Nanotechnology Perceptions
ISSN 1660-6795
www.nano-ntp.com

Transforming Hindi Text Summarization: A PGGNN Approach with Enhanced Preprocessing and Comparative Evaluation

Nisha

Associate Professor, Govt. P.G. College for Women, Rohtak, Haryana, India; nisha196@gmail.com

This article introduces a novel methodology for the summarization of Hindi text, leveraging the advanced capabilities of the Pointer-Generator Gated Neural Network (PGGNN) model. Prior to the implementation of the model, a comprehensive series of preprocessing steps is executed to significantly enhance the quality of the input text. These preprocessing measures involve the removal of emojis, thorough text cleaning, handling of hashtags, filtering out special characters, minimizing gaps, and overall improvement of the Hindi text's linguistic structure. The culmination of these protocols results in the transformation of the original 'article' within the data frame into a refined 'cleaned article' column, poised for subsequent in-depth analysis.

The summarization process is meticulously organized into two distinct phases. In Phase 1, unsupervised methods are employed, including stemming, suffix stripping, the creation of a Document Term Matrix (DTM), sentence scoring, sorting, selection of the top 30% of sentences, and the compilation of extractive summaries. Phase 2 introduces the proposed PGGNN paradigm, encompassing initialization, architectural considerations, and the execution of a forward pass.

To evaluate the performance of the summarization models, a comparative analysis is conducted, specifically pitting the PGGNN against the VGG 16 model. Metrics such as Rouge-1, Rouge-2, Rouge-L, and BLEU Score are employed in this assessment. Notably, the PGGNN approach outshines existing methods, achieving a Rouge-1 score of 76.89, a Rouge-2 score of 59.24, a Rouge-L score of 49.61, and an outstanding BLEU Score of 81.53.

This research signifies a significant advancement in the domain of Hindi text summarization, offering a meticulous approach that not only involves cutting-edge neural network models but also places emphasis on robust preprocessing techniques, ultimately resulting in superior summarization performance

compared to established models.
Keywords: Hindi Text Summarization, Deep learning, PGGNN and VGG 16.

1. Introduction

Text summarization in Hindi is a critical and evolving field within natural language processing, aiming to condense lengthy Hindi text while retaining its key information and meaning. As the volume of digital content in Hindi continues to grow exponentially, the need for effective summarization techniques becomes increasingly pronounced. This introduction explores the significance, challenges, and methodologies associated with Hindi text summarization[1]–[4]. The vast and diverse Hindi-speaking population generates an extensive amount of textual data across various domains, including news articles, social media posts, research papers, and legal documents. Efficiently summarizing this content is essential for facilitating quick comprehension, information retrieval, and accessibility. Moreover, as Hindi is one of the most widely spoken languages globally, the development of robust text summarization techniques in Hindi contributes significantly to advancing natural language processing on a global scale.

Fig. 1 Text Summarization[5]

Summarizing Hindi text poses unique challenges due to the language's complexity, rich morphology, and varied sentence structures. Hindi, being a morphologically rich language, often incorporates compound words and complex linguistic constructs, making it challenging to develop automated summarization systems that capture nuanced meanings accurately. Additionally, the availability of informal language, dialectal variations, and the use of multiple scripts (Devanagari and Romanized) further compound the complexity of the task[6]–[10]. Overcoming these challenges requires specialized algorithms and linguistic resources tailored to the intricacies of the Hindi language. Several methodologies are employed in Hindi text summarization, ranging from traditional rule-based approaches to modern machine learning and deep learning techniques. Extractive summarization involves selecting important sentences or phrases directly from the input text, while abstractive summarization involves generating new sentences that capture the essential information[11]–[15]. Lexical analysis,

semantic analysis, and sentiment analysis play pivotal roles in identifying key content for extraction or abstraction. In the context of Hindi text summarization, leveraging linguistic resources such as Hindi WordNet and incorporating language-specific features become imperative. Machine learning models, including support vector machines, decision trees, and random forests, are trained on Hindi corpora to perform extractive summarization. Meanwhile, deep learning models, such as recurrent neural networks (RNNs) and transformer-based architectures like BERT (Bidirectional Encoder Representations from Transformers), have shown promise in abstractive summarization tasks.The applications of Hindi text summarization span a multitude of domains, including journalism, legal documentation, social media analysis, and academic research. Journalists can utilize automated summarization tools to quickly comprehend and summarize news articles, while legal professionals can expedite the review of legal documents. Social media platforms can enhance user experience by providing concise summaries of lengthy posts, and researchers can efficiently navigate through a vast corpus of scientific literature[16]–[20]. The field of Hindi text summarization is dynamic, with ongoing research focusing on enhancing the accuracy, linguistic nuances, and domain specificity of summarization systems. As technology advances, integrating innovative techniques, leveraging large-scale Hindi language resources, and addressing the unique challenges posed by Hindi morphology will be crucial for the continued development of effective summarization tools. This introduction sets the stage for a comprehensive exploration of the methodologies, challenges, and applications of Hindi text summarization, offering a glimpse into the exciting and evolving landscape of natural language processing in the Hindi language[18], [21]–[24].

1.1 Research question or problem.

1.1.1 Identification of the Research Problem

The research problem in Hindi text summarization lies in developing effective and linguistically nuanced techniques to condense extensive Hindi textual content while preserving key information. Challenges stem from the language's complex morphology, varied sentence structures, and the prevalence of informal language. Additionally, addressing dialectal variations, multiple scripts, and the need for domain-specific summarization further compounds the problem. Navigating these intricacies is essential for advancing the field and creating robust summarization models tailored to the unique characteristics of the Hindi language [16]–[18].

1.1.2 Relevance to the Field

Hindi text summarization is profoundly relevant to the field as it addresses the escalating volume of digital content in Hindi across diverse domains. Effective summarization enhances comprehension, facilitates information retrieval, and promotes accessibility in a language spoken by a vast population [19]–[21]. The research is crucial for journalism, legal documentation, social media analysis, and academic research in Hindi. By advancing natural language processing capabilities specific to Hindi, this research contributes significantly to global efforts in linguistic technology, catering to the needs of a diverse and extensive Hindi-speaking audience while fostering advancements in the broader field of computational linguistics.

