matrix. Using the cosine similarity approach, a similarity matrix is prepared which depends on the sentence vector. Cosine similarity is a way of measuring how two similar vectors are, whether they represent words, sentences, or documents. It uses the cosine function to calculate the similarity. For instance, if A and B are two vectors that we want to compare, their cosine similarity is defined in the Equations (2) to (4)

$$\cos(h) = \gamma A:B \quad (2)$$

$$\gamma \frac{A:B}{(3) \, jjAjj:jjBjj} \quad (3)$$

$$P_n^{i=1} A_i B_i \quad (4)$$

¼ pffiffiffiffiffiffiffiffiffiffiffiffiffiffiffiffiffiffiffiP n A2i

pffiffiffiffiffiffiffiffiffiffiffiffiffiffiffiffiffiffiffiP ni¼1 B2i ð4Þ

i¼1

The angle between A and B is calculated by the measure in equation (2). The vector have no match when the cosine value is 0, which means they are perpendicular to each other. The match is greater when the value is closer to 1, which means the angle is smaller. At first an empty similarity matrix is made with dimensions (n * n) and this matrix is initialized with cosine similarities of the sentences. To calculate the sentence rank, a graph is generated from the similarity matrix, with similarity scores as edges and sentences as vertices. TextRank algorithm is applied on this graph to get the rankings of sentences. Starting from arbitrary values assigned to each node in the graph for constructing the edges proportional with the number of vertices, this step is finding the scores of each sentence until convergence by iterating the algorithm. The sentences are arranged in descending order based on their scores. The text summary is generated with first k sentences which have high scores. The figure 2 shows the flowchart of the extractive text summarization process. The selected sentences are given as the input text to the abstractive text summarization model with LSTM neural network to produce the summary.

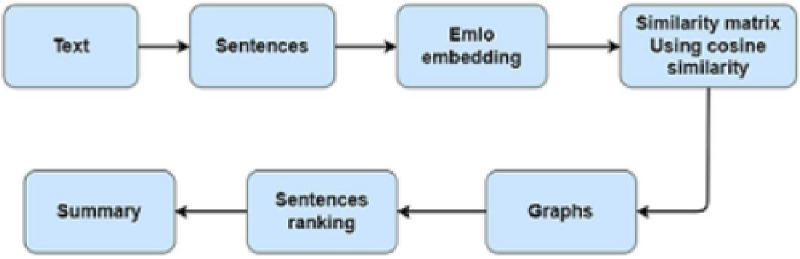


Figure 2. Flowchart of extractive text summarization process.

3.2 Abstractive text summarization with sequenceto-sequence attention model

For the abstractive text summarization, attention-based sequence-to-sequence model is utilized in this paper. This model utilizes beam search decoder, Bi-LSTM encoder and the Attention Mechanism with weight normalization. The word with highest probability is selected as the output by the beam search decoder. The input of the decoder is a single vector with all the information about the context. Generally, exactly process the long input sequences by the standard sequence to sequence model is unable to do, since only last hidden state of the encoder is utilized as the input for the decoder. In the attention mechanism this issue is solved by utilizing all the hidden states of the encoder as the input to the decoder. This can be achieved by generating a particular mapping between the decoder output's each time step to all the hidden states of encoder.

The encoding model consists of two different parts: embedding layer and RNN layers. The higher representative power for the words is provided by the embedding layer which is the first part of the model. The representation of each word in a sentence is given by the number of features. To initialize word-embeddings, Glove pre-trained vectors are utilized. In the second part of model, several bidirectional RNN layers are stacked. The combined outputs of backward and forward layer are utilized as the input of the next layer. There are two processes in the decoding model: inference and training. To improve the performance of the model, Adam Optimizer is utilized.

The tokens of the text t_i are given as the input to the encoder, producing an encoder hidden states'sequence h_i. The decoder receives the previous word's word embedding and has decoder state s_t on each step t. when the decoder generates a new word, it needs to focus on some words in the source text that are relevant, which was done by the attention distribution a^t. It is a probability distribution over the source words. The attention distribution a^t is computed for each timestep t of the decoder as defined in Equation (5) & (6)

$$e_{ti} ¼ v_T tanhðW_h h_i þ W_s s_t þ b_{attn}Þ \quad ð5Þ$$

$$a^t ¼ softmaxðe^tÞ \quad ð6Þ$$

v, W_h, W_s and b_{attn} represents learnable parameters. The decoder uses the attention distribution a^t to decide how much attention to pay to each word in the source text when it produce a new word. the attention distribution a^t is a probability distribution over the source words, and it is learned for each timestep t of the decoder using the learnable parameters. The attention weight a^t_i are part of the attention distribution, and they indicate how relevant a source word is

