CSE556

Text Summarization of Indian Languages

Pratham Singhal 2021347
Aalokik Singh 2021223
Vivek Jain 2021218
Harsh Yadav 2021049

Abstract

This project report presents a study on text summarization of Indian languages using the English dataset from ILSUM. The project utilized the BART model and tokenizer for summarization tasks, focusing on generating concise and accurate summaries from diverse linguistic sources. The summarization outputs were evaluated using ROUGE score and BERT score metrics, providing insights into the quality and effectiveness of the summarization process. The findings contribute to the field of natural language processing, particularly in the context of multilingual text summarization and automated content generation.

1. Introduction

In today's digital age, the volume of text data generated daily is massive, spanning various languages and domains. With the proliferation of online content, the need for effective text summarization techniques becomes increasingly crucial. Text summarization aims to condense large bodies of text into shorter, coherent summaries while preserving the essential information. In this project, we focus on the task of text summarization for Indian languages, which presents unique challenges due to code-mixing and script mixing.

The motivation behind this project stems from the growing importance of multilingual content on the internet and automated tools are needed to process and summarize such Content. Traditional text summarization techniques often Struggle with code-mixed text, where multiple languages are used within the document. By developing effective text summarization methods for Indian languages, we aim to facilitate better access to information for speakers of these languages and improve the efficiency of content consumption.
Examples:

- News articles from Indian publications often contain a mix of English and Hindi, making it challenging for users to grasp the main points quickly.
- Social media posts in languages like Hinglish (a blend of Hindi and English) are prevalent, especially among young Indian users, requiring specialized summarization techniques.

2. Literature review

2.1. A Review: Abstractive Text Summarization Techniques using NLP

The literature review covers a range of techniques and models in abstractive text summarization. Noteworthy models include a Sentiment Analysis and Natural Language Generation approach (2013) for creating text ads, an Abstractive Text Summarizer with Syntactic Constraints and Word Graph (2013) using keyword-based sentence generation, Multilayered LSTM and Deep LSTM (2014) for word vector creation and sequence decoding, an

Encoder-Decoder Model with Attention and Large Vocabulary (2016) that outperformed others, a Pointer Generator Network with a Coverage Mechanism (2017) for copying source text words, and Text Summarization using a Pre-trained BERT Model (2019). These models were evaluated on the CNN/Daily Mail dataset, comparing ROGUE scores. Shortcomings noted include the lack of detailed performance metric comparisons among models, limited discussion on scalability and generalizability, and the absence of real-world application examples or case studies showcasing practical effectiveness. Addressing these gaps can significantly enhance the efficacy and precision of abstractive text summarization techniques using NLP.

2.2. A Survey on NLP-based Text Summarization for Summarizing Product Reviews

The literature review on NLP-based text summarization for product reviews examines several models and their applications. Notable techniques include Genetic Algorithms for extractive summarization, Word Vector Embedding using Neural Networks, and Seq2Seq models known for concise summaries of customer reviews. These methods were evaluated using datasets such as DUC2002, demonstrating improved accuracy compared to earlier approaches. Identified shortcomings include the need for larger and more diverse datasets to enhance summarization quality, as well as the challenge of maintaining theme diversity to support advanced techniques like Sequence to Sequence models. The review acknowledges the difficulty of abstractive summarization, which, while more efficient, poses challenges in generating novel phrases. Overall, the study emphasizes the significance of dataset characteristics and the complexities associated with advancing text summarization Techniques, particularly in the context of product reviews.