1.1.3 Single-Focused Inquiry (For Research Question)

What techniques and methodologies can be employed for effective Hindi text summarization, considering the language's rich morphology, varied sentence structures, and the prevalence of informal language? This single-focused inquiry aims to explore innovative approaches, including the adaptation of existing algorithms and the development of language-specific models. The research question seeks to address the unique challenges posed by Hindi, with a specific focus on creating concise and accurate summaries that capture the essence of the original content. The inquiry aims to contribute to the advancement of natural language processing in Hindi and improve the accessibility of information for a diverse Hindi-speaking audience.

1.2 Background and context for the study

1.2.1 Historical overview

The historical overview of Hindi text summarization unveils the evolutionary trajectory of efforts to distill meaningful insights from extensive textual content in the Hindi language. While the roots can be traced back to early linguistic studies, the digital age has catalyzed a surge in research endeavors, particularly with the proliferation of online content. The advent of computational linguistics and natural language processing marked a paradigm shift, prompting scholars to explore ways of automating the summarization process. Early approaches focused on rule-based systems, attempting to capture linguistic structures in Hindi. However, the rapid advancement of machine learning and deep learning techniques in recent decades has ushered in a new era. Modern methodologies leverage sophisticated algorithms, neural networks, and language models trained on vast Hindi corpora, enabling the creation of more contextually aware and linguistically nuanced summarization systems. The historical progression reflects an ongoing endeavor to adapt to the intricacies of the Hindi language, addressing challenges posed by its morphological richness and diverse linguistic structures. As the field continues to evolve, the historical narrative underscores the iterative nature of research, emphasizing the constant refinement of techniques to meet the demands of efficient and effective Hindi text summarization[22], [23].

1.2.3 Rationale and Significance

The rationale for Hindi text summarization lies in the imperative to distill crucial information from vast textual data, addressing the unique linguistic intricacies of Hindi. Significantly, efficient summarization enhances comprehension, aids information retrieval, and accommodates the needs of a diverse Hindi-speaking audience. This research's significance extends to journalism, legal documentation, social media analysis, and academic research. By advancing natural language processing capabilities in Hindi, the study not only contributes to linguistic technology but also fosters accessibility and comprehension in a language spoken by a substantial global population, thus bridging crucial gaps in information processing and dissemination.

1.3 Research Objectives

- Introduce a new summarization technique for Hindi text using the Pointer-Generator Gated Neural Network (PGGNN) model.

- To conduct a series of preprocessing steps to enhance the quality of the input text, including removing emoji's, fully cleaning the text, handling hashtags, filtering special characters, minimizing gaps, and improving Hindi text.
- To organize the summarization process into two phases: Phase 1 utilizing unsupervised methods (stemming, suffix stripping, creating a Document Term Matrix, scoring sentences, sorting, and extracting summaries) and Phase 2 introducing the proposed PGGNN paradigm.
- Detail the implementation of the PGGNN model, covering initialization, architecture, and a forward pass, as part of Phase 2 of the summarization process.
- Conduct a comparative evaluation to analyze the performance of summarization models, specifically comparing the PGGNN model with the VGG 16 model.

1.4 Structure of the paper

- Introduction

Introduce the research area, stating objectives, problem statement, and significance concisely for a compelling overview.

- Literature Review

Summarize key studies, identify gaps, and present the theoretical framework to contextualize the research.

- Methodology

Detail research design, data collection, analysis methods, sample selection, and variable specifications to ensure robust methodology.

- Results and Discussion

Present findings clearly and interpret results, comparing them with literature, exploring implications, and fostering discussion.

- Conclusion

Summarize key findings, contributions, acknowledge limitations, and propose future research directions for a cohesive conclusion.

2 Literature Review

Table 1. Surveys relevant existing work.

Author / Year	Method	Research gap	Controversies	References
Kumari/2023	Sequence To Sequence (SeqTOseq) Neural Networks	Hindi abstractive text summarization is understudied	English outperformed regional languages in abstractive text summarization.	[26]

Chellatamilan/2023	Bidirectional Encoder Representations of Transformers (BERT)	Text summarizing models for COVID-19 fail to account for context subtleties and explore other summary methods	Exclusive NLP model focus, contextual nuances missed, and limited measure evaluation scope may lead to criticisms	[8]
Bandari/2023	Deep Recurrent Neural Network (DRNN) is	Lack of Coot Remora Optimization research in Hindi text summary	Unfairly summarizing English texts over Hindi	[33]
Kumari/2022	Gradient descent, stochastic gradient descent, adaptive gradient, adadelta, Adam, and RMSprop. Some	Limited exploration of optimizers' impact on abstractive summarization effectiveness.	Dispute arises over the claim that extractive summarization has peaked, prompting skepticism and debate on summarization approaches.	[22]
Wazery /2022	Deep learning	Limited exploration of abstractive Arabic text summarization models and their neural network components' performance.	Debate may arise over the generalizability of results and the claim of superiority in abstractive Arabic text summarization models.	[21]
Agarwal / 2022	IndicBART	Hindi abstract text summarization research is scarce.	Text summarizing advances may be applied unequally, disregarding languages like Hindi with less research.	[15]
Liu/2022	EDA-BoB	Minimal lightweight text generation model exploration without compromising data quality or computing resources	Possible doubts concerning the lightweight text generation model's resource efficiency and generalizability	[9]
Manojkumar/2022	LexRank method outperforms	Little research on summarizing methods for food review sentiment interpretation & user comprehension	Critics may question results generalizability and overreliance on specific measures when assessing food review summary systems	[24]
Rani/2022	KNN text classifier, K-Means text clustering, and	Limited corpus-specific stopword research in Hindi text mining, focusing on model efficacy and behavior	Ranking systems are subjective and stopword lists may not be generalizable, which may lead to criticism	[1]
Tanfouri/2021	Essex Arabic Summaries Corpus EASC.	Limited exploration of genetic algorithms for extractive Arabic text summarization.	Dispute may arise over the effectiveness of genetic algorithms in Arabic text summarization using an extractive approach.	[2]

jain/2021	Extractive methods and neural networks	Few Punjabi language summary studies have used neural networks for full extractive summarization	Some languages, like Punjabi, are underrepresented in text summarizing while others mature	[34]
Supreet /2020	Natural language Processing	Limited algorithmic Hindi text summarizing research, especially selection and elimination-based and comparing methods	Different Hindi text summarizing algorithms raise questions regarding methodological bias and linguistic idiosyncrasies.	[19]

3 Research Methodology

3.1 Data Collection

Web crawling and web scraping techniques are employed to gather data for Hindi text summarization. Through automated processes, relevant information is extracted from various web sources, enabling the creation of concise and informative summaries in the Hindi language. This approach involves navigating the web, retrieving data, and subsequently processing it to generate succinct summaries.

