[1]Jeetendra Kumar,
[2]Shashi Shekhar,
[3]Rashmi Gupta

Hindi Abstractive Text Summarization using Transliteration with Pre-trained Model

Abstract: - Automatic text summarization is a subarea of natural language processing that generates a summary of the text by keeping its key points. The research work done on summarizing low-resourced language text is very limited. In India, the Hindi language is being spoken by central and north Indian people and only a few research works have been done on abstractive summarization of Hindi language. Having *matras* in Hindi makes it difficult to tokenize so it is difficult to summarize Hindi text using abstractive text summarization. In the proposed method, abstractive Hindi text summarization is done using transliteration and fine-tuning. In this work, the model is trained to generate both summaries and headlines. ROUGE-score and BERT-score have been utilized to check summary quality. A new semantic similarity score-based performance measure is also proposed to measure semantic similarity between reference summaries and predicted summaries. Using the proposed method, we have achieved the highest 55.16 ROUGE score, 0.80 BERT score, and 0.98 similarity score. Along with these performance measures, human evaluation of predicted summaries is also done and it is found that summaries and headlines were generated at a human-acceptable level.

Keywords: Hindi text summarization, transliteration, fine-tuning, pre-trained model, semantic similarity.

1. INTRODUCTION

Text summarization is getting popular nowadays because digital data is flooding every day on the internet. If someone is searching for any information on the internet then a lot of digital data is shown on the screen and everyone can't read all the data to extract useful facts. Automatic Text Summarization (ATS) techniques have gained popularity because they produce a useful and concise summary of large texts automatically without human intervention. This reduces the time need extract useful information from large texts. One of the important requirements of ATS techniques is to generate semantically correct summaries without losing the original intent of the text. The concept of ATS is not new but it came into existence, when (Luhn, 1958) proposed text summarization using statistical techniques. Earlier text summarization methods were based on statistical methods to produce summaries. With the increase in high-speed memory and parallel computing resources, many artificial intelligence-based techniques have been used in ATS. The deep learning and AI-based text summarization methods achieve better results in generating better, coherent, and more human-like summaries. ATS has many advantages (1) It reduces the reading time (2) During the research, summary selection from literature is easy with ATS. (3) Automatic summarization methods are not biased like humans and (4) They can generate summaries of different lengths from the same text. Sentiment-based summarization (Ali et al., 2020), headline generation(Arora, 2020), and customer review summarization (Alsaqer & Sasi, 2017) are some application areas of ATS. Text summarization can be done in many ways, depending on the strategies used. Figure 1 shows the important categories of text summarization (El-Kassas et al., 2021).

[1] [1] Atal Bihari Vajpayee University, Bilaspur, India, jeetendragupta@bilaspuruniversity.ac.in
[2] GLA University, Mathura, India, shashi.shekhar@gla.ac.in
[3] Atal Bihari Vajpayee University, Bilaspur, India, rashmigupta@bilaspuruniversity.ac.in

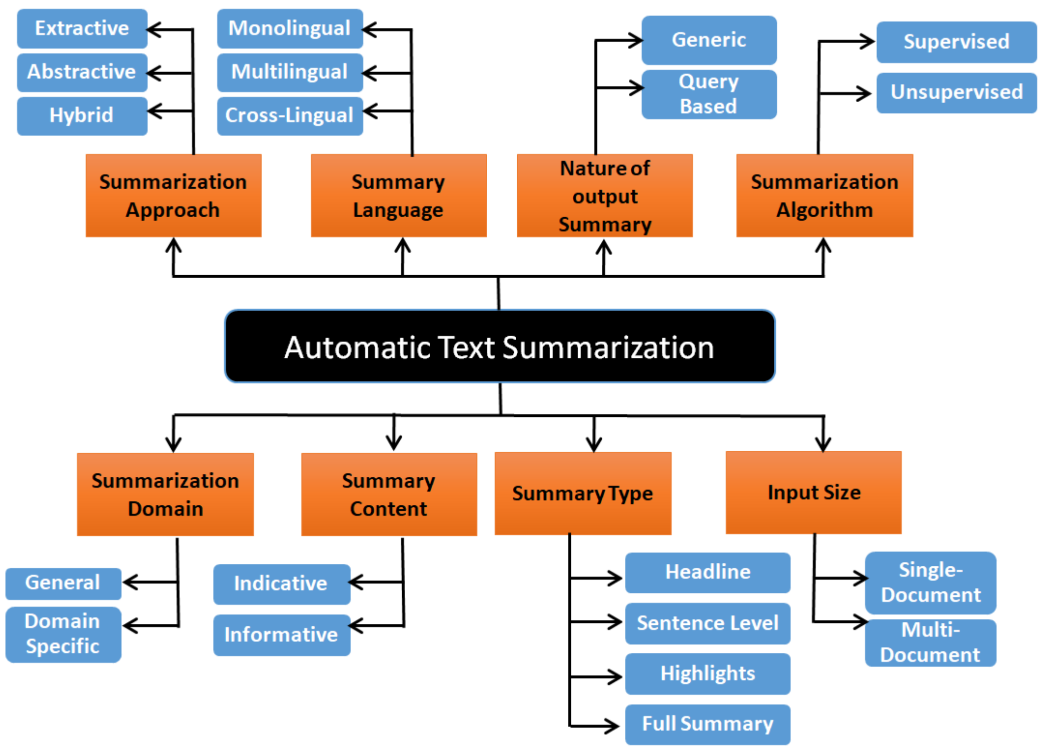

Fig 1. Categories of ATS methods.

Based on the input size, text summarization methods can be single-document summarization and multi-document summarization. In single-document summarization, single input is given and a single summary is generated but in multi-document summarization, multiple document inputs are given and a single summary is generated. Query-based summarization generates summaries from multiple documents based on query results while generic summarization generates summaries from one or multiple documents. In the supervised text summarization method, supervised training algorithms need to be trained on the labeled data. In unsupervised text summarization methods, labeled data is not required. The indicative summary identifies the document's topic, and the informative summary elaborates on some topics according to the user's interest. In domain-based methods, summarization is done on the text of the general domain while in domain-specific summarization, summarization is done on the text of specific domain texts like medical documents, sports documents, legal documents, etc. A monolingual summarization summarizes text written in a single language, a multilingual summarization summarizes texts written in multiple languages, and a cross-lingual summarization summarizes text written in one language to a summary written in another language. Based on the length of generated summaries, different length summaries can be generated like headlines generation, single sentence summary generation, highlights generation, or full summary generation. The last category of summarization is based on the summarization approach. It is very common and most text summarization algorithms are categorized accordingly. The extractive summarization method extracts important sentences from the text to form a summary. Abstractive summarization methods apply Deep Learning based algorithms to generate human-like summaries. Hybrid summarization uses both abstractive and extractive approaches to generate summaries. Extractive summarization uses statistical methods while abstractive summarization uses AI-based Seq2Seq models and advanced transformer models.

A lot of research work has already been done on English text summarization but very little work has been done in the area of Hindi text summarization. Most of the residents in the northern and central Indian states like Madhya Pradesh, Rajasthan, Delhi, etc. speak Hindi as their first language. Hindi is also spoken in Mauritius, Fiji, Guyana, Suriname, Trinidad & Tobago, and Nepal. It is the fourth most-spoken language in the world with 344 million native speakers (*Top 10 Most Spoken FIRST Languages in the World in 2023 | TranslateDay*, n.d.). Furthermore, Hindi is widely spoken in many nations other than India. It has encouraged us to study NLP-based techniques for summarization.

Most of the research works done for summarization of Hindi text utilized extractive approaches which are extensively based on statistical methods to select important sentences from large texts. Summaries generated using extractive text summarization are not coherent and semantically disconnected. Hindi is more complex than English because of *matras.* As a result, little research has been done on Hindi text summarization.

Research Problems and Objectives

- We found that due to *matras* in the Hindi language, it is very difficult to tokenize Hindi words in the right way. Very few Hindi tokenizers are available, and sometimes they cannot tokenize Hindi words correctly. We have seen the case of popular Hindi tokenizers from the inltk library[(Arora, 2020)] and tried to tokenize some sample words. It was found that many Hindi words were wrongly tokenized. Like ”**कियारा**” was tokenized into two token i.e ”**किया**” and ”**रा**”, ”**करगिल**” word is tokenized as ”**कर**” and ”**गिल**”. ”**कोहरा**” word is tokenized as ”**को**” and ”**हरा**”. Due to this, tokenization can generate summaries in the wrong context. To tackle this problem, we have proposed that instead of developing new tokenizers for the Hindi language, the transliteration process can be used to transliterate the text from Hindi to English languages. The English language doesn’t have *matras* therefore it is easy to tokenize.
- In continuation of the above objective, it is already known that fewer pre-trained models are available for Hindi text summarization whereas, for English text summarization, enhanced pre-trained models are available. Consequently, the second objective of the proposed work is to summarize the transliterated Hindi text using the model that is already trained on English text.
- The third objective of the proposed work is to explore a new method to calculate semantic similarity between reference summaries and predicted summaries.

The main contributions of this paper are:

- Transliteration of Hindi text before summarization is done to generate better summaries.
- A pre-trained model trained on English text is fine-tuned to summarize Hindi text.
- To compare the semantic similarity of predicted and reference summaries, we have proposed a method to calculate the semantic similarity between reference summaries and predicted summaries.
- Randomly selected summaries have been evaluated by experts.