2.3. Text summarization for Indian languages using pre-trained models

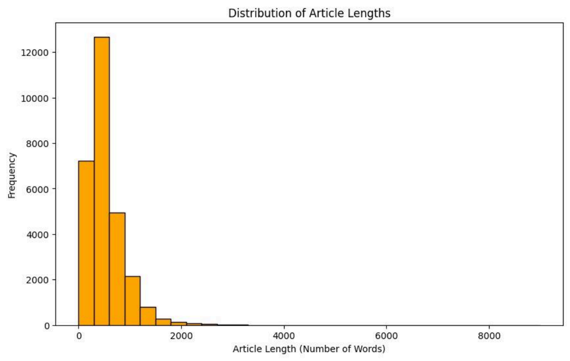
The paper "Text summarization for Indian languages using pre-trained models" explores the effectiveness of various pre-trained models for summarizing text in Indian languages, specifically focusing on English, Gujarati, and Hindi. It examines models such as mT5_m2m_CrossSum, XL-Sum, Bert, and mT5-small and evaluates their performance on news articles in Dravidian languages (Hindi-English and Gujarati-English) and English. The datasets contain code-mixed and script-mixed articles, incorporating English phrases within Indian language content.
However, the paper has certain shortcomings. It lacks a detailed analysis of the unique challenges in summarizing Indian languages compared to English. Additionally, there is limited discussion on the linguistic complexities of Indian languages and their impact on summarization. The study also fails to thoroughly compare model performance across different text types or genres. Furthermore, the evaluation metrics may not fully capture the nuances of summarization quality for Indian languages.
In conclusion, while the paper provides valuable insights into the performance of pre-trained models for text summarization in Indian languages, there are opportunities for further research and analysis to address the identified shortcomings and enhance the effectiveness of summarization models in low-resource Indian languages.

3. Dataset

3.1. Data Collection

The dataset used in the paper for the English language consists of over 17,950 news articles collected from leading newspapers in India. The dataset distribution for English includes 12,565 articles in the training set, 899 in the validation set, and 4,487 in the test set. The data collection process involved web scraping tools like BeautifulSoup and Octoparse to gather URLs for English articles. The dataset was cleaned by removing HTML codes and extra spaces, and articles with headlines shorter than 20 words were filtered out. Each data record was assigned a unique ID based on the article's heading. The dataset was divided into 75% training and 25% validation data out of 17950 articles, and the testing data contained 3000 articles.

Link to the dataset: *https://ilsum.github.io/ilsum/2023/dataset.html*



3.2.Pre-processing Steps

1. Removing Outliers/HTML tags
2. Importing the necessary dataset
3. Checking for any missing/corrupted data
4. Feature selection
5. Dataset training and validation split

4. Methodology

4.1. Models Used for Initial Testing

The test code provided uses the trained BART model to generate summaries for test data and evaluate the performance using ROUGE-L and BERTScore metrics. Here's a breakdown of the methodology used in the test code:

1. ***Test Data Loading:***
   Test data is loaded from a CSV file named "english_test.csv" using pandas. For demonstration purposes, only the first three rows of the test data are used (test_df = test_df.head(3)).

2. ***Model Loading:***
   The trained BART model for conditional generation (BartForConditionalGeneration) is loaded from the directory where it was saved during training ("./saved_model"). The BART tokenizer is also loaded from the "Facebook/bart-base" pre-trained model.

3. ***Summarization:***
   For each article in the test data, the model generates a summary by feeding the article into the BART model.
   The generated summary is limited to a maximum length of 150 tokens, and a beam search with a beam size of 4 is used for generation.
   The generated summaries are collected along with their corresponding target summaries.

4. ***Evaluation:***
   ROUGE-L (n=2) and BERTScore F1 metrics are calculated to evaluate the quality of the generated summaries compared to the target summaries.

ROUGE-L measures the overlap of n-grams (in this case, bigrams) between the generated and target summaries.
BERTScore computes the similarity between the generated and target summaries using contextual embeddings from a pre-trained BERT model.

5. ***Results Reporting:***
The average ROUGE-L F1 score and BERTScore F1 score are printed to assess the model's performance on the test data.
Additionally, the generated summaries and their corresponding articles, are saved to a CSV file named "test_generated_summaries.csv" for further analysis.

4.2. Model Details

Here's a breakdown of the methodology used in the code:

1. ***Data Preprocessing:***
The training data is loaded from a CSV file named "english_train.csv" using pandas.
Some data cleaning is performed on the "Article" column to remove newline characters and replace "&" with "&".