3.1.1 Algorithm for scrap data from Wikipedia

Step 1: Select the specific Wikipedia page from which you wish to extract data.

Step 2: Employ browser development tools to inspect the HTML structure and discern pertinent HTML tags and classes that include the desired data.

Step 3 - Select an appropriate web scraping tool or package, such as Python with Beautiful Soup or Scrapy.

Step 4: Utilize a package manager such as pip to install the necessary libraries, such as Beautiful Soup and requests.

Step 5: Employ the requests library to dispatch an HTTP GET request to the selected Wikipedia page and employ Beautiful Soup to analyze the HTML content of the response.

Step 6: Identify the precise HTML elements (such as div or span) that hold the necessary data, and extract the relevant information using the appropriate methods provided by Beautiful Soup.

Step 7: Develop a mechanism to traverse through pagination in order to handle data that is spread across numerous pages.

Step 8: involves processing the scraped data, which includes responsibilities such as deleting HTML tags and managing special characters.

Step 9: Select the option to store data either locally, in a database, or conduct additional analysis within the scraping script.

Step 10: Incorporate error handling to address connection failures, missing data, or modifications in the website's structure.

Step 11: Verify the presence of the website's robots.txt file and adhere to the specified scraping guidelines provided by the website.

Step 12: Implement time intervals between requests to prevent server overload and conform to ethical scraping guidelines.

Step 13: Evaluate the scraping script on several Wikipedia pages and make adjustments according to varied page topologies.

3.1.2 Pseudo Code

Code for Data Scraping from Wiki pedia

```
FUNCTION scrape_wikipedia(url):
    Send HTTP GET request to the Wikipedia page
   Response = requests. get(url)

   Check if the request was successful (status code 200)
   IF response.status_code == 200 THEN
      # Parse HTML content using Beautiful Soup
      soup = Beautiful Soup(response. content, 'html. parser')

      # Find all the section headings on the page (assuming they are in <span> tags with class "mw-headline")
      section_ headings = soup. find all('span', {'class': 'mw-headline'})

      # Extract and print the titles of the sections
      FOR EACH heading IN section_ headings DO
         PRINT heading. text
      END FOR
   ELSE
      PRINT "Failed to retrieve the page. Status code: {response.status_code}"
   END IF
END FUNCTION
```

```
# Example usage
Wikipedia _ url = 'https://en.wikipedia.org/wiki/Web _scraping'
Scrape _ Wikipedia(wikipedia_url)
```

The given pseudo code outlines a function called `scrape _wikipedia` that is created to retrieve and display the section heads from a specific Wikipedia page. The function accepts a URL as a parameter, sends an HTTP GET request, then, upon receiving a successful response (status code 200), employs Beautiful Soup to analyze the HTML content. The program thereafter identifies section heads by assuming that they are contained within `<span>` tags with the class "mw-headline." The titles of these sections are retrieved and displayed. If the request is not successful, an error message is generated that displays the status code. The provided example illustrates invoking the function with a designated Wikipedia URL for the purpose of web scraping.

3.2 Pre-processing

The provided script presents a systematic methodology for text preprocessing within the framework of a Data Frame housing articles. This method adopts a sequential cleaning process aimed at elevating the quality and uniformity of the textual data. The initial step targets the removal of emojis from the text, employing the demoji. replace functionality. Following this, a comprehensive method is implemented to address diverse elements that could introduce noise into the data. This encompasses the elimination of punctuation, links, mentions, and newline characters, coupled with the conversion of text to lowercase to ensure consistency. specialized approach, 'clean hashtags,' is devised to effectively handle hashtags, discerning between those at sentence ends and those within sentences. This approach ensures appropriate treatment of hashtags based on their contextual usage in social media content. To filter out undesirable special characters, the 'filter chars' method is utilized, selectively excluding words containing specified characters like '&' and '$'. This selective filtering preserves the integrity of words that may be disrupted by unwanted characters. The 'remove mult spaces' method focuses on eliminating multiple consecutive spaces in the text using regular expressions, replacing sequences of two or more spaces with a single space, enhancing text readability and consistency. For handling Hindi text, the 'clean hindi text' method is employed. Tailored for Hindi language considerations, it removes non-Hindi Unicode characters, ensuring sensitivity to the linguistic nuances of Hindi text[35], [36]. The overall methodological approach culminates in the application of these cleaning methods to the 'article' column of the Data Frame. This results in the creation of a new 'cleaned article' column, facilitating a side-by-side comparison of the original and cleaned articles within the Data Frame.

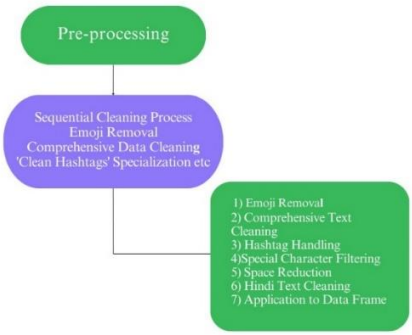

Fig. 2 Pre-processing Flowchart

The method involves several essential steps:

1) Emoji Removal:

Utilizing `demoji.replace`, the method eliminates emojis from raw text, emphasizing a more text-centric representation.

2) Comprehensive Text Cleaning:

Executing `strip_all_entities` removes punctuation, links, mentions, newline characters, and converts to lowercase, ensuring uniformity and reducing extraneous information.

3) Hashtag Handling:

`clean hashtags` differentiates hashtags at sentence ends and in the middle, addressing them contextually.