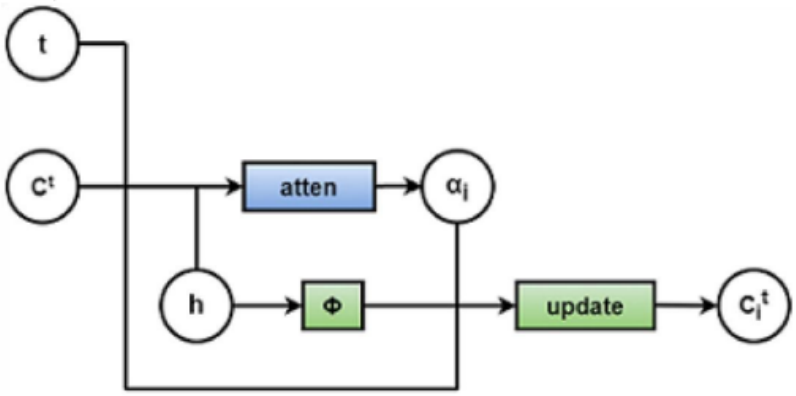


Figure 3. Coverage-based attention model

for generating an output word at this time step. The context vector h_t is a weighted sum of the encoder hidden states, and it reflects what the decoder has read from the source text at this timestep. The context vector h_t can be computed as Equation (7)

$$X\, h_t ¼ a_{ti} h_i \quad ð7Þ$$

The h_t is concatenated with the s_t and given to two linear layers to generate the P_{vocab}. Here P_{vocab} represents the vocabulary distribution as defined in Equation (8)

Input: Text document T, let n = size of summary
Output: Summarized Text
//Pre-processing
1. Encode T into $T^{'}$
2. Tokenize $T^{'}$ into individual sentence (S_i)
3. For each S_i in $T^{'}$ do:
$S_i^{'} \leftarrow$ Clean sentence
4. For each $S_i^{'}$
//extractive summarization
5.Calculate similarity matrix
6. Build a graph G
7. Calculate ranks of sentences
8. Return important sentences
//abstractive summarization
9. Encode the text using Bi-LSTM
Get embedding of the sentences
Produce encoder hidden states h_i
10. Decode the text using Bi-LSTM
Produce decoder state s_t
11. For each step t
Calculate attention distribution a^t with coverage vector c^t, s_t
Produce context vector h_t^*
Produce the vocabulary distribution P_{vocab} with h_t^*
predict words w with P_{vocab}
12. return summarized text

$$P_{vocab} = \text{softmax}(V'(V[s_t, h_t^*] + b) + b') \quad (8)$$

Where V, V′, b and b′ have been used to represent the learnable parameters. The words w is predicted with P_{vocab} as expressed in Equation (9)

$$P(w) = P_{vocab}(w) \quad (9)$$

At training, for time step t, the loss is calculated as expressed in Equation (10)

$$loss_t = -\log P(w_t) \quad (10)$$

3.2.1 Coverage mechanism

The converge-based attention model is shown in figure 3. It uses a coverage vector c^t to remember which source words have been translated before the time t. It makes alignment decision a_t by considering both the past alignment information in c^t and the untranslated source words. This helps the attention model to focus on the words that need to be translated.

Algorithm 1. Pseudo code for the proposed hybrid text summarization approach

Input text: క్యు3లో ఐసిఐసిఐ బ్యాంక్ కన్సాలిడేటెడ్ నికర లాభం 16.7 శాతం క్షీణించి 2,611 కోట్ల రూపాయలుగా నమోదైంది. మొండి పద్దులు పెరిగిపోవటం పనితీరును దెబ్బతీసిందని పేర్కొం ది. ఎన్పిఎలు ఏకంగా 4.72 శాతం నుంచి 7.91 శాతానికి పెరిగిపోయాయి. ఈ కాలంలో మొండి పద్దులు 7,000 కోట్ల రూపాయలుగా ఉన్నాయి. మొత్తం రాబడులు 25,585.14 కోట్ల రూపాయల నుంచి 27,875.67 కోట్ల రూపాయలకు పెరిగాయి. స్టాండ్ఎలోన్ ప్రాతిపదికన ఐసిఐసిఐ బ్యాంక్ 17,556.41 కోట్ల రూపాయల మొత్తం రాబడులపై 2,441.82 కోట్ల రూపాయల నికర లాభాన్ని నమోదు చేసింది. ఐలను ఆదేశించింది.