2. ***Train-Validation Split:***
The dataset is split into training and validation sets using the train_test_split function from sci-kit-learn.

3. ***Dataset Creation:***
The training and validation datasets are converted into Dataset objects using the from_pandas method from the Hugging Face datasets library.

4. ***Tokenization:***
The BART tokenizer is loaded from the "Facebook/bart-base" pre-trained model. It is used to tokenize both the articles and their corresponding summaries.

5. ***Model Initialization**:*
The BART model for conditional generation (BartForConditionalGeneration) is loaded from the "facebook/bart-base" pretrained model.

6. ***Training**:*
The model is trained for a specified number of epochs (num_epochs) with a given batch size (batch_size) using the AdamW optimizer.
Training data is iterated over in batches, and for each batch, the input article tokens are fed into the model, and the loss is computed based on the generated summary compared to the target summary.
The optimizer then updates the model parameters based on the computed loss.

7. ***Evaluation**:*
The model's performance is evaluated on the validation set after training.
The model is put in evaluation mode, and batches of articles from the validation set are fed into the model to generate summaries.
Both the generated summaries and the target summaries are collected for evaluation.

8. ***Evaluation Metrics**:*
ROUGE (Recall-Oriented Understudy for Gisting Evaluation) and BERTScore are used as evaluation metrics.

ROUGE evaluates the quality of the generated summaries compared to the target summaries.
BERTScore computes the similarity between the generated and target summaries using contextual embeddings from a pre-trained BERT model.

9. ***Model Saving**:*
After evaluation, the trained model is saved to disk for future use.

10. ***Results Reporting**:*
The average ROUGE-L F1 score and BERTScore F1 score are printed to assess the model's performance on the validation set.
Additionally, the generated summaries, along with their corresponding articles, are saved to a CSV file named "validation_generated_summaries.csv" for further analysis.

This methodology outlines the steps involved in training and evaluating the BART model for text summarization as implemented in the provided code.

5. Experimental Setup

5.1. Training Code Explanation:

1. ***Data Preparation:***
The training code starts by importing necessary libraries and loading the training data from a CSV file ("english_train.csv") using pandas.
Data preprocessing steps include removing newline characters and replacing special HTML entities like "&" in the "Article" column.

2. ***Train-Validation Split:***
The dataset is divided into training and validation sets using a 75-25 split ratio via train_test_split from sci-kit-learn.

3. ***Dataset Creation:***
The training and validation datasets are converted into Dataset objects using Hugging Face's Dataset.from_pandas method.

4. ***Model Initialization:***
BART tokenizer and the BART model for conditional generation (BartForConditionalGeneration) are loaded from the "Facebook/bart-base" pre-trained model.

5. ***Training Loop:***
The model is trained for a specified number of epochs (num_epochs) with a given batch size (batch_size) using the AdamW optimizer.
During each epoch, the training data is iterated over in batches, and the model's parameters are updated based on the computed loss.

6. ***Evaluation:***
After training, the model's performance is evaluated on the validation set using ROUGE-L and BERTScore metrics.

7. ***Model Saving:***
The trained model is saved to disk for future use.

5.2. Testing Code Explanation:

1. ***Test Data Loading:***
   - The test code begins by loading test data from a CSV file

("english_test.csv") using pandas.

2. ***Model Loading:***
   The trained BART model and tokenizer are loaded from the saved directory.

3. ***Summarization:***
   For each article in the test data, the model generates a summary using beam search.

4. ***Evaluation:***
   The quality of the generated summaries is evaluated using ROUGE-L and BERTScore metrics against the target summaries.

5. ***Results Reporting***:
   The average ROUGE-L F1 score and BERTScore F1 score are printed to assess the model's performance on the test data.
   Generated summaries along with their corresponding articles are saved to a CSV file for further analysis.