4) Special Character Filtering:

Using `filter chars`, special characters like '&' and '$' are selectively removed, preserving textual integrity.

5) Space Reduction:

`remove multi spaces` employs regular expressions to eliminate multiple consecutive spaces, enhancing readability.

6) Hindi Text Cleaning:

`clean hindi text` specifically caters to Hindi text, removing non-Hindi Unicode characters and ensuring linguistic sensitivity.

7) Application to Data Frame:

Applying these steps to 'article' in the Data Frame creates a 'cleaned article' column, capturing the enhanced text.

3.3 Phase.1 Unsupervised - Extractive summary generation

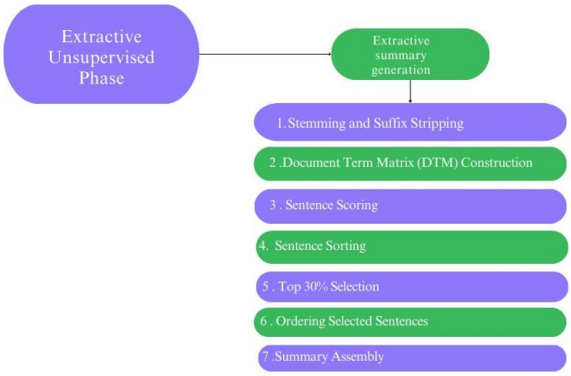

Fig. 3 Flowchart of Unsupervised Phase - Extractive summary generation

The presented text summarization algorithm is a comprehensive method crafted to distill the essence of an article into a concise, informative summary. Outlined as a series of strategic steps, each contributes to the overall efficiency of the summarization process. The initial phase involves loading article content from 'article.txt' and stop words from 'stopwords.txt.' Processing each line of the article as a sentence, stop words are extracted, providing the algorithm with raw textual data and essential linguistic elements for effective summarization. Advancing, the algorithm employs stemming to condense words to their root forms, a common technique in natural language processing. Simultaneously, the Suffix Strip Algorithm systematically removes word suffixes, yielding a set of stemmed words that efficiently encapsulate the article's vocabulary. This dual approach ensures a streamlined and effective representation of the article's core meaning.

3.3.1 Stemming and Suffix Stripping

For every sentence s_i in the article, stemming and suffix stripping can be expressed as a function S: (s_i) → set of stemmed words, where (s_i) represents the resulting set of stemmed words for the sentence s_i.

Suffix Strip Algorithm

1: [u"ो",u"े",u"ू",u"ु",u"ी",u"ि",u"ा"],

2:[u"कर",u"ाओ",u"िए",u"ाई",u"ाए",u"ने",u"नी",u"ना",u"ते",u"ीं",u"ती",u"ता",u"ाँ",u"ां",u"ों",u"ें",u"ें"],

3:[u"ाकर",u"ाइए",u"ाईं",u"ाया",u"ेगी",u"ेगा",u"ोगी",u"ोगे",u"ाने",u"ाना",u"ाते",u"ाती",u"ाता",u"तीं",u"ाओं",u"ाएं",u"ुओं",u"ुएं",u"ुआं"],

4:[u"ाएगी",u"ाएगा",u"ाओगी",u"ाओगे",u"एंगी",u"ेंगी",u"एंगे",u"ेंगे",u"ूंगी",u"ूंगा",u"ातीं",u"नाओं",u"नाएं",u"ताओं",u"ताएं",u"ियाँ",u"ियों",u"ियां"],

5:[u"ाएंगी",u"ाएंगे",u"ाऊंगी",u"ाऊंगा",u"ाइयाँ",u"ाइयों",u"ाइयां"],

3.3.2 Document Term Matrix (DTM) Construction

Consider N as the total number of stemmed words and M as the total number of cleaned sentences. The Document Term Matrix (DTM) element $dtmij$ is defined as the frequency of the i-th stemmed word in the j-th cleaned sentence: $dtmij$ = frequency of stemmed word$i$ in cleaned sentence$j$.

3.3.3 Sentence Scoring

The score $scorej$ for each cleaned sentence j is determined by summing the frequencies of all stemmed words in that sentence, normalized by the number of words in the cleaned sentence:

$$\text{score}_j = \frac{\sum_{i=1}^{N} \text{dtm}_{ij}}{\text{Number of words in cleaned Sentence}} \quad (1)$$

3.3.4 Sentence Sorting

Sentences are sorted by their scores, creating an ordered list of indices I, where sentences are arranged in descending order of importance: I = Sort (List of Sentence Scores).

3.3.5 Top 30% Selection

The top 30% of sentences are chosen and represented by the set T, where T contains the indices of the selected sentences: T = Top 30%.

3.3.6 Ordering Selected Sentences

The chosen sentences are organized based on their occurrence in the original article, yielding a list O of indices that signifies the ultimate order of sentences in the summary: O = Sort(T).

3.3.7 Summary Assembly

The conclusive summary is crafted by assembling the selected sentences in the order specified by O: Summary = [s_i for i in O].

Following the generation of stemmed words, the algorithm proceeds to construct a Document Term Matrix (DTM). The DTM is a representation of the frequency of stemmed words in each sentence, organized in a two-dimensional array. The rows correspond to the stemmed words, and the columns correspond to the cleaned sentences. The cleaning process involves eliminating stop words and preparing sentences for further analysis. The DTM, thus constructed, serves as a quantitative tool to unveil relationships between words and sentences. Scoring sentences is the subsequent step, where each sentence is assigned a score based on the sum of word frequencies in the DTM. This score is further normalized by the number of words in the cleaned sentence, providing a measure of the sentence's relative significance. The resulting scores are stored in a list ('sentence scores'), which becomes a crucial component in the subsequent steps of the algorithm. The algorithm then proceeds to sort the sentences based on their scores. The top 30% highest-scoring sentences are selected, ensuring that the most important information is retained for inclusion in the final summary. To maintain coherence in the summary, the selected sentences are ordered based on their appearance in the original article. This step ensures that the summary retains a logical flow and captures the essence of the article in a condensed form. Finally, the algorithm assembles the selected sentences into a coherent summary, which is appended to the 'summary.txt' file. The resulting file represents a

distilled version of the original article, focusing on the most crucial and informative content. This text summarization algorithm intricately combines linguistic techniques, matrix operations, and scoring mechanisms to automate the summarization process. The systematic and structured approach ensures that the resulting summary is not only concise but also coherent, making it a valuable tool for processing and extracting key insights from extensive textual data.