2. REVIEW OF LITERATURE

Text summarization is the task of generating concise summaries from a large text by keeping the important points. Text summarization is an active research area with many exciting developments and advancements. We may anticipate more advancements in the caliber and efficiency of text summarizing systems as NLP technology develop further. This field has seen a great deal of study, and continuing research is still being done in it. Initially, the text summarization algorithms were focused on English text, but research work is also being carried out on different languages like French, Chinese, German, Spanish, etc., and other low-resource languages (Žagar & Robnik-Šikonja, 2022). In the last few years, research work on the summarization of Indian languages like Hindi, Tamil, Telugu, and Bengali is also been done. Since our proposed work is on Hindi text summarization so at first we will discuss some recent research works on low-resource language text summarization, then we will confer research works on Indian language summarization and Hindi text summarization.

2.1. Summarization of Low Resource Language

Many authors have proposed abstractive summarization of low-resource languages. (Nawaz et al., 2020) proposed extractive summarization of Urdu. They used local weight and global weight-based approaches and obtained promising results. (Bani-Almarjeh & Kurdy, 2023) proposed Arabic text summarization with RNN-based transformer architecture. They used four pre-trained language models mBERT, AraBERT, AraGPT2, and AraT5 for Arabic text summarization. They found that AraT5 and AraGPT2 performed better than other language models. (Shafiq et al., 2023) proposed abstractive summarization of Urdu. They used multilayer encoder and single layer decoder-based transformer. Using the proposed method, they achieved acceptable ROUGE scores. (Wijayanti et al., 2023) proposed abstractive summarization of low-resource languages. They analyzed two bilingual embeddings i.e. VecMap and BiMap for Indonesian languages. Their concept improved the performance of VecMap by 16.6% when evaluated intrinsically.

2.2. Summarization of Indian Languages

The work of abstractive text summarization is also done to summarize the text of other Indian languages like Punjabi, Tamil, Telugu, Malayalam, etc. For summarization of Punjabi text**,** many authors have proposed extractive approaches. (Garg et al., 2021) proposed extractive Punjabi text summarization using an unsupervised machine learning approach. After pre-processing the text, they generated a similarity matrix and ranked it, based on the similarity matrix. Using the proposed method, they achieved values of ROUGE_1, ROUGE_L, and ROUGE_s scores as 0.71, 0.56, and 0.56, respectively. (Jain, Arora, et al., 2021) proposed Punjabi text summarization using three-phase extractive text summarization. After pre-processing, they extracted statistical and linguistic features. They used neural networks with sigmoid activation functions with optimization. Using their proposed method, they achieved 90.0%, precision, 89.28%, recall, and 89.65% F-measure. (Jain, Yadav, et al., 2021) proposed particle swarm optimization for Punjabi text summarization. They used two datasets, and using their proposed method, they achieved 78.36% precision, 79.5% recall, and 78.96% F-measure as, 79.57. For Tamil text summarization, (Dhivyaa et al., 2022) proposed transliteration and transformer-based summarization of Tamil text. For fine-tuning, they used three pre-trained models i.e. GPT-2, T5, and BERT models. They found that GPT-2 model achieved better summarization performance T5 and BERT. (Manjari, 2020) proposed extractive Telugu summarization using the TextRank algorithm. Followed by text preprocessing, they used the continuous bag of words model architecture. After the calculation of word embedding, they calculated a similarity matrix for generating summaries. They achieved average values of precision, recall, and F-measure as 0.60, 0.63, and 0.62 respectively. (Mamidala & Sanampudi, 2021) proposed a heuristic approach for extractive Telugu text summarization. They used event score and named entity score for sentence ranking of Telugu text. Using their proposed method, they achieved better performance than other Telugu text summarization methods. (Babu & Badugu, 2023) proposed Telugu text summarization using the Seq2Seq model. They used a Bi-LSTM-based encoder and LSTM-based decoder in their proposed Seq2Seq model. They found that their proposed method outperformed other Telugu text summarization methods. (Durga et al., 2022) proposed Telugu text summarization using histo-fuzzy means and Median Support Based Grasshopper Optimization Algorithm (MSGOA). After pre-processing the data, they clustered text documents. Using their proposed method, they achieved 84% accuracy. For Malayalam text summarization, (Jaya et al., 2019) proposed the summarization of Malayalam documents using the clause identification method. They used a POS tagger and morphological analyzer for Malayalam text. Using their proposed method, they were able to generate a clear and concise summary. (Nambiar et al., 2021) proposed abstractive text summarization of

Malayalam text. They used the Seq2Seq model with an attention mechanism for summarizing Malayalam text. Using their proposed model, they achieved better results. (K. Nambiar et al., 2023) proposed Malayalam text summarization using an enhanced attention model and POS tagging. Using their proposed method, they achieved a 0.28 ROUGE score.

2.3. Extractive Hindi Text Summarization

Extractive text summarization is the process of extracting highly important sentences from a large text. (Bafna & Saini, 2019) proposed Hindi summarization using unsupervised learning. They utilized TF-IDF, cosine-based document similarity measures, and cluster dendrograms. Using the proposed method, they achieved values of entropy score and precision as 0.31 and 0.90, respectively. (Rani & Lobiyal, 2021) proposed LDA (Linear Discriminant Analysis) tagged-based topic modeling for text summarization. They proposed a PoS tagger for Hindi and also proposed four different POS-based, topic modeling-based schemes based on different weighting strategies. They achieved 70% ROUGE score. (Joshi et al., 2021) proposed semantic graph-based extractive text summarization. In their proposed method, they constructed a semantic graph of Hindi documents by establishing the semantic relationship between sentences using wordnet ontology. They found that their proposed methods performed better than other TextRank algorithms. (Dhankhar & Gupta, 2022) proposed a sentence-scoring method based on statistics for Hindi text summarization. With their mathematical combination of nine textual feature-based methods and achieved 0.40 ROUGE score. (Jain et al., 2022) proposed Hindi text summarization using a real-coded genetic algorithm. After preprocessing the data, they extracted eight features from the text. Then they used genetic algorithms. Using their proposed method, they achieved 78% ROUGE_1 recall and 68% ROUGE_2 recall. (Supreet et al., 2020) proposed selection and elimination-based approach to summarize Hindi text. After pre-processing the data, they calculated word frequency, sentence frequency, dice coefficient of similarity, and jacquard coefficient of similarity. Their proposed method produced a summary that was 30%-40% similar to the original text. (Tawatia et al., 2022) proposed extractive summarization of Hindi documents using neural methods. They extracted the data from the website of “AajTak” Hindi news and train the data with a neural extractive model summarizer. By using their proposed method, they achieved values of F1 scores for ROUGE_1 as 20.32 and for ROUGE_2 as 39.81. (Bandari & Bulusu, 2023) proposed political elephant herding optimization-based LSTM for extracting Hindi text summarization and generated the summary for Hindi text in four stages i.e., preprocessing, feature extraction, score generation, and score extraction. Using the proposed method, they achieve 77.5% ROUGE score. (Aote et al., 2023) proposed binary particle swarm optimization-based multi-document Hindi text summarization. They proposed a combination of binary particle swarm optimization to find the optimal score and generate a final summary. They found that their method performed better than other methods in terms of ROUGE, precision, and recall.

2.4. Abstractive Hindi Text Summarization

In the context of the Hindi language, very little work is done in the area of abstractive text summarization. (Karmakar et al., 2021) proposed to summarize Hindi text using Seq2Seq neural network with attention. They summarized Hindi and Marathi texts using the proposed method. Their model was able to accept the text in Hindi and Marathi text and produce the summary. (Kumari & Singh, 2022) proposed Hindi text summarization using Seq2Seq neural network. After pre-processing the data and word embedding, they applied Seq2Seq neural networks. They used two optimizers i.e., RMSprop optimizer and Adam optimizer. Using their proposed method, they achieved values of precision, recall, and F-measure as 79.23%, 72.92%, and 72.95% respectively. They found that the Adam optimizer was superior to the RMSprop optimizer. (Tangsali et al., 2022) proposed a deep learning-based approach for Indian language text summarization. For Hindi text summarization, they used indicBART, XL-Sum, and mBART. They achieved the highest 0.55 ROUGE_1 score with fine-tuned indicBART.

3. METHODOLOGY

In the proposed work, transliteration and fine-tuning-based Hindi text summarization is proposed. At first, the Hindi text has been transliterated into English. After that, a pre-trained model is fine-tuned on the data. For summarization, we have used fastai and blurr libraries for fine-tuning the pre-trained model Facebook/bart-large-cnn. For performance comparison, along with the ROUGE score and BERT score, a semantic similarity-based performance measure has also been used. Besides calculating performance measures, summaries are also evaluated by humans.

3.1. Dataset

For Hindi text summarization, a few datasets are available, but none are specifically for Hindi text summarization. Like XL-Sum (Hasan et al., 2021) and Cross-Sum(Bhattacharjee et al., 2022) datasets are benchmark multilingual datasets for summarization containing Hindi articles and summaries. But the XL-Sum dataset has very fewer instances of Hindi-to-Hindi articles and summary pairs i.e. only 51,715 pairs. The Cross-Sum dataset contains Hindi-to-other-language articles and summary pairs, but no Hindi-to-Hindi articles and summary pairs. Therefore, we couldn't consider these datasets for the proposed work. Instead, we used a publicly available dataset (*Hindi Text Short and Large Summarization Corpus | Kaggle*, n.d.). This dataset contains articles, headlines, and summaries collected from Hindi news websites. This dataset contains two files: train data and test data. Train data contains 143867 pairs of articles, headlines, and summaries for training, and test data contains 35968 pairs of articles, headlines, and summaries for training. In this dataset, Hindi text is written in Unicode. Table 1 shows dataset statistics.