Pre-processed text: క్యు3లో ఐసిఐసిఐ బ్యాంక్ కన్సాలిడేటెడ్ నికర లాభం శాతం క్షీణించి కోట్ల రూపాయలుగా నమోదైంది. మొండి పద్దులు పెరిగిపోవటం పనితీరును దెబ్బతీసిందని పేర్కొం. ఎన్పిఎలు శాతం శాతానికి పెరిగిపోయాయి. మొండి పద్దులు కోట్ల రూపాయలుగా. రాబడులు కోట్ల రూపాయల కోట్ల రూపాయలకు పెరిగాయి. స్టాండ్ఎలోన్ ఐసిఐసిఐ బ్యాంక్ కోట్ల రూపాయల కోట్ల రూపాయల నికర లాభాన్ని నమోదు. ఐలను ఆదేశించింది.

Figure 4. Example for input and pre-processed text

Table 1. Obtained results for the news article.

Context	No. of sentences	No of words	Summary sentences	Summary words
News1	10	69	2	12
News2	12	82	3	19

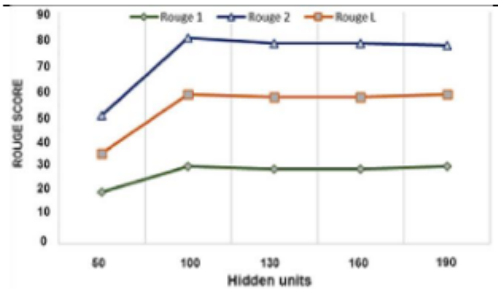


Figure 5. Average performance of proposed model with various number of hidden units

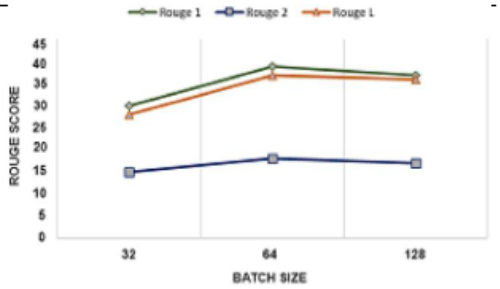


Figure 6. Average performance of proposed model with various number of batch sizes

When generating text with multi-sentence, repetition is a general issue for sequence to-sequence models [3]. The model keeps track of how much attention it has paid to each word in the source text by using a coverage vector, which is the sum of the previous attention distributions time steps. The coverage vector is used in the attention mechanism to show how well the source words have been covered and summarized so far. This way, the model can avoid paying

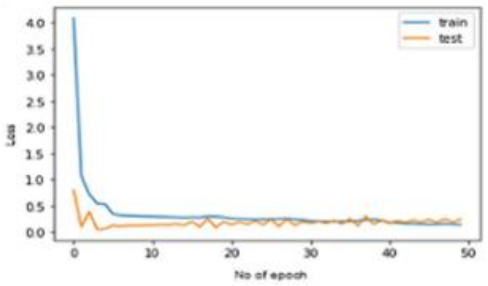


Figure 7. Training and testing loss vs Epochs

attention to any word that it has already used for summarization. Here the coverage vector c^t is the sum of a^t over time steps of all previous decoders which is defined in Equation (11)

$$c^t = \sum_{j=0}^{t-1} a^j \qquad \delta 11 \text{Þ}$$

Here c^0 represents a zero vector. The reason for c^0 is no source document has been covered on the first timestep. For the attention mechanism the c^t is employed as extra input, from equation (1), and then the e^t_i is derived in Equation (12) $e_{ti} = v^T \tanh(W_h h_i + W_s s_t + W_c c_{ti} + b_{attn})$ ð12Þ

Here w_c represents a learnable parameter vector which has the similar length as v. This shows that current decision of the attention mechanism is informed by its previous decisions' reminder. This helps the attention mechanism to avoid the repetition in text.

4. Experiments

This section explains the text pre-processing steps and dataset utilized for training and also explains the evaluation metrics utilized to evaluate the performance of the proposed model.

4.1 Text pre-processing

For pre-process the text, we first tokenize the text into sentences. To extract all the sentences from the text document, sentences are either split at full stops or whitespaces. After that the sentences are split into words. Now we clean the text document by removing full stops, special characters, numbers and stop words. Figure 4 shows the example for input and pre-processed text.

4.2 Dataset

On the internet many popular e-newspapers in Telugu language are freely available. In this paper, the e-Newspapers in Telugu language namely, Sakshi, Eenadu, Andhrajyothy, Andhrabhoomi and Vaartha are utilized to collect data. Our Telugu news dataset consists of 140 Telugu news ranging from the 1st of January 2020 to 30th of September 2020. This Telugu dataset is collected and saved in a excel file for text summarization.