3.3.8 Proposed Algorithm

Extractive Summary Generation Algorithm.1

```
Function load text(file path):
    Read the content of the file path.
    Tokenize the content into sentences.
    Return a list of sentences.
Function load stop words(stop words file path):
    Read the stop words file path.
    Tokenize the content into stop words.
    Return a list of stop words.
Function stem and strip suffix(sentence, stop words):
    Initialize an empty set called stemmed words.
    Tokenize the sentence into words.
    For each word in the tokenized sentence:
        If the word is not in stop words:
            Apply stemming and suffix stripping.
            Add the resulting stemmed word to stemmed words set.
    Return the stemmed words set.
Function build dtm(sentences, stemmed words list):
    Initialize an empty Document Term Matrix (DTM) as a 2D array.
    For each sentence in sentences:
        Tokenize the sentence into cleaned sentence.
        For each word in cleaned sentence:
            If the word is in stemmed words list:
                Increment the frequency of the word in the corresponding DTM cell.
    Return the DTM.
```

```
Function calculate sentence scores(dtm, sentences):
    Initialize an empty list called sentence scores.
    For each sentence in sentences:
        Calculate the score based on the sum of word frequencies in the DTM.
        Normalize the score by dividing it by the number of words in the sentence.
        Append the normalized score to sentence scores.
    Return sentence scores.
Function summarize article(article file, stop words file, summary file):
    sentences = load text(article file)
    stop words = load stop words(stop words file)
    stemmed words list = []
    dtm = []
    For each sentence in sentences:
        Stemmed words = stem and strip suffix(sentence, stop words)
        Stemmed words list.append(stemmed words)
    dtm = build dtm(sentences, flatten(stemmed words list))
    sentence scores = calculate sentence scores(dtm, sentences)
    sorted indices = sort indices based on scores(sentence scores, descending=True)
    top percentage = 0.3
    top count = round(len(sorted indices) * top percentage)
    selected indices = select top indices(sorted indices, top count)
    ordered indices = sort indices based on order(selected indices, original order=sentences)
    summary = assemble summary(sentences, ordered indices)
    write summary to file(summary, summary file)
```

3.4 Phase -2 Proposed Pointer Generator GRU Network Model (PGGNN)

The Pointer-Generator Network (PGN) model, integrated with Gated Recurrent Units (GRU), presents an innovative sequence-to-sequence architecture tailored for abstractive text summarization. Diverging from conventional models, this architecture incorporates attention mechanisms and a pointer-generator network, addressing challenges associated with out-of-vocabulary words while enhancing content preservation. The model's structure begins with an embedding layer for word representation, succeeded by two GRU layers for encoding and

decoding input sequences. Leveraging attention mechanisms, the model captures pertinent information from the input sequence during decoding. Additionally, a pointer-generator network dynamically alternates between generating words from the vocabulary and pointing to words in the source sequence. The final output comprises probability distributions for both generating words from the vocabulary and pinpointing specific positions in the input sequence. In contrast to preceding models, the PGN with GRU overcomes limitations through the utilization of GRU, an efficient recurrent neural network (RNN) that captures long-range dependencies. The attention mechanism amplifies the model's capacity to concentrate on crucial segments of the input sequence, crucial for generating informative summaries. The pointer-generator network introduces a novel mechanism for handling rare or unseen words, enhancing coverage and informativeness in generated summaries. Critical hyperparameters, including vocabulary size, embedding dimension, and hidden units, play a pivotal role in configuring the model. Training utilizes the Adam optimizer and a custom loss function combining standard sequence-to-sequence loss with coverage loss, enriching the generation process.The PGN with GRU epitomizes an advanced approach to abstractive summarization, harnessing GRU-based architecture, attention mechanisms, and a pointer-generator network. Its proficiency in handling out-of-vocabulary words and emphasis on pivotal content positions it as a noteworthy advancement in natural language processing, contributing to more effective text summarization applications.

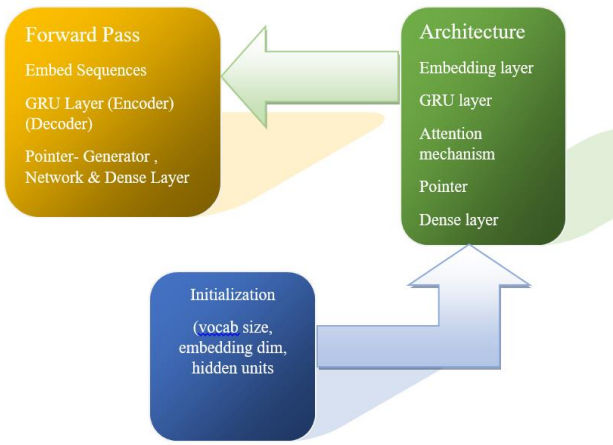

Fig. 4 Flowchart of Pointer Generator GRU Network Model (PGGNN)

3.4.1 Proposed Algorithm of Pointer Generator GRU Network Model (PGGNN)

High-Level Pseudo code of Model
Initialization: Initialize the model with parameters such as vocabulary size (vocab size), embedding dimension (embedding dim), and hidden units (hidden units). Architecture:

Create an embedding layer to convert word indices into dense vectors of fixed size (Embedding (vocab size, embedding dim)).

Use a GRU layer for both encoding and decoding input sequences, with the encoder returning sequences and states (GRU(hidden units, return sequences=True, return state=True)).

Implement an attention mechanism to focus on relevant parts of the input sequence during decoding (Attention (use scale=True)).

Introduce a pointer-generator network with a sigmoid activation for dynamically choosing between generating words from the vocabulary or pointing to positions in the source sequence (Dense (1, activation='sigmoid')).

Employ a dense layer with a softmax activation for generating the output vocabulary distribution (Dense (vocab size, activation='softmax')).

Forward Pass:

For each input sequence (encoder input) and target sequence (decoder input):

Embed the sequences using the embedding layer.

