SN Computer Science (2023) 4:614
https://doi.org/10.1007/s42979-023-01983-y

REVIEW ARTICLE

A Comprehensive Analysis of Indian Legal Documents Summarization Techniques

Saloni Sharma[1] · Surabhi Srivastava[2] · Pradeepika Verma[3] · Anshul Verma[2] · Sachchida Nand Chaurasia[2]

Received: 17 March 2023 / Accepted: 30 May 2023

Abstract

In the Legal AI field, the summarization of legal documents is very challenging. Since the Indian case documents are much noisier and poorly organized, the summarization of legal documents can be useful for legal professionals, who often have to read and analyze large amounts of legal text. During the review process of the legal documents, a team of reviewers may be needed to understand and for taking further actions. A branch of text summarization called 'legal text summarization' which is concerned with summarizing legal texts, such as court opinions, contracts, and legal briefs may reduce the need of these reviewers. Legal text summarization aims to highlight the key points of a legal document and convey them in a concise form so that decisions can be made in quick manner. In this paper, we experimented on seven machine learning-based summarization models to analyse their performance on judgment report datasets that has been collected from Indian national legal portal. The models that are taken here for the analysis are BART, LexRank, TextRank, Luhn, LSA, Legal Pegasus, and Longformer. We experimented with these models to find which model may perform well on the legal data. As a result, we observed that Legal Pegasus outperforms over all other models in the case legal summarization.

Keywords Legal document summarization · BART · Legal Pegasus · Longformer · LexRank · TextRank · Luhn · LSA

Introduction

Legal text summarization is a subfield of Natural Language Processing where the aim is to make a system that can automatically generate the summary of a large complex legal dataset. Unlike the most commonly used datasets, like news articles, Wikipedia pages, or social media content, the structure of the legal text is much richer [25]. Very long and complex sentences with several abbreviations, named entities, and citations are common in legal documents. Legal organizations generally prepares headnotes with the help of lawyers to understand large set of legal notes. However, such a process is labor-intensive, time-consuming, and very expensive. To manage and process the enormous volume of legal papers produced internationally by different legal authorities in less time, automatic summarise of these documents can be quite useful to lawyers. With the use of such an approach, many cases that are currently pending in Indian courts can be accelerated.

Text summarization is the process of reducing a text document with a computer program to create a summary that retains the most important points of the original document [18, 38–40]. There are two types of summarization approaches: one is extractive summarization and another is abstractive summarization. In legal text document summarization, most of the time, extractive summarization is used. In this approach, the most important sentences will be extracted and concatenated to form a summary [13, 32, 42]. It ranks all

This article is part of the topical collection "Research Trends in Computational Intelligence" guest edited by Anshul Verma, Pradeepika Verma, Vivek Kumar Singh and S. Karthikeyan.

✉ Pradeepika Verma
pradeepikav.verma093@gmail.com

Saloni Sharma
engineersaloni159@gmail.com

Surabhi Srivastava
surabhi.2000.sss@gmail.com

Anshul Verma
anshul.verma@bhu.ac.in

Sachchida Nand Chaurasia
snchaurasia@bhu.ac.in

[1] Jawaharlal Nehru University, New Delhi, India

[2] Banaras Hindu University, Varanasi, India

[3] TIH, Indian Institute of Technology, Patna, India

Published online: 11 August 2023

the sentences according to the relevance of the text and then chooses the most important text. It picks up the sentences directly from the document (based on the scoring function). Another approach is abstractive summarization which generally generates new text, rather than selecting the existing text, to provide a summary of the original text. There are both supervised and unsupervised approaches to generate abstractive summaries. It is more complex than extractive summarization and also computationally expensive.

In today's fast-paced world, it is imperative for legal professionals to quickly digest large amounts of information contained in legal documents. Legal single document summarization is a crucial tool that helps lawyers and judges to quickly understand the essence of legal cases and make informed decisions. With the advancement in Natural Language Processing (NLP) techniques, automated summarization has become a reality.

In this comparative analysis, we have examined seven state-of-the-art models for legal single document summarization, known as BART (Bidirectional Auto-Regressive Transformers), LexRank, TextRank, Luhn, LSA, Legal Pegasus, and Longformer. BART [21] is a pre-trained transformer-based language model that has been fine-tuned for various NLP tasks and has achieved outstanding results in the field of summarization. Next, Legal Pegasus [6, 15] is a recently proposed language model that is specifically designed for legal document summarization and has shown promising results. LexRank [7] is a graph-based algorithm that calculates the importance of a sentence based on its similarity to other sentences in the text. It uses PageRank [27] to determine the sentence's scores. Sentences with higher scores are considered more important and selected for the summary. TextRank [22] is another graph-based algorithm that uses a similar approach to LexRank, but instead of using sentence similarity, it uses word similarity to create