Table 2. ROUGE-1 score

Input text: క్యు3లో ఐసిఐసిఐ బ్యాంక్ కన్సాలిడేటెడ్ నికర లాభం 16.7 శాతం క్షీణించి 2,611 కోట్ల రూపాయలుగా నమోదైంది. మొండి పద్దులు ఎరిగిపోవడం పనితీరును దెబ్బతీసిందని పేర్కొం ది. ఎన్పిఎలు ఏకంగా 4.72 శాతం నుంచి 7.91 శాతానికి ఎరిగిపోయాయి. ఈ కాలంలో మొండి పద్దులు 7,000 కోట్ల రూపాయలుగా ఉన్నాయి. మొత్తం రాబడులు 25,585.14 కోట్ల రూపాయల నుంచి 27,875.67 కోట్ల రూపాయలకు పెరిగాయి. ప్రొవిజనింగ్ ప్రాతిపదికన ఐసిఐసిఐ బ్యాంక్ 17,556.41 కోట్ల రూపాయల మొత్తం రాబడులపై 2,441.82 కోట్ల రూపాయల నికర లాభాన్ని నమోదు చేసింది. బులను ఆదేశించింది.

Summary: ప్రొవిజనింగ్ ఐసిఐసిఐ బ్యాంక్ రూపాయల నికర నమోదు క్యు3లో కన్సాలిడేటెడ్ లాభం శాతం క్షీణ

Figure 8. A qualitative example and comparison with proposed model

Context	Epochs	ROUGE-1

		F1 score	Precision	Recall
News1	50	0.382	0.363	0.393
	100	0.373	0.358	0.387
News2	50	0.351	0.343	0.354
	100	0.339	0.334	0.328

Table 3. ROUGE-2 score

		ROUGE-2		
Context	Epochs	F1 score	Precision	Recall
News1	50	0.162	0.185	0.158
	100	0.153	0.176	0.140
News2	50	0.158	0.164	0.143
	100	0.138	0.157	0.131

Table 4. ROUGE-L score

		ROUGE-L		
Context	Epochs	F1 score	Precision	Recall
News1	50	0.371	0.353	0.373
	100	0.363	0.338	0.364
News2	50	0.364	0.343	0.369
	100	0.353	0.338	0.352

4.3 Evaluation metrics

The ROUGE toolkit gives n-gram overlap between the reference summary and the system generated summary. To evaluate the performance of our text summarization model, it is utilized. For automatically calculate the summary quality it compares with human generated summary. The N-gram units common between the system generated ted summary and reference summary is measured by ROUGEN. The Longest Common Subsequence (LCS) metric is computed by ROUGE-L. Rouge generates the F-measure, recall and precision metric.

Precision

$$\text{Precision} = \frac{\text{Gold standard summary} \setminus \text{Modelgeneratedsummary}}{\text{Modelgenerated summary}} \quad \text{ð13Þ}$$

Recall

$$\text{Recall} = \frac{\text{Gold standard summary} \setminus \text{Modelgeneratedsummary}}{\text{Gold standard summary}} \quad \text{ð14Þ}$$

F measure

$$\text{F measure} = \frac{2 \text{ Precision Recall}}{\text{Precision þ Recall}} \quad \text{ð15Þ}$$

In this paper for performance evaluation we have used F-measure, recall and precision metrics for ROUGE-1, ROUGE-2 and ROUGE-L scores.

5. Experimental results

This section details the experimental results for the performance evaluation of our proposed model using ROUGE toolkit. The table 1 shows the comparison for Telugu news dataset with the number of sentences and words before and after summarization. This table shows that the number of sentences and number of words after summarization is reduced considerably. For two validation set these results are calculated.

Different parameters are tuned in our model to perform some comparative analysis. By changing the size of the hidden units the performance of our model has been observed. The number of hidden units selected for the experiment is within the range 100 to 220. From the figure 5 we can know that the hidden units more than 100 causes overfitting so that instead of learning the the model will memorize the data. Thus for the unseen data the summarization effficency of model will be less. However if we increase the number of hidden units than 100 on the performance of our model it will not make any significant impact. But, it will increase the training time. So that, in our proposed model the size of hidden units is kept to 100 for improving the performance and to reduce the training time.

Moreover, for different batch sizes we have tested our model to determine the amount of training data of the model. For three different batch sizes, the performance of our model is evaluated with size of hidden unit at 100 as shown in figure 6. At the batch size 32, our model has very low performance. Until the batch size reached 64 the performance increased sharply and then flattens out. The reason behind this is that to the sharp minimizers of the training function, the large batch size tends to converge. This will reduce the ability to precisely treat the unseen data and the generalization power of the model. Thus, keeping the batch size above 64 has diverse effect on the model performance. But the training

time will be higher if we increase the batch size. So that, to get better results on the model performance the batch size is kept at 64 in our model.