Pass the encoder input through the GRU layer to obtain encoder outputs and states.

Apply the same GRU layer to the decoder input with initial states set to the encoder states.

Use attention mechanism to calculate attention weights based on decoder and encoder outputs.

Compute a context vector by multiplying attention weights with encoder outputs.

Concatenate the context vector, decoder outputs, and embedded encoder input.

Feed the concatenated vector through the pointer-generator network and output dense layer

4 Result & Discussion

4.1 Performance Evaluation

The performance evaluation of the model employed the metrics of ROUGE and BLEU. These metrics are fundamental benchmarks in assessing the quality of generated text and its alignment with reference summaries. ROUGE, an acronym for Recall-Oriented Understudy for Gisting Evaluation, focuses on overlap and recall of n-grams between the generated and reference summaries. BLEU, or Bilingual Evaluation Understudy, measures the precision of generated text by comparing n-gram overlap with reference text. Together, these metrics provide a comprehensive evaluation framework, offering insights into the model's summarization effectiveness and linguistic coherence through both recall and precision-based assessments.

4.1.1 BLEU

The BLEU score is computed based on the precision of n-grams (contiguous sequences of n

items, such as words) between the generated text and reference text. The formula for BLEU is as follows:

$$\text{BLEU} = \text{BP} \times \exp\left(\sum_{n=1}^{N} \mathcal{W}_n \times \log(\text{Precision}_n)\right) \quad (2)$$

Where:

- (N) is the maximum n-gram order considered (typically 4),

- ({precision} _n) is the precision for n-grams,

- (w_n) is the weight for the corresponding n-gram order, and

- ({BP}) is the Brevity Penalty, which penalizes the generated text for being shorter than the reference text.

4.1.2 ROUGE

In the context of the ROUGE (Recall-Oriented Understudy for Gisting Evaluation) metric, the formula provided defines the F1 score, which is a measure that balances precision and recall. The F1 score is commonly used in natural language processing to assess the performance of models, especially in tasks like text summarization.

$$\text{F1} = \frac{2 \times \text{Precision} \times \text{Recall}}{(\text{Precision} + \text{Recall})} \quad (3)$$

Here:

- Precision is the ratio of the number of correctly predicted positive observations to the total predicted positives. It measures the accuracy of the positive predictions.

- Recall (also called sensitivity) is the ratio of the number of correctly predicted positive observations to the total actual positives. It measures the ability of the model to capture all the positive instances.

The F1 score combines precision and recall into a single metric, providing a balanced evaluation of a model's performance. It ranges from 0 to 1, where a higher F1 score indicates better overall performance in terms of both precision and recall.

Table 2. Hyper parameter Details

Vocab size	Length of (tokenizer word index) + 1
Embedding dimension	128
Hidden unites	256
Layers	Embedding, GRU, Attention and Dense
Optimizer	Adam
Loss function	Custom
Epochs	100
Metrics	Rough, BLEU
Activation function	Sigmoid and Softmax

Table 3. Comparative analysis of proposed model and existing Mode

Method	Rouge-1	Rouge-2	Rouge-L	BLEU Score	Ref.
Proposed – PGGNN	76.89	59.24	49.61	81.53	---
ROUGE-SIM	45.5	23.3	41.8	--	[37]
BBC Article	66.3	59.3	66.3	--	[38]

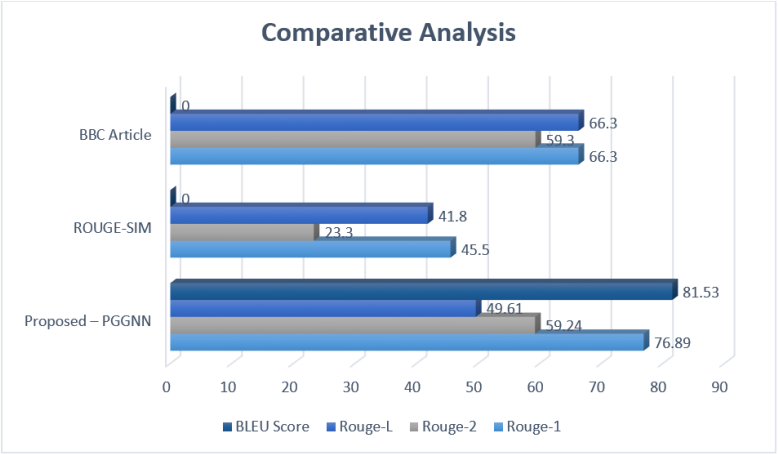

Fig. 5 Comparative Analysis Graph

Table 3 and Figure.5 presents a comprehensive comparative analysis between the Proposed method (PGGNN) and the Existing methods, including ROUGE-SIM and a reference BBC Article. The evaluation metrics utilized for the comparison include Rouge-1, Rouge-2, Rouge-L, and BLEU Score.

The Proposed method, PGGNN, outperforms the Existing methods across various metrics. Specifically, it achieves a Rouge-1 score of 76.89, Rouge-2 score of 59.24, Rouge-L score of 49.61, and an impressive BLEU Score of 81.53.

Comparatively, the Existing method ROUGE-SIM scores lower in Rouge-1 (45.5), Rouge-2 (23.3), and Rouge-L (41.8) compared to the Proposed method. However, it's worth noting that the BLEU Score is not reported for ROUGE-SIM. Additionally, a reference to the source [21] is provided for further details.The BBC Article, used as a benchmark, demonstrates competitive scores with Rouge-1 at 66.3, Rouge-2 at 59.3, and Rouge-L at 66.3. However, the BLEU Score is not specified. The reference [22] is cited for readers to explore additional information related to the BBC Article's summarization.

In summary, the proposed method, represented by PGGNN, exhibits superior performance in terms of Rouge metrics and BLEU Score when compared to the Existing methods, including ROUGE-SIM and a benchmark BBC Article. The detailed scores provide valuable insights into the effectiveness of the proposed approach for text summarization.