Table 1. Dataset statistics

Columns	Avereage number of words	
	Train Data (Total number of instances-143867)	Test Data (Total number of instances-35968)
Headline	13.54	13.57
Summary	29.37	29.43
Articles	537.60	543.72

For the proposed experiment, we have used two combinations of columns: (1) articles and headlines for generating headlines and (2) articles and summaries for generating short summaries. For pairs of headline and article, there were no missing values; therefore, we have used all instances of train data and test data for training and testing the model, but the summary column contains many missing values thus, for the article and summary columns, we have removed missing values for training and testing. Table 2 shows statistics of summary and article data after removing missing values.

Table 2. Statistics of summary and article data after removing missing values

Columns	Average number of words	
	Train Data (Total number of instances-69033)	Test Data (Total number of instances-17265)
Summary	29.37	29.43
Articles	537.60	543.72

3.2. Transliteration

The process of converting a word from one language's alphabet to another is called transliteration. Transliteration assists people in pronouncing words and names in foreign languages. A transliteration, unlike a translation, offers you an idea of how the word is spoken by placing it in a familiar alphabet. It replaces letters from the original alphabet with similar-sounding letters from another alphabet. Following are reasons for transliteration of Hindi text before summarization.

- Hindi is a very complex language due to *matras,* and some special symbols. Therefore, tokenization of Hindi sentences is difficult. Sometimes existing tokenizers don't tokenize Hindi sentences properly. Consequently, rather than tokenizing the complex language sentences we have transliterated the Hindi text into English text.
- Most of the pre-trained models for the English language are already trained on huge quantity of English text, whereas pre-trained models for the Hindi language have been trained on less Hindi text. It is already known that the models which are trained on huge data perform better than the models that are trained on fewer data. For the English language, many pre-trained models are available. Thus, in the proposed work, we have used a pre-trained model trained on English text rather than on Hindi text.
- Transliteration can help in text summarization by converting text from one script or writing system into another. This makes it possible to work with multiple language scripts in a single summarization model. For example, if a text summarization model is trained on a certain script, it may not perform well on texts written in another script. Transliterating these texts into the script on which the model is trained can improve its performance.
- Additionally, transliteration helps to overcome language barriers and make text written in different scripts accessible to a wider audience.

For example, original Hindi text "दिल्ली में डीजल टैक्सियों पर बैन से मुश्किल में पड़ा चुनाव आयोग" transliterated in English as "dilli mem dijala taiksiyom para baina se mushkila mem pada chunav aayoga".

3.3. Fastai

The PyTorch-based deep learning library fastai (Howard & Gugger, 2020) provides a simple interface for creating and training various deep learning models. For tasks like image classification, object detection, and language modeling, it offers a wide range of functionality and pre-trained models that can be customized for particular use cases. The foundation of Fastai's architecture is its data block API, which offers a method for quickly preparing and preprocessing data for deep learning model training. With this API, users can specify the structure of their data as well as how it should be transformed and set up for training. Fastai has a four-layered architecture. The low-level API layer contains the basic building blocks of deep learning models such as tensors, auto-grad, pipelines, reversible transforms, and optimized operations. The mid-level API layer contains core deep learning and data processing methods like callback, generic optimizer, generic metric, and data core. The high-level API layer contains learner and data blocks. Beginners and professionals who are primarily interested in using existing deep learning techniques will likely find the high level of the API to be most helpful. Architecture, an optimizer, and data are combined in the learner class, which automatically selects a suitable loss function. The upper application API layer specified application areas like vision, text, tabular, and time series data.

3.4. Blurr Library

The blurr library (*Ohmeow-Blurr · PyPI*, n.d.) is designed specifically for developers who use both fast.ai and Hugging Face Transformers. It essentially acts as a bridge between these two frameworks, providing a comprehensive and easy-to-use toolkit for training, evaluating, and deploying transformer models with fast.ai. So, if you're already familiar with fast.ai and want to leverage the power of transformers, then ohmeow-blurr can be a valuable tool.

3.5. Pre-trained Model

In the proposed work, we have used Facebook/BART-large-cnn(Lewis et al., 2019) pre-trained model. BART is a denoising autoencoder used to pre-train sequence-to-sequence models. To train BART, the text is first corrupted with a random noise function. Then a model is learned to recreate the original text, as shown in Fig. 2. It uses typical Transformer-based neural machine translation architecture, which, despite its simplicity, may be considered as generalizing BERT (due to the bidirectional encoder), GPT (with the left-to-right decoder), and numerous other more contemporary pretraining approaches. In the base BART model, many noising techniques have been used for pre-training like token masking, token deletion, token infilling, sentence permutation, and document rotation. In the BART-large model, they used 12 layers of encoder and decoder. In their encode-decoder architecture, they used GeLU (Hendrycks & Gimpel, 2016) activation function with parameter initialization from N (0, 0.02).

The version of the BART model used in the proposed work is BART-large-cnn model. This BART-Large model is trained on CNN-dailymail dataset with 406 million parameters. The CNN / DailyMail Dataset is an English-language dataset including around 300,000 unique news articles published by CNN and Daily Mail writers. So it can be easily said that facebook/bart-large-cnn is trained on sufficiently significant amounts of data to perform abstractive summarization tasks.

3.6. Proposed Method to Train and Test the Model

In the proposed method, at first train data in Hindi Unicode is transliterated into English. After that, we downloaded the Facebook/ BART-large-cnn model using fastai and blurr libraries. After batch creation, fine-tuning parameters were specified. Then the model was trained for two epochs. The whole process of training and testing is shown in Fig. 3. Following are step-by-step details of the training and testing phases.

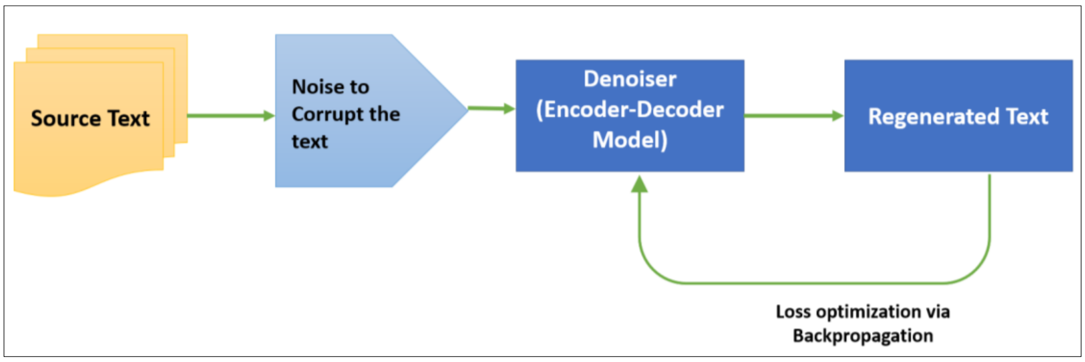
Source Text
Noise to Corrupt the text
Denoiser (Encoder-Decoder Model)
Regenerated Text
Loss optimization via Backpropagation

Fig 2. Basic working of BART model.

3.6.1. Training Phase

Following are detailed steps for training the model

i. Load the training data.

ii. Remove the rows containing missing values.

iii. Transliterate the train data.

iv. After transliteration of the training data, get the hugging face object with the help of blurr.get_hf_object method. While getting the hugging face object using model_CLS method, the original model, its architecture, tokenizer, and config is downloaded. Here we have used BartForConditionalGeneration as model_cls and pre-trained model facebook/bart-large-cnn.

v. Specify fine-tuning parameters.

vi. Define batch. For creating batches, the Seq2SeqBatchTokenizeTransform instance has been created by passing hf object received from blurr.get_hf_object method.

vii. Create data blocks. The data block is a blueprint that describes how raw data is converted to data that can be modeled. Blurr also provides support for working with multiple data blocks in parallel, allowing for the efficient processing of large amounts of data in real-time. For creating the data block, HF_Seq2SeqBlock method has been used.

viii. Pass the data using datablock.dataloaders for creating an actual block of the data.

ix. Create an instance of BaseModelWrapper class by passing the hf_model object. BaseModelWrapper is a class from the Hugging Face Transformers library, which provides a high-level API for working with pre-trained language models. The class is designed to wrap a pre-trained language model and provide an easy-to-use interface for performing common natural language processing tasks, such as text classification, sentiment analysis, and question answering.

x. Define the list of callbacks. A callback in fastai is a piece of code that can be executed at certain points during the training process, such as after the completion of an epoch or after the prediction of a batch of data. Callbacks can be used to modify training process behavior or to perform additional tasks, such as logging, early stopping, and model checkpointing.

xi. Create an instance of the learner class by passing the data block, model object, optimizer function, loss function, list of callbacks, and splitter. Data blocks are blocks that were created in step vii and the model is an instance of the model that will be retrieved as a result in step ix. In the proposed method a ranger optimizer and CrossEntropyLossFlat loss function have been used. The list of callbacks is all those callbacks that were defined in step x. The splitter function splits the model into groups of layers for different learning rates. In the proposed method, the seq2seq_splitter function is used, with the arch argument set to hf_arch. In the proposed method, the to_fp16 method was also used to enable the model to use half-precision floating-point numbers, which can speed up the training on GPUs with limited memory.

xii. Create an instance of optimizer using learn.create_opt() method.

xiii. Freeze the model, so that only the last layer's weight can be updated during the training process. Use learn.lr_find() to find the optimal learning rate.

xiv. Train the model. In the proposed method, we have trained our frozen model for two epochs.