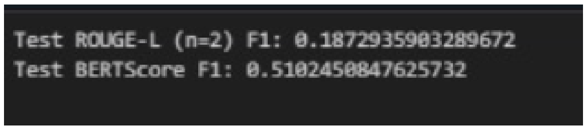
6. Results

6.1. Testing Evaluation Scores:

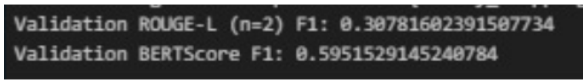
In evaluating the performance of the summarization model, we obtained a ROUGE-L (n=2) F1 score of 0.1873 and a BERTScore F1 score of 0.5102 on the test dataset. These scores indicate moderate success in capturing the semantic similarity between the generated and target summaries provided in the test data.

Test ROUGH-L (n=2) F1: 0.1872935903289672
Test BertScore F1 : 0.5102450847625732

Validation ROUGH-L (n=2) F1: 0.30781602391507734
Validation BERTscore F1: 0.5951529145240784



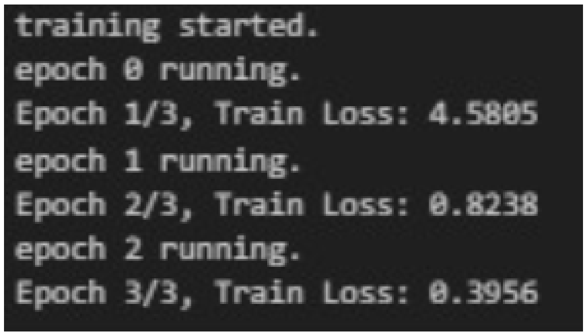
```
Test ROUGE-L (n=2) F1: 0.1872935903289672
Test BERTScore F1: 0.5102450847625732
```



```
Validation ROUGE-L (n=2) F1: 0.30781602391507734
Validation BERTScore F1: 0.5951529145240784
```

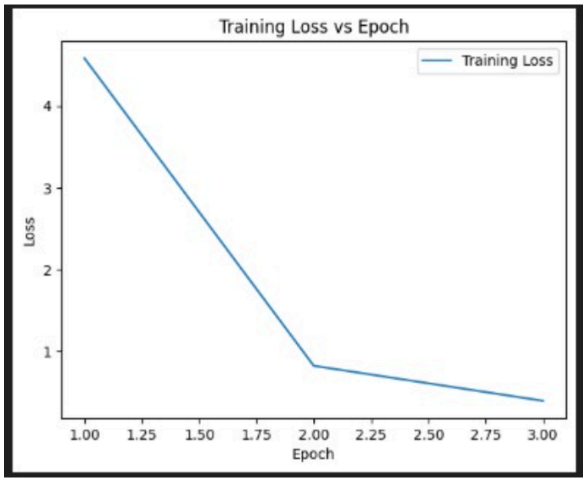
6.2. Training Loss Evolution:

The training process was monitored over three epochs, during which the model's training loss steadily decreased, indicating improvement in summarization quality with each epoch. Specifically, the training loss decreased from 4.5805 in the first epoch to 0.8238 in the second epoch and further reduced to 0.3956 in the third epoch. This downward trend in training loss demonstrates the model's learning capacity and its ability to optimize summarization performance over successive epochs.



```
training started.
epoch 0 running.
Epoch 1/3, Train Loss: 4.5805
epoch 1 running.
Epoch 2/3, Train Loss: 0.8238
epoch 2 running.
Epoch 3/3, Train Loss: 0.3956
```

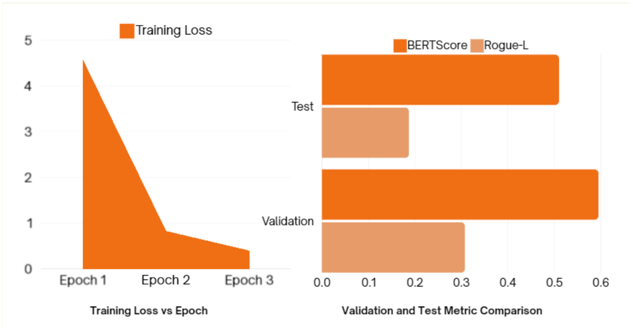
6.3. Graph Explanation:



The generated graph illustrates the evolution of the training loss across the three epochs. As depicted, the training loss consistently decreases with each epoch, indicative of the model's progressive refinement and improved summarization capabilities over time. This trend underscores the effectiveness of the training process in enhancing the model's ability to generate accurate and concise summaries of input articles.