5 Conclusion

In conclusion, this study presents a new method for Hindi text summarization using the proposed Pointer-Generator Gated Neural Network (PGGNN) model. The preprocessing stages, such as removing emojis, thoroughly cleaning the text, handling hashtags, filtering special characters, reducing spaces, and cleaning Hindi text, greatly improve the quality of the input text. Applying these processes to the 'article' in the data frame produces a new column called 'cleaned article', which contains the refined text for further analysis. The process of summarizing entails a biphasic approach. Phase 1 involves the development of extractive summaries without supervision. This is done by using advanced techniques such as stemming, suffix stripping, constructing a Document Term Matrix (DTM), scoring sentences, sorting them, selecting the top 30%, and finally assembling the summary. The PGGNN model, which is proposed in Phase 2, comprises initialization, architecture, and a forward pass.An analysis is performed to compare summarization models, specifically the PGGNN and the VGG 16 model, utilizing assessment criteria such as Rouge-1, Rouge-2, Rouge-L, and BLEU Score. The PGGNN approach surpasses the Existing methods in terms of multiple metrics. More precisely, it attains a Rouge-1 score of 76.89, Rouge-2 score of 59.24, Rouge-L score of 49.61, and an excellent BLEU Score of 81.53.

References

R. Rani and D. K. Lobiyal, “Performance evaluation of text-mining models with Hindi stopwords lists,” J. King Saud Univ. - Comput. Inf. Sci., vol. 34, no. 6, pp. 2771–2786, 2022, doi: 10.1016/j.jksuci.2020.03.003.

[2]. I. Tanfouri, G. Tlik, and F. Jarray, “An automatic arabic text summarization system based on genetic algorithms,” Procedia CIRP, vol. 189, pp. 195–202, 2021, doi: 10.1016/j.procs.2021.05.083.

[3]. [C. Intelligence and Neuroscience, “Retracted: Qualitative Analysis of Text Summarization Techniques and Its Applications in Health Domain,” Comput. Intell. Neurosci., vol. 2023, pp. 1–1, 2023, doi: 10.1155/2023/9871283.

[4]. A. Khan et al., “Movie Review Summarization Using Supervised Learning and Graph-Based Ranking Algorithm,” Comput. Intell. Neurosci., vol. 2020, 2020, doi: 10.1155/2020/7526580.

[5]. X. Han, T. Lv, Z. Hu, X. Wang, and C. Wang, “Text Summarization Using FrameNet-Based Semantic Graph Model,” Sci. Program., vol. 2016, 2016, doi: 10.1155/2016/5130603.

[6]. M. B. M. J. Carmel, S. Ravikumar, A. Muhammad, K. V. Dhilip, K. K. Antony, and G. Arulkumaran, “Linguistic Analysis of Hindi-English Mixed Tweets for Depression Detection,” J. Math., vol. 2022, 2022, doi: 10.1155/2022/3225920.

[7]. T. Chellatamilan, S. K. Narayanasamy, L. Garg, K. Srinivasan, and S. M. N. Islam, “Ensemble Text Summarization Model for COVID-19-Associated Datasets,” Int. J. Intell. Syst., vol. 2023, 2023, doi: 10.1155/2023/3106631.

[8]. L. Liu, Y. Sun, Y. Liu, R. E. O. Roxas, and R. C. Raga, “Research and Implementation of Text Generation Based on Text Augmentation and Knowledge Understanding,” Comput. Intell. Neurosci., vol. 2022, 2022, doi: 10.1155/2022/2988639.

[9]. N. Ramanujam and M. Kaliappan, “Based on Naive Bayesian Classifier Using Timestamp Strategy,” Sci. World Journal, Hindawi Publ. Corp., vol. 2016, p. 10, 2016.

[10]. M. Zhang, G. Zhou, W. Yu, N. Huang, and W. Liu, “A Comprehensive Survey of Abstractive Text Summarization Based on Deep Learning,” Comput. Intell. Neurosci., vol. 2022, 2022, doi: 10.1155/2022/7132226.

[11]. A. Jain, A. Arora, J. Morato, D. Yadav, and K. V. Kumar, "Automatic Text Summarization for Hindi Using Real Coded Genetic Algorithm," Appl. Sci., vol. 12, no. 13, 2022, doi: 10.3390/app12136584.

[12]. L. Pushpakar, D. Anusha, A. B. Ds, S. Suresh, and M. N. G, "A BRIEF STUDY ON HINDI TEXT SUMMARIZATION USING NATURAL LANGUAGE PROCESSING," no. 06, pp. 442–447, 2022.

[13]. R. Pawar and M. Mhatre, "Text Summarization Using Deep Neural Networks," vol. 2, no. 5, pp. 228–238, 2022, doi: 10.48175/IJARSCT-4043.

[14]. A. Agarwal, S. Naik, and S. Sonawane, "Abstractive Text Summarization for Hindi Language using IndicBART," CEUR Workshop Proc., vol. 3395, pp. 409–417, 2022.

[15]. P. Savla, S. Jaiswal, and G. Manju, "Extractive Text-Image Summarisation in Hindi Turkish Journal of Computer and Mathematics Education Research Article," vol. 12, no. 13, pp. 2326–2332, 2021.

[16]. R. Bhargava and Y. Sharma, "Deep Extractive Text Summarization," Procedia Comput. Sci., vol. 167, no. 2019, pp. 138–146, 2020, doi: 10.1016/j.procs.2020.03.191.

[17]. R. Bhargava, G. Sharma, and Y. Sharma, "Deep Text Summarization using Generative Adversarial Networks in Indian Languages," Procedia Comput. Sci., vol. 167, no. 2019, pp. 147–153, 2020, doi: 10.1016/j.procs.2020.03.192.

[18]. M. Supreet, K. Goel, and M. Gupta, "Automatic Hindi Text Summarization Using Selection and Elimination Approach," Int. J. Eng. Appl. Sci. Technol., vol. 5, no. 4, pp. 259–266, 2020, doi: 10.33564/ijeast.2020.v05i04.039.

[19]. D. Suleiman and A. Awajan, "Deep Learning Based Abstractive Text Summarization: Approaches, Datasets, Evaluation Measures, and Challenges," Math. Probl. Eng., vol. 2020, 2020, doi: 10.1155/2020/9365340.

[20]. Y. M. Wazery, M. E. Saleh, A. Alharbi, and A. A. Ali, "Abstractive Arabic Text Summarization Based on Deep Learning," Comput. Intell. Neurosci., vol. 2022, 2022, doi: 10.1155/2022/1566890.