3.6.2. Testing phase

Following are steps for testing the model

i. Transliterate the test data.

ii. Generate the output using learn.blurr_generate method.

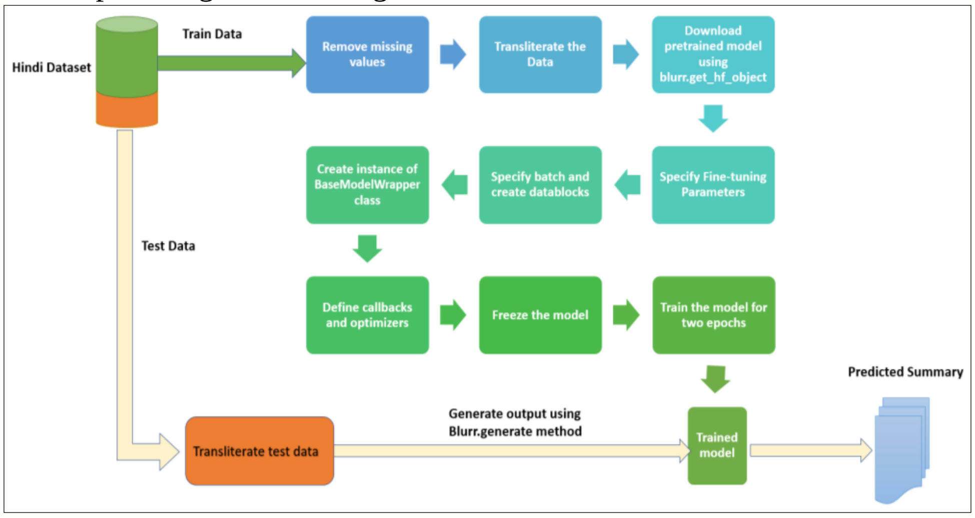
Hindi Dataset
Train Data
Remove missing values
Transliterate the Data
Download pretrained model using blurr.get_hf_object
Specify Fine-tuning Parameters
Specify batch and create datablocks
Create instance of BaseModelWrapper class
Test Data
Define callbacks and optimizers
Freeze the model
Train the model for two epochs
Predicted Summary
Generate output using Blurr.generate method
Transliterate test data
Trained model

Fig 3. Block diagram of the proposed method.

3.7. Performance Matrices

For the performance measurement of the proposed method, ROUGE score, and BERT score have been calculated. Along with these traditional performance comparison methods, we have also used semantic similarity-based metrics to assess the performance of the model.

3.7.1. ROUGE score

It compares an automatically generated summary to a reference or gold-standard summary and scores the similarity between the two. ROUGE is often used in the evaluation of text summarization models (Lin, 2004), as well as machine translation models, image captioning models, and other tasks involving the generation of text. There are several variations of the ROUGE metric, including ROUGE_N, ROUGE_L, and ROUGE_W, which measure the overlapping n-grams, longest common sequences, and weighted overlapping of words, respectively. The choice of the ROUGE metric depends on the task and the desired evaluation criteria. But some disadvantages are also associated with the ROUGE score, like

- ROUGE score focuses on recall and is limited in evaluating the content, meaning, and style of the generated summary, as it only considers overlapping n-grams between the reference summary and the generated summary.
- ROUGE is biased towards longer n-grams, which might result in a higher score for summaries that repeat parts of the reference summary rather than generating new and concise information.
- ROUGE score heavily depends on the quality of the reference summaries. This might not reflect human summary quality if the reference summaries are poorly written.
- ROUGE is sensitive to the length of the summaries, which might result in a higher score for longer summaries, even if they contain redundant information.

3.7.2. BERT score

It is a text generation evaluation metric based on the pre-trained language model BERT (Zhang et al., 2019). Unlike traditional evaluation metrics such as BLEU and ROUGE, BERT Score evaluates the similarity of the generated text to the reference text. This is done by considering both the context and the meaning of the words in the text. It also calculates the cosine similarity between the contextualized representations of the generated text and the reference text, which are obtained from the BERT model. The final BERT score is the average of the pair-wise cosine similarities between all word pairs in the generated and reference texts. BERT scores are context-aware and have less reference dependency. BERT score has some limitations as well, like high computational cost and limited coverage.

3.7.3. Semantic Similarity-based Metric

We have also proposed a performance metric to measure the semantic similarity of reference summary and predicted summary. Semantic similarity refers to the degree of relatedness or similarity between the meanings of two or more pieces of text. This is typically measured using computational techniques based on NLP and machine learning. Following is the proposed method (as shown in Fig 4) to calculate semantic similarity between reference summary and predicted summary.

- Encode the reference summary and predicted summary using a sentence transformer. In the proposed method, we have used an all-MiniLM-L6-v2 sentence transformer (Reimers & Gurevych, 2019). It converts sentences and paragraphs into a 384-dimensional dense vector space that can be used for tasks such as clustering and semantic search.
- Find cosine similarity between each pair of reference summary and predicted summary. Cosine similarity is frequently used in NLP to compare the similarity of two document vectors or word embedding. It can be used to perform tasks like document retrieval, clustering, and recommendation systems. It computes the cosine of the angle between the vectors and has a

value ranging from -1 (completely dissimilar) to 1 (identical), with 0 indicating orthogonality (no correlation).

- Convert tensor values to scalars.
- Replace the cosine score for each summary pair with 1 if the cosine score is greater than 0.50 otherwise, replace it with 0. Convert all scores into one list D1.
- Take another list D2 that has all ones, and the same length as D1.
- Find the percentage similarity between D1 and D2. This percentage similarity shows the value of the similarity-based metric.

4. RESULTS

For training the text summarization model, we have used fastai along with the blurr library. The dataset we have used in our experiments has three columns i.e. headline, summary, and article. So, we have trained and fine-tuned the pre-trained model for the following two tasks i.e. generate the short summaries of the articles and generate the headlines of the articles.

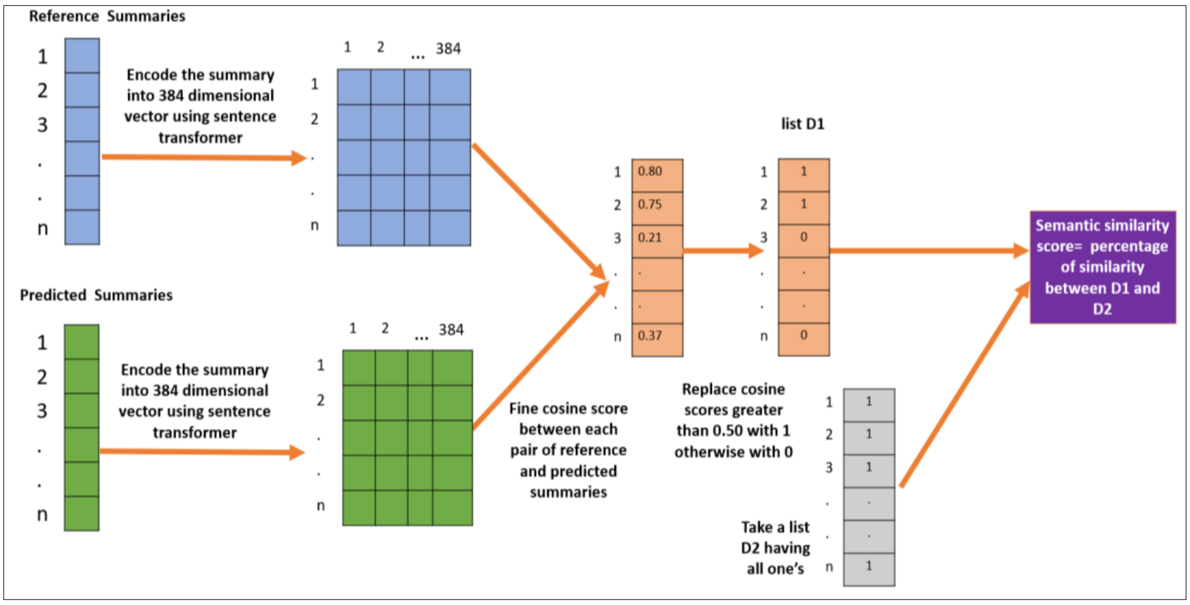
Reference Summaries
Encode the summary into 384 dimensional vector using sentence transformer
Predicted Summaries
Encode the summary into 384 dimensional vector using sentence transformer
1 2 ... 384
list D1
0.80
0.75
0.21
0.37
Fine cosine score between each pair of reference and predicted summaries
Replace cosine scores greater than 0.50 with 1 otherwise with 0
Take a list D2 having all one's
Semantic similarity score= percentage of similarity between D1 and D2

Fig 4. Block diagram of semantic similarity score calculation.

4.1. Generating short summaries of articles

For generating short summaries of the articles, we have used two columns of the dataset i.e. summary and article for training the model. For this, a pre-trained model Facebook/bart-large-cnn has been fine-tuned on the dataset. After training and fine-tuning, we have obtained results as shown in Table 3.