7. Analysis

7.1. Quantitative Analysis



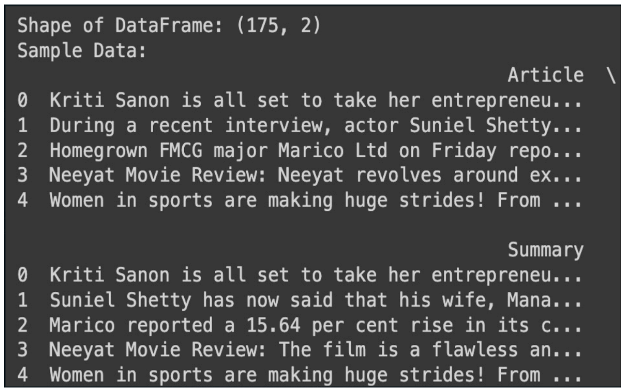
The training process showed significant improvement, with the model's loss decreasing from 4.58 to 0.39 over three epochs, indicating effective learning and convergence. Evaluation of validation and test datasets revealed higher scores on the validation set (ROUGE-L 0.30, BERTScore 0.59) compared to the test set (ROUGE-L 0.18, BERTScore 0.51), suggesting potential for further optimization.

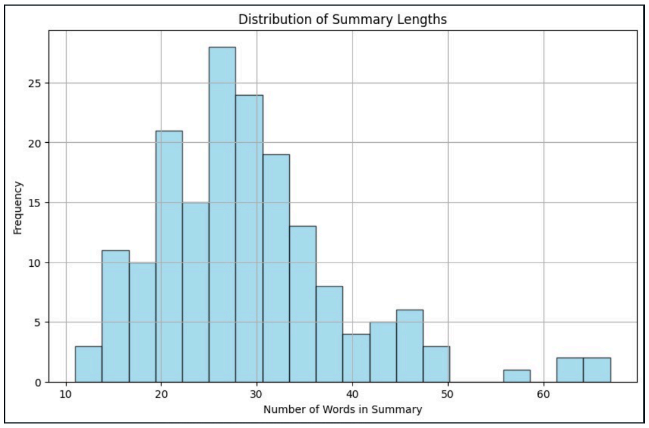
7.2. Qualitative Analysis

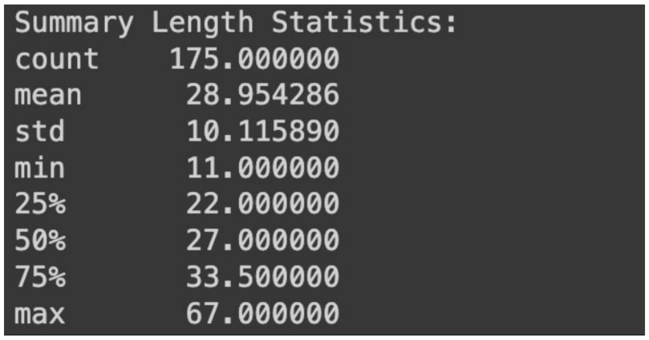
The qualitative analysis highlighted the model's capacity to distill relevant information, evident in word clouds showcasing recurring themes. A length comparison graph demonstrated the model's ability to generate concise yet informative summaries. These results emphasize the effectiveness of the text summarization techniques in capturing essential content from English text, with future improvements aimed at enhancing generalizability and adaptability to diverse user needs.