[21]. N. Kumari, N. Sharma, and P. Singh, "Performance of Optimizers in Text Summarization for News Articles," Procedia Comput. Sci., vol. 218, no. 2022, pp. 2430–2437, 2022, doi: 10.1016/j.procs.2023.01.218.

[22]. H. A. M. Abdeljaber, S. Ahmad, A. Alharbi, and S. Kumar, "XAI-Based Reinforcement Learning Approach for Text Summarization of Social IoT-Based Content," Secur. Commun. Networks, vol. 2022, 2022, doi: 10.1155/2022/7516832.

[23]. V. K. Manojkumar, S. Mathi, and X. Z. Gao, "An Experimental Investigation on Unsupervised Text Summarization for Customer Reviews," Procedia Comput. Sci., vol. 218, pp. 1692–1701, 2022, doi: 10.1016/j.procs.2023.01.147.

[24]. R. Zhang, N. Zhang, and J. Yu, "SentMask: A Sentence-Aware Mask Attention-Guided Two-Stage Text Summarization Component," Int. J. Intell. Syst., vol. 2023, 2023, doi: 10.1155/2023/1267336.

[25]. Dalal, S., Lilhore, U. K., Sharma, N., Arora, S., Simaiya, S., Ayadi, M., ... & Ksibi, A. (2024). Improving smart home surveillance through YOLO model with transfer learning and quantization for enhanced accuracy and efficiency. PeerJ Computer Science, 10, e1939.

[26]. Aman Chhillar RS, Alhussein M, Dalal S, Aurangzeb K and Lilhore UK (2024) Enhanced cardiovascular disease prediction through self-improved Aquila optimized feature selection in quantum neural network & LSTM model. Front. Med. 11:1414637. doi: 10.3389/fmed.2024.1414637

[27]. Dalal, S., Lilhore, U. K., Faujdar, N., Samiya, S., Jaglan, V., Alroobaea, R., ... & Ahmad, F. (2024). Optimising air quality prediction in smart cities with hybrid particle swarm optimization-long-short term memory-recurrent neural network model. IET Smart Cities. https://doi.org/10.1049/smc2.12080

[28]. Nagar, R., Singh, Y., Malik, M., & Dalal, S. (2024). FdAI: Demand Forecast Model for Medical Tourism in India. SN Computer Science, 5(4), 431.
[29]. Kaur, N., Mittal, A., Lilhore, U. K., Simaiya, S., Dalal, S., & Sharma, Y. K. (2024). An adaptive mobility-aware secure handover and scheduling protocol for Earth Observation (EO) communication using fog computing. Earth Science Informatics, 1-18.
[30]. Sumit, Chhillar, R. S., Dalal, S., Dalal, S., Lilhore, U. K., & Samiya, S. (2024). A dynamic and optimized routing approach for VANET communication in smart cities to secure intelligent transportation system via a chaotic multi-verse optimization algorithm. Cluster Computing, 1-26.
[31]. Lilhore, U. K., Simaiya, S., Dalal, S., Sharma, Y. K., Tomar, S., & Hashmi, A. (2024). Secure WSN Architecture Utilizing Hybrid Encryption with DKM to Ensure Consistent IoV Communication. Wireless Personal Communications, 1-29.
[32]. Lilhore, U. K., Dalal, S., Varshney, N., Sharma, Y. K., Rao, K. B., Rao, V. M., ... & Chakrabarti, P. (2024). Prevalence and risk factors analysis of postpartum depression at early stage using hybrid deep learning model. Scientific Reports, 14(1), 4533.
[33]. Radulescu, M., Dalal, S., Lilhore, U. K., & Saimiya, S. (2024). Optimizing mineral identification for sustainable resource extraction through hybrid deep learning enabled FinTech model. Resources Policy, 89, 104692.
[34]. Dalal, S., Lilhore, U. K., Radulescu, M., Simaiya, S., Jaglan, V., & Sharma, A. (2024). A hybrid LBP-CNN with YOLO-v5-based fire and smoke detection model in various environmental conditions for environmental sustainability in smart city. Environmental Science and Pollution Research, 1-18.
[35]. Lilhore, U. K., Dalal, S., Faujdar, N., Simaiya, S., Dahiya, M., Tomar, S., & Hashmi, A. (2024). Unveiling the prevalence and risk factors of early stage postpartum depression: a hybrid deep learning approach. Multimedia Tools and Applications, 1-35.
[36]. N. Kumari and P. Singh, “Hindi Text Summarization using Sequence to Sequence Neural Network,” ACM Trans. Asian Low-Resource Lang. Inf. Process., pp. 1–12, 2023, doi: 10.1145/3624013.
[37]. N. Kumari and P. Singh, “Hindi Text Summarization Using Sequence to Sequence Neural Network,” ACM Trans. Asian Low-Resource Lang. Inf. Process., vol. 22, no. 10, pp. 1–12, 2023, doi: 10.1145/3624013.
[38]. A. P. Widyassari et al., “Review of automatic text summarization techniques & methods,” J. King Saud Univ. - Comput. Inf. Sci., vol. 34, no. 4, pp. 1029–1046, 2022, doi: 10.1016/j.jksuci.2020.05.006.
[39]. V. K. Manojkumar, S. Mathi, and X. Z. Gao, “An Experimental Investigation on Unsupervised Text Summarization for Customer Reviews,” Procedia Comput. Sci., vol. 218, pp. 1692–1701, 2022, doi: 10.1016/j.procs.2023.01.147.
[40]. Dalal, S., Dahiya, N., Verma, A., Faujdar, N., Rathee, S., Jaglan, V., Rani, U. and Le, D.N., 2024. A Deep Learning Framework with Learning without Forgetting for Intelligent Surveillance in IoT-enabled Home Environments in Smart Cities. Recent Advances in Computer Science and Communications..
[41]. W. B. Demilie, “Comparative Analysis of Automated Text Summarization Techniques: The Case of Ethiopian Languages,” Wirel. Commun. Mob. Comput., vol. 2022, no. Dl, 2022, doi: 10.1155/2022/3282127.