Table 3. Results obtained during each epoch of training for article and summary pairs.

Epoch	Train Loss	Validation Loss	ROUGE_1	ROUGE_2	ROUGE_L	BERT score precision	BERT score recall	BERT score f1
0	1.097	1.042	0.473	0.336	0.414	0.806	0.821	0.813
1	0.834	0.967	0.476	0.338	0.416	0.807	0.822	0.814

From Table 3, we can see that at epoch 0, training loss and validation loss were 1.097 and 1.042, respectively. ROUGE_1 was 0.472, ROUGE_2 was 0.336 and ROUGE_L was 0.441. At epoch 0, precision, recall and F1 score of BERT score were 0.806, 0.821, and 0.813. At epoch 1, all the performance measures were slightly increased and losses were slightly decreased i.e. at epoch 1, training loss and validation loss were 0.834 and 0.967 respectively. ROUGE_1 was 0.476, ROUGE_2 was 0.338, and ROUGE_L was 0.416. At epoch 1, precision, recall, and F1 score of BERT score were 0.807, 0.822, and 0.814. Epoch 0 took approximately 6 hours and 10 minutes for training, and epoch1 took approximately 6 hours and 22 minutes. For generating short summaries from articles, we have used test data having 17265 instances. After testing ROUGE score and BERT score have been obtained as shown in Table 4.

Table 4. ROUGE and BERT score of reference summaries and generated summaries in test data

Performance measure	Recall	Precision	F-score
ROUGE_1 score	0.5510	0.4319	0.4749
ROUGE_2 score	0.3748	0.3033	0.3279
ROUGE_L score	0.5028	0.3978	0.4358
BERT score	0.80	0.82	0.81

.

We have obtained 0.55 recall, 0.43 precision, and 0.47 F-score values of ROUGE_1; 0.37 recall, 0.30 precision, and 0.33 F-score values of the ROUGE_2 score; 0.50 recall, 0.40 precision, and 0.44 F-score values of ROUGE_L score; and 0.80 recall, 0.82 precision, and 0.81 F-score value of BERT score.

We also proposed and used a semantic similarity-based score to measure the similarity between the reference summaries and generated summaries. For this, we have calculated the cosine similarity between two sentences. We found following:

Number of rows for which cosine score is greater than 0.50	-	16956
Number of rows for which cosine score is less than 0.50	-	309

After replacing the cosine score greater than 0.50 with 1 and the cosine score less than 0.50 with 0, we calculated classification accuracy and found the following classification matrix:

	Precision	Recall	F1-Score	Support
0	0.00	0.00	0.00	0
1	1.00	0.98	0.99	17265
Accuracy			0.98	17265
Macro average	0.50	0.49	0.50	17265
Weighted average	1.00	0.98	0.99	17265

The similarity score obtained between reference summaries and generated summaries is 0.98. So, we can consider that there is a good level of similarity between the reference summaries and generated summaries.

4.2. Generating headlines from articles

For generating the headline of the article, we have used two columns of the dataset i.e. headline and article for model training. Here we have used the same model that was used in section 5.1. After training and fine-tuning, we have obtained results as shown in Table 5.

Table 5. Results obtained during each epoch of training for article and headline pairs.

Epoch	Train Loss	Validation Loss	ROUGE_1	ROUGE_2	ROUGE_L	BERT score precision	BERT score recall	BERT score f1
0	1.495	1.501	0.310	0.132	0.250	0.742	0.783	0.762
1	1.358	1.380	0.317	0.138	0.258	0.745	0.786	0.764

From Table 5, we can see that at epoch 0, training loss and validation loss were 1.495 and 1.501, respectively. ROUGE_1 was 0.310, ROUGE_2 was 0.132, and ROUGE_L was0.250. At epoch 0, precision, recall and F1 score of the best score were 0.742, 0.783, and 0.762. At epoch 1, all the performance measures were slightly increased and losses were slightly decreased i.e. on epoch 1, training loss and validation loss were 1.358 and 1.380, respectively. ROUGE_1 was 0.317, ROUGE_2 was 0.138 and ROUGE_L was 0.258. At epoch 1, precision, recall, and F1 score of BERT score were 0.745, 0.789, and 0.764. Epoch 0 took approximately 9 hours and 35 minutes for training and epoch 1 took approximately 10 hours and 6 minutes for training.

For generating headlines from the article, we have used the same test data as used in Section 5.1 for summaries generation but used article and headline columns. After testing, we obtained the ROUGE score and BERT score as shown in Table 6.

Table 6. ROUGE and BERT score of reference headline and generated headline pairs in test data.

Performance measure	Recall	Precision	F-score
ROUGE_1 score	0.4274	0.2331	0.2948
ROUGE_2 score	0.1811	0.0904	0.1172
ROUGE_L score	0.3655	0.1993	0.2520
BERT Score	0.79	0.74	0.76

We have obtained 0.43 recall, 0.23 precision, and 0.29 F-score values of ROUGE_1; 0.18 recall, 0.09 precision, and 0.11 F-score values of the ROUGE_2 score; 0.36 recall, 0.20 precision, and 0.25 F-score values of ROUGE_L score; and 0.79 recall, 0.74 precision, and 0.76 F-score value of BERT score.

While calculating semantic similarity, we found following:

Number of rows for which cosine score is greater than 0.50	-	16956
Number of rows for which cosine score is less than 0.50	-	309

After replacing the cosine score greater than 0.50 with 1 and the cosine score less than 0.50 with 0, we calculated classification accuracy and found following classification matrix.

	Precision	Recall	F1-Score	Support
0	0.00	0.00	0.00	0
1	1.00	0.95	0.98	17265
Accuracy			0.95	17265
Macro average	0.50	0.48	0.49	17265
Weighted average	1.00	0.95	0.98	17265

The similarity score obtained between pairs of reference headlines and predicted headlines is 0.95. Table 7 and Table 8 show samples of generated summaries and Headlines.

Table 7. Reference summary and predicted summaries.

Original Summary	Transliterated Summary	Predicted Transliterated Summary	Predicted Hindi Summary
जिमनास्टिक में कालीफाई	jimanastika me kvaliphai karane vali pehali	olimpika ke lie kvaliphai kara itihasa ka pahala adhyaya	ओलम्पिक के लिए कालीफाई कर इतिहास

करने वाली पहली भारतीय महिला हैं दीपा रियो में ही ओलिंपिक के लिए त्रिपुरा की इस लड़की ने किया था क्वालिफाई कोच ने कहा- कभी भी आसानी से हार नहीं मानती है दीपा	bharatiya mahila hai dipa riyo me hi olimpika ke liye tripura ki is ladaki ne kiya tha kvaliphai kocha ne kaha- kabhi bhi asani se haar nahim manati hai dipa	likha chuki dipa karmakara ravivara ko jaba riyo khelom mem utaremgi. khela mem padaka jitane vali pahali bharatiya mahila jimanasta haim. maharata haasil karane ke lie unhomne apana saba kucha jhomka diya aura vaha apane pradarshana ke lie usa para vishvasa kara rahi haim	का पहला अध्याय लिखा चुकी दीपा कर्मकार रविवार को जब रियो खेलो में उतरेंगी। खेल में पदक जीतेने वाली पहली भारतीय महिला जिमनास्ता है। महारत हासिल करने के लिए उन्हें अपना सब कुछ झोंका दिया और वहा अपने प्रदर्शन के लिए हम पर विश्वास कर रही है
लखनऊ से लेकर दिल्ली तक होगा शक्ति प्रदर्शन जन्मदिन का निमंत्रण सहयोगी दलों को भी भेजा गया अखिलेश के साथ सीट बंटवारे पर हो सकता है एलान	lakhanau se lekara dilli taka hoga shakti pradarshana janmadina ka nimamtrana sahayogi dalom ko bhi bheja gaya akhilesha ke satha sita bamtavare para ho sakata hai elana	basapa pramukha mayavati ki railiyom ka karyakrama bhi taya ho chuka hai. karyakartao ne kai jagaha para unake hordimga aura postar lagaye haim jisamem unake janmadina ki badhaiyom ke samdesha diye gaye haim janmadina lakhanau ke satha-satha dilli mem bhi manane ka phaisala liya gaya hai	बसपा प्रमुख मायावती की रेलियों का कार्यक्रम भी तय हो चूका है। कार्यकर्ताओं ने काई जगह पर उनेक होर्डिमगा औरा पोस्टर लगाये हैं उसी तरह उनके जन्मादिन की बढ़ाइयों के समदेश दिए गए हैं जन्मदीना लखनौ के साथ-साथ दिल्ली में भी मनाने का फैसला लिया गया है

Table 8. Reference headlines and predicted headlines.