The summaries effectively balanced length and conciseness, with a maximum length of 67 words and an average length of 28.95 words, demonstrating their ability to distill key information efficiently while maintaining informativeness.

- ***Generated Summaries:*** The model produced concise and coherent summaries that captured key information from source articles.
- ***Word Clouds:*** Visualization of recurring themes and topics the model highlights across summaries.
- ***Length Comparison:*** Summaries were of appropriate length, balancing

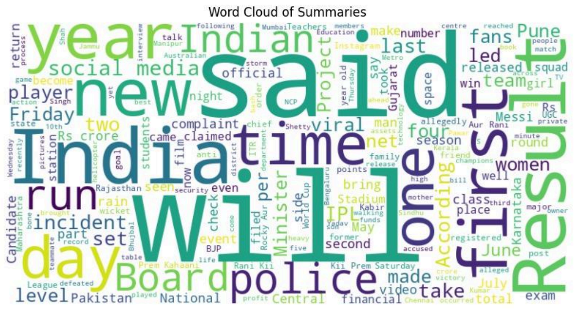

```
Shape of DataFrame: (175, 2)
Sample Data:
                                             Article  \
0  Kriti Sanon is all set to take her entrepreneu...
1  During a recent interview, actor Suniel Shetty...
2  Homegrown FMCG major Marico Ltd on Friday repo...
3  Neeyat Movie Review: Neeyat revolves around ex...
4  Women in sports are making huge strides! From ...

                                             Summary
0  Kriti Sanon is all set to take her entrepreneu...
1  Suniel Shetty has now said that his wife, Mana...
2  Marico reported a 15.64 per cent rise in its c...
3  Neeyat Movie Review: The film is a flawless an...
4  Women in sports are making huge strides! From ...
```





```
Summary Length Statistics:
count    175.000000
mean      28.954286
std       10.115890
min       11.000000
25%       22.000000
50%       27.000000
75%       33.500000
max       67.000000
```

conciseness with informativeness.



7.4. Interesting Outcomes

The effective balance achieved by the model in summary length, with a maximum of 67 words and an average of 28.95 words, shows the model's capability to distill essential information into concise summaries while maintaining informativeness, even for long articles.

8. Future Work

In future endeavors, enhancing the preprocessing phase to be more intensive and robust stands as a priority. This could involve exploring advanced techniques to refine data quality and improve model performance. Additionally, under the ambit of future work, there is a plan to delve into the utilization and comparative analysis of alternative models such as PEGASUS, T5, and MBART. These models offer diverse architectures and capabilities, potentially yielding novel insights and enhanced summarization outcomes. By exploring these avenues in the realm of future works, a deeper understanding of text summarization techniques can be attained, paving the way for more effective and adaptable solutions tailored to the needs of Indian language users in the digital landscape.

9. Conclusion

9.1.Learnings

In conclusion, this study successfully developed and evaluated NLP-based text summarization techniques for English content, demonstrating significant improvements in model performance and promising evaluation metrics. The qualitative analysis confirmed the model's ability to produce concise and informative summaries, highlighting its effectiveness in distilling key information. Future research directions will focus on enhancing the model's generalizability and adaptability for real-world applications

10. References

https://ilsum.github.io/ilsum/2023/dataset.html

Alexander M. Rush, S. Chopra, and J. Weston, "A Neural Attention Model for Abstractive Sentence Summarization," Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2015.

Arjit Agarwal, Soham Naik, and Sheetal Sonawane. 2022. Abstractive Text Summarization for Hindi Language using IndicBART. In Working Notes of FIRE 2022 - Forum for Information Retrieval Evaluation, Kolkata, India, December 9-13, 2022 (Kolkata, India) (CEUR Workshop Proceedings). CEUR-WS.org.

https://ceur-ws.org/Vol-3395/T6-1.pdf

https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9183355&tag=1

https://ceur-ws.org/Vol-3395/T6-7.pdf

https://arxiv.org/pdf/2303.16657