Original Headline	Transliterated Headline	Predicted Transliterated Headline	Generated Headline
पवन सिंह से आम्रपाली दुबे ने की शिकायत 'शिव मानत नहीं', Video हुआ वायरल	pavana simha se amrapali dube ne ki shikayata 'shiva manata nahim', video hua vayarala	amrapali dube ne pavana simha ke satha milakara yutyuba para machaya dhamal, bar-bar dekha ja raha hai video, jane kya hai vajaha...	आम्रपाली दुबे ने पवन सिन्हा के साथ मिलकर युत्युब पर मचाया धमाल, बार-बार देखा जा रहा है वीडियो जाने क्या है वजह...
रियो ओलिंपिक : यहीं पर ओलिंपिक में स्थान बनाया था, इसी जगह पर क्या	riyo olimpika : yahim para olimpika mem sthana banaya tha, isi jagaha para	jimanastika me olimpika ke lie kvaliphai kara itihasa ka pahala adhyaya likha chuki dipa karmakara ravivara ko jaba riyo khelom mem	जिमनास्तिका में ओलम्पिका के लिए क्वालीफाई कर इतिहास का पहला अध्याय लिखा चूकी

पदक जीत पाएंगी दीपा...	kiya padaka jita paemgi dipa...	utaremgi to nigahem nai umchai ko chhune para lagi hongi	दीपा कर्मकार रविवार को जब रियो खेलों मे उतरेगी तो निगाहे नई ऊंचाई को छूने पर लगी होंगी

4.3. Human Evaluation of Generated Summaries

Human assessment of summaries is an essential part of NLP and automated summarization. It entails having human assessors read and grade the quality of machine-generated summaries to assess their correctness, readability, coherence, and general usefulness. Human review can take many forms, such as evaluating summaries on a scale of 1 to 5, offering input on specific features of the summary, or comparing several summaries to determine which one is best. Human assessment is used to gain a reliable and accurate gauge of the summary's quality and to suggest areas for improvement. In the proposed work, summaries have been rated in terms of consistency, fluency, coherence, and relevance(Fabbri et al., 2021). Consistency refers to the uniformity and coherence of the text in terms of style, tone, grammar, punctuation, and spelling. Inconsistent use of these elements can lead to confusion and distract the reader from the message being conveyed. Fluency, on the other hand, refers to the ease with which the text can be read and understood by the reader. Coherence refers to the overall organization and structure of the text, including the logical flow of ideas and the use of transitional phrases and other devices to link sentences and paragraphs. Relevance refers to the extent to which text content is appropriate and useful to its intended audience. In the proposed work, human evaluation of predicted summaries is also done. For human evaluation, we have selected twelve independent experts. All these experts were highly educated (having postgraduate or Ph.D. degrees) and belonged to different subjects. For evaluation, eighty pairs of articles, generated summaries, and generated headlines are selected randomly. Each article, summary, and headline pair is evaluated by three independent experts. All experts rated the generated headline and summaries in terms of consistency, fluency, coherence, and relevance. All these parameters were ranked on a scale of 1 to 5 (1-lowest rating, 5 – Highest rating). While evaluating generated summaries, few incorrect *matras* were ignored as a limitation of the reverse transliteration. The average ranking of generated summaries and generated headlines in terms of consistency, fluency, coherence, and relevance is shown in Table 9. We have also calculated the number of generated summaries and headlines in different ranges of ranking as shown in Table 10.

Table 9. Average ranking by human experts.

Case	Consistency	Fluency	Coherence	Relevance	Overall Score
Summaries Generation	3.7070	3.4783	3.6804	3.7612	3.6567
Headlines Generation	3.4375	3.3733	3.4629	3.4837	3.4393

Table 10. Number and percentage of summaries and headlines in different ranges of rating.

Case	**Rating range**	**Consistency No.(%)**	**Fluency No.(%)**	**Coherence No.(%)**	**Relevance No.(%)**	**Overall No.(%)**

Generation of Summaries from Articles	2 or below	3 (4%)	3(4%)	2 (3%)	2 (3%)	2 (3%)
	2< and <=3	9 (11%)	19 (24%)	14 (18%)	14 (18%)	7 (9%)
	3< and <=4	51 (64%)	50 (63%)	46 (58%)	44 (55%)	53 (65%)
	Above 4	17 (21%)	8 (10%)	18 (23%)	20 (25%)	18 (23%)
Generation of Headlines from Articles	2 or below	3 (4%)	3 (4%)	3 (4%)	3 (4%)	2 (3%)
	2< and <=3	22 (28%)	30 (38%)	19 (24%)	22 (28%)	20 (25%)
	3< and <=4	46 (58%)	36 (45%)	50 (63%)	42 (53%)	49 (61%)
	Above 4	9 (11%)	11 (14%)	8 (10%)	13 (16%)	9 (11%)

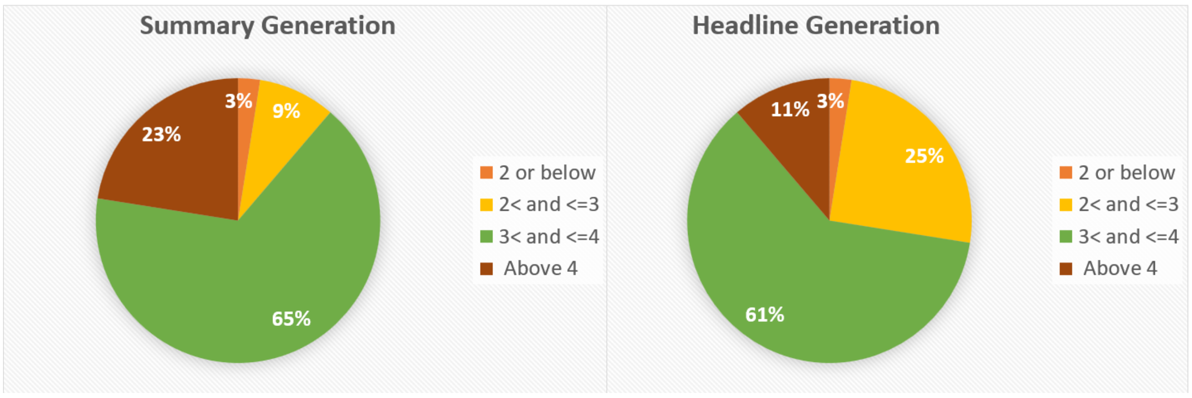

Fig 5. Summaries and Headlines in different ranges of rating.

From above Table 9, it can be seen that in the case of summary generation from articles, achieved average ratings in terms of consistency, fluency, coherence, and relevance are 3.7070, 3.4783, 3.6804, and 3.7612, respectively, and overall rating is 3.6567. In the case of headline generation from articles, achieved average ratings in terms of consistency, fluency, coherence, and relevance are 3.4375, 3.3733, 3.4629, and 3.4837, respectively, and the overall rating is 3.4393. Table 10 and Fig. 5 show that in the case of summary generation from articles, 65% of summaries are rated in the range 3< and <=4, 23% of summaries are rated above 4, 9% of summaries are rated in the range 2< and <=3 and 3% of summaries are rated in the range 2 or below. In the case of headline generation from articles, 61% of headlines are rated in the range 3< and <=4, 25% of headlines are rated in the range 2< and <=3, 11% of headlines are rated above 4, and 3% of headlines are rated in the range 2 or below.

5. Discussion

In this section, we will discuss the results of our experiment to assess the performance and limitations of the proposed system. At first, we will discuss the cases in which our proposed method performs better and after that, we will compare our proposed model with other state-of-the-art models.

In this paper, two cases are considered. In the first case, headlines are generated from articles, and in the second case, short summaries are generated from articles. From Table 5 and Table 6 and Fig. 6, it can be seen that higher ROUGE score, BERT score, and semantic similarity score have been obtained with summary generation from articles. The reason for obtaining lower scores with headline generation could be the very short length of headlines in the data. In Table 1, it is mentioned that the maximum number of words in headlines is 13. It is too short to convey key points of news articles. Sometimes headlines are created by news agencies to create hype among readers. To create hype among readers, headlines are created in such a way that their exact meaning doesn't represent the article's essence, and after reading the article, the exact meaning is realized. But in the case of summaries, most of the summaries are aligned with the article context. In this paper, the average length of summaries is 29 words as shown in Table 1 and Table 2. These summaries are of sufficient length to incorporate most of the key points of the articles.

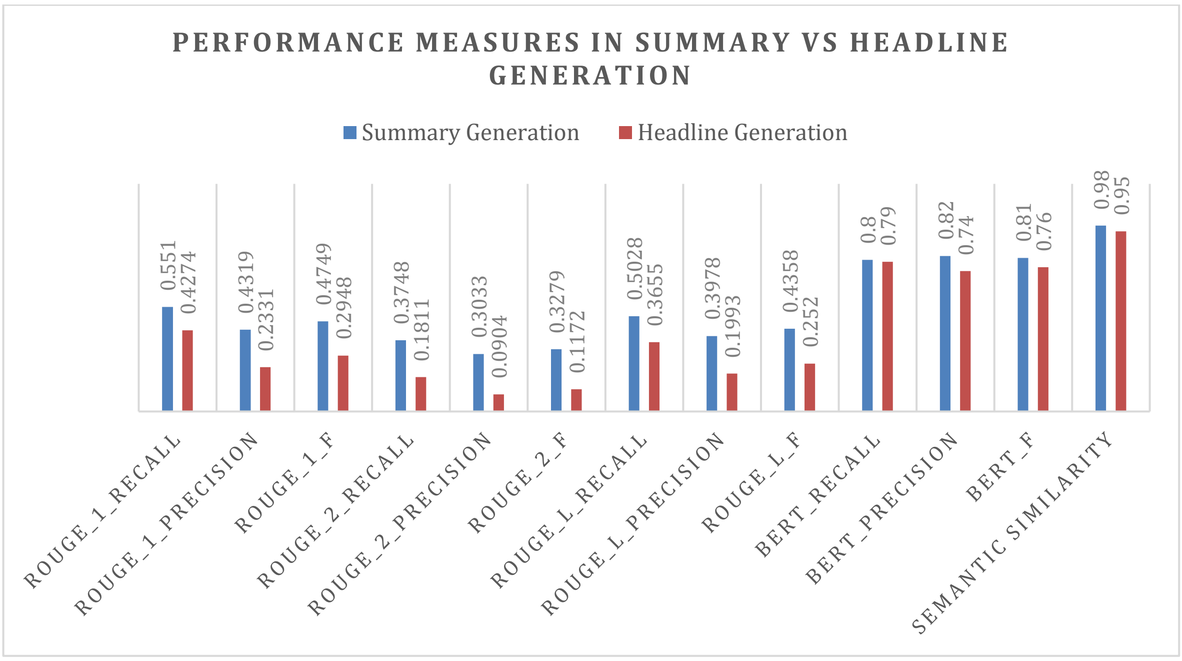

Fig 6. Performance measures in summary vs headline generation.

Human evaluation of randomly selected samples of generated summaries and headlines was also done. From Table 9, it can be seen that the average rating of all evaluation parameters i.e. consistency, fluency, coherence, and relevance is higher with summaries generated from articles than the average rating of headlines generated from articles. The overall rating of summary generation is 3.6567 and the overall rating of headline generation is 3.4394. The reason for achieving higher scores with summary generation is the same reason mentioned in the previous paragraph i.e., most of the time headlines are created by news agencies for publicity among readers and sometimes the exact meaning of headlines also differs from the intended meaning. But in the case of short summaries, publicity is not the agenda, so summaries contain the exact gist of the article. From Table 10, it can be seen that the maximum number of summaries and headlines are rated in the range 3< and <=4 out of 5. It shows that our proposed system generates headlines and summaries at human-acceptable levels.

In the third section of the discussion, we will compare the results of the proposed method with other state-of-the-art methods. Even for Hindi text summarization, no specific standard models are available. But some models, like BART and T5, that are fine-tuned on a large amount of Hindi text can be considered state-of-the-art models. Table 11 shows the comparison of the proposed method with other state-of-the-art methods.

Table 11. Comparison with state-of-the-art methods on Hindi text summarization.

Model/Method used	**Dataset**	**Number of Articles**	**Number of Articles (Hindi)**	**Results**
mT5_multilingual_XL Sum (Hasan et al., 2021a)	XL_Sum	1 M	51K	38.5882 ROUGE-1 16.8802 ROUGE-2 32.0132 ROUGE-L
indicBART (Dabre et al., 2022)	XL_Sum	1 M	51K	31.71 ROUGE_L
mT5_m2m_crossSum (Bhattacharjee et al., 2022)	Cross_Sum	1.7 M	88K	0.967 pearson correlation between ROUGE-2 and Language-agnostic Summary Evaluation (LaSE) for Hindi to other language summarization
Facebook/BART-large-cnn Proposed Method	Hindi text summarization long and short dataset	179K	179K (articles and headlines) 86K (article and summary)	0.5510 ROUGE-1 Score

For comparison, we have selected three state-of-the-art models. Although text summarization models specific to Hindi text are not available, some authors have worked on Hindi text summarization. These models can be considered state-of-the-art models. One of the reasons for not having state-of-the-art models for Hindi text summarization can be the non-availability of benchmark data. Research work 1 is proposed by (Hasan et al., 2021) and they proposed a model mT5_multilingual_XLSum for the Hindi text summarization XL-Sum dataset. XL-Sum is a very big dataset containing data from 40 languages. It contains 51,715 samples of Hindi data. Using the proposed method they achieved a 38.58 ROUGE-1, 16.88 ROUGE-2, and 32.01 ROUGE_L. In 2022, (Dabre et al., 2022) proposed second research work based on the model indicBART for Hindi text summarization on the same XL-Sum dataset. The indicBART model is specially trained on Indian languages but they achieved 31.71 ROUGE_L score which is less than the ROUGE score obtained in research work 1. The third research work is proposed by (Bhattacharjee et al., 2022). They performed cross-lingual summarization on the Cross-sum dataset. Even though the Cross_Sum dataset is very big and contains 88,472 instances of Hindi language text but all instances were for cross-lingual summarization. None of the data instances was for Hindi-to-Hindi summarization. In the proposed work, we have used a publicly available dataset that 143867 records for training and 35268 records for testing. This dataset can be considered large enough for Hindi text summarization. We have achieved the highest 55.10 ROUGE score using the proposed method based on transliteration. One other work was proposed by (Urlana et al., 2023) for Hindi, English, and Gujarati text summarization. But we have not included this work under comparison of state-of-the-art models because even though they claimed a higher ROUGE score for Hindi text summarization, their model was trained on a very small dataset containing only 7958 records for training and 2842 records for testing. Whereas the proposed work model is trained and tested on large data (approximately 188K instances), we can say that even though the ROUGE score of research work proposed by (Urlana et al., 2023) is higher than the proposed work but our proposed model is more robust because it is trained and tested on a large Hindi dataset containing data from various domains like health, sport, medicine, nation, politics, etc.

6. CONCLUSION

Text summarization is the process of reducing a long piece of text to its essence while retaining the most important information and meaning. The goal of text summarization is to condense a large amount of information into a concise and readable summary that is easy to understand. In the proposed work, Hindi text summarization is done through transliteration and fine-tuning of the pre-trained model. A new semantic similarity-based score is also proposed in this work. Using the proposed method, 55.16 ROUGE score, 0.80 BERT score, and 0.98 similarity score have been obtained. Some randomly selected samples of generated summaries and headlines are also evaluated by experts and it is found that the proposed system generates headlines and summaries at human-acceptable levels. We have also compared the performance of the proposed work with other recent research work on Hindi text summarization and found that the proposed model performed better than others.

Statements and Declarations:
Acknowledgment- None
Funding – None
Conflict of Interests/Competing Interests- None

REFERENCES

[1] Ali, S. M., Noorian, Z., Bagheri, E., Ding, C., & Al-Obeidat, F. (2020). Topic and sentiment aware microblog summarization for twitter. *Journal of Intelligent Information Systems*, *54*(1), 129–156. https://doi.org/10.1007/S10844-018-0521-8/TABLES/13

[2] Alsaqer, A. F., & Sasi, S. (2017). Movie review summarization and sentiment analysis using rapidminer. *2017 International Conference on Networks & Advances in Computational Technologies (NetACT)*, 329–335. https://doi.org/10.1109/NETACT.2017.8076790

[3] Aote, S. S., Pimpalshende, A., Potnurwar, A., & Lohi, S. (2023). Binary particle swarm optimization with an improved genetic algorithm to solve multi-document text summarization problem of Hindi documents. *Engineering Applications of Artificial Intelligence*, *117*, 105575. https://doi.org/https://doi.org/10.1016/j.engappai.2022.105575

[4] Arora, G. (2020). inltk: Natural language toolkit for indic languages. *ArXiv Preprint ArXiv:2009.12534*.

[5] Babu, G. L. A., & Badugu, S. (2023). Deep learning based sequence to sequence model for abstractive Telugu text summarization. *Multimedia Tools and Applications*, *82*(11), 17075–17096. https://doi.org/10.1007/s11042-022-14099-x

[6] Bafna, P. B., & Saini, J. R. (2019). Hindi multi-document word cloud based summarization through unsupervised learning. 2009 *9th International Conference on Emerging Trends in Engineering and Technology - Signal and Information Processing (ICETET-SIP-19)*, 1–7. https://doi.org/10.1109/ICETET-SIP-1946815.2019.9092259

[7] Bandari, S., & Bulusu, V. V. (2023). Feature extraction based deep long short term memory for Hindi document summarization using political elephant herding optimization. *International Journal of Intelligent Robotics and Applications*, *7*(1), 103–118. https://doi.org/10.1007/s41315-022-00237-z

[8] Bani-Almarjeh, M., & Kurdy, M. B. (2023). Arabic abstractive text summarization using RNN-based and transformer-based architectures. *Information Processing & Management*, *60*(2), 103227. https://doi.org/10.1016/J.IPM.2022.103227

[9] Bhattacharjee, A., Hasan, T., Ahmad, W. U., Li, Y.-F., Kang, Y.-B., & Shahriyar, R. (2022). *CrossSum: beyond English-centric cross-lingual abstractive text summarization for 1500+ language pairs*. https://arxiv.org/abs/2112.08804v2

[10] Dabre, R., Shrotriya, H., Kunchukuttan, A., Puduppully, R., Khapra, M. M., & Kumar, P. (2022). IndicBART: A pre-trained model for indic natural language generation. *Proceedings*

of the Annual Meeting of the Association for Computational Linguistics, 2, 1849–1863. https://doi.org/10.18653/v1/2022.FINDINGS-ACL.145

[11] Dhankhar, S., & Gupta, M. K. (2022). A statistically based sentence scoring method using mathematical combination for extractive Hindi text summarization. *Journal of Interdisciplinary Mathematics, 25*(3), 773–790. https://doi.org/10.1080/09720502.2021.2015096

[12] Dhivyaa, C. R., Nithya, K., Janani, T., Kumar, K. S., & Prashanth, N. (2022). Transliteration based generative pre-trained transformer 2 model for tamil text summarization. *2022 International Conference on Computer Communication and Informatics (ICCCI),* 1–6. https://doi.org/10.1109/ICCCI54379.2022.9740991

[13] Durga, C. B. V., & BABU, D. (2022). Telugu text summarization using histo fuzzy c-means and median support based grasshopper optimization algorithm (MSGOA). *Journal of Theoretical and Applied Information Technology*, *100*(17), 5418–5432.

[14] El-Kassas, W. S., Salama, C. R., Rafea, A. A., & Mohamed, H. K. (2021). Automatic text summarization: A comprehensive survey. *Expert Systems with Applications*, *165*, 113679. https://doi.org/10.1016/J.ESWA.2020.113679

[15] Fabbri, A. R., Kryściński, W., McCann, B., Xiong, C., Socher, R., & Radev, D. (2021). Summeval: Re-evaluating summarization evaluation. *Transactions of the Association for Computational Linguistics*, *9*, 391–409. https://doi.org/10.1162/TACL_A_00373/100686/SUMMEVAL-RE-EVALUATING-SUMMARIZATION-EVALUATION

[16] Garg, K. D., Khullar, V., & Agarwal, A. K. (2021). Unsupervised machine learning approach for extractive Punjabi text summarization. *Proceedings of the 8th International Conference on Signal Processing and Integrated Networks, SPIN 2021*, 750–754. https://doi.org/10.1109/SPIN52536.2021.9566038

[17] Hasan, T., Bhattacharjee, A., Islam, M. S., Samin, K., Li, Y. F., Kang, Y. Bin, Rahman, M. S., & Shahriyar, R. (2021). XL-Sum: Large-scale multilingual abstractive summarization for 44 languages. *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, 4693–4703. https://doi.org/10.18653/V1/2021.FINDINGS-ACL.413

[18] Hendrycks, D., & Gimpel, K. (2016). Gaussian error linear units (GeLUs). *ArXiv Preprint ArXiv:1606.08415*. https://doi.org/10.48550/arXiv.1606.08415

[19] *Hindi Text Short and Large Summarization Corpus | Kaggle*. (n.d.). Retrieved March 12, 2023, from https://www.kaggle.com/datasets/disisbig/hindi-text-short-and-large-summarization-corpus

[20] Howard, J., & Gugger, S. (2020). Fastai: A layered API for deep learning. *Information, 11*(2). https://doi.org/10.3390/info11020108

[21] Jain, A., Arora, A., Morato, J., Yadav, D., & Kumar, K. V. (2022). Automatic text summarization for Hindi using real coded genetic algorithm. *Applied Sciences, 12*(13). https://doi.org/10.3390/app12136584

[22] Jain, A., Arora, A., Yadav, D., Morato, J., & Kaur, A. (2021). Text summarization technique for Punjabi language using neural networks. *International Arab Journal of Information Technology*, *18*(6), 807–818. https://doi.org/10.34028/IAJIT/18/6/8

[23] Jain, A., Yadav, D., & Arora, A. (2021). Particle swarm optimization for Punjabi text summarization. *International Journal of Operations Research and Information Systems (IJORIS), 12*(3), 1–17. https://doi.org/10.4018/IJORIS.20210701.oa1

[24] Jaya, A., Ganesh, A., & Rahman, Bsa. (2019). Automatic summarization of Malayalam documents using clause identification method. *International Journal of Electrical and Computer Engineering (IJECE),* *9*(6), 4929–4938. https://doi.org/10.11591/ijece.v9i6.pp4929-4938

[25] Joshi, M. L., Joshi, N., & Mittal, N. (2021). SGATS: semantic graph-based automatic text summarization from Hindi text documents. *ACM Trans. Asian Low-Resour. Lang. Inf. Process.*, *20*(6). https://doi.org/10.1145/3464381

[26] K. Nambiar, S., Peter S., D., & Mary Idicula, S. (2023). Abstractive summarization of text document in Malayalam language: enhancing attention model using pos tagging feature. *ACM Trans. Asian Low-Resour. Lang. Inf. Process.*, *22*(2). https://doi.org/10.1145/3561819

[27] Karmakar, R., Nirantar, K., Kurunkar, P., Hiremath, P., & Chaudhari, D. (2021). Indian regional language abstractive text summarization using attention-based LSTM neural network. *2021 International Conference on Intelligent Technologies (CONIT)*, 1–8. https://doi.org/10.1109/CONIT51480.2021.9498309

[28] Kumari, N., & Singh, P. (2022). *Hindi Text Summarization using Sequence to Sequence Neural Network*. https://doi.org/10.21203/RS.3.RS-2036546/V1

[29] Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., & Zettlemoyer, L. (2019). BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *ArXiv Preprint ArXiv:1910.13461*. https://doi.org/10.48550/arXiv.1910.13461

[30] Lin, C.-Y. (2004). ROUGE: A package for automatic evaluation of summaries. *Text Summarization Branches Out*, 74–81. https://aclanthology.org/W04-1013

[31] Luhn, H. P. (1958). The automatic creation of literature abstracts. *IBM Journal of Research and Development*, *2*(2), 159–165. https://doi.org/10.1147/rd.22.0159

[32] Mamidala, K. K., & Sanampudi, S. K. (2021). A heuristic approach for Telugu text summarization with improved sentence ranking. *Turkish Journal of Computer and Mathematics Education (TURCOMAT)*, *12*(3), 4238–4243. https://doi.org/https://doi.org/10.17762/turcomat.v12i3.1714

[33] Manjari, K. U. (2020). Extractive summarization of Telugu documents using textrank algorithm. *2020 Fourth International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*, 678–683. https://doi.org/10.1109/I-SMAC49090.2020.9243568

[34] Nambiar, S. K., Peter S, D., & Idicula, S. M. (2021). Attention based abstractive summarization of malayalam document. *Procedia Computer Science*, *189*, 250–257. https://doi.org/https://doi.org/10.1016/j.procs.2021.05.088

[35] Nawaz, A., Bakhtyar, M., Baber, J., Ullah, I., Noor, W., & Basit, A. (2020). Extractive text summarization models for Urdu language. *Information Processing & Management*, *57*(6), 102383. https://doi.org/10.1016/J.IPM.2020.102383

[36] *ohmeow-blurr · PyPI*. (n.d.). Retrieved March 12, 2023, from https://pypi.org/project/ohmeow-blurr/

[37] Rani, R., & Lobiyal, D. K. (2021). An extractive text summarization approach using tagged-LDA based topic modeling. *Multimedia Tools and Applications*, *80*(3), 3275–3305. https://doi.org/10.1007/S11042-020-09549-3/METRICS

[38] Reimers, N., & Gurevych, I. (2019). Sentence-BERT: Sentence embeddings using siamese BERT-networks. *EMNLP-IJCNLP 2019 - 2019 Conference on Empirical Methods in Natural Language Processing and 9th International Joint Conference on Natural Language Processing, Proceedings of the Conference*, 3982–3992. https://doi.org/10.48550/arxiv.1908.10084

[39] Shafiq, N., Hamid, I., Asif, M., Nawaz, Q., Aljuaid, H., & Ali, H. (2023). Abstractive text summarization of low-resourced languages using deep learning. *PeerJ Computer Science*, *9*, e1176. https://doi.org/10.7717/PEERJ-CS.1176

[40] Supreet, M., Goel, K., & Gupta, M. (2020). Automatic hindi text summarization using selection and elimination approach. *International Journal of Engineering Applied Sciences and Technology*, *5*(4), 259–266. https://doi.org/10.33564/IJEAST.2020.V05I04.039

[41] Tangsali, R., Pingle, A., Vyawahare, A., Joshi, I., & Joshi, R. (2022). Implementing Deep Learning-Based Approaches for Article Summarization in Indian Languages. *ArXiv Preprint ArXiv:2212.05702*. https://doi.org/10.48550/arXiv.2212.05702

[42] Tawatia, K., Jain, N., & Kundu, S. (2022). Hindi document extractive summarization: neural method on a new data set. *2022 5th International Conference on Computational Intelligence and Networks (CINE)*, 1–6. https://doi.org/10.1109/CINE56307.2022.10037327

[43] *Top 10 Most Spoken FIRST Languages in the World in 2023 | TranslateDay*. (n.d.). Retrieved April 4, 2023, from https://www.translateday.com/most-spoken-languages-in-the-world/

[44] Urlana, A., Bhatt, S. M., Surange, N., & Shrivastava, M. (2023). Indian language summarization using pretrained sequence-to-sequence models. *arXiv preprint arXiv:2303.14461*. https://arxiv.org/abs/2303.14461v1

[45] Wijayanti, R., Khodra, M. L., Surendro, K., & Widyantoro, D. H. (2023). Learning bilingual word embedding for automatic text summarization in low resource language. *Journal of King Saud University - Computer and Information Sciences*, *35*(4), 224–235. https://doi.org/10.1016/J.JKSUCI.2023.03.015

[46] Žagar, A., & Robnik-Šikonja, M. (2022). Cross-lingual transfer of abstractive summarizer to less-resource language. *Journal of Intelligent Information Systems*, *58*(1), 153–173. https://doi.org/10.1007/S10844-021-00663-8/METRICS

[47] Zhang, T., Kishore, V., Wu, F., Weinberger, K. Q., & Artzi, Y. (2019). BERTscore: Evaluating text generation with BERT. *ArXiv Preprint ArXiv:1904.09675*. https://doi.org/10.48550/arXiv.1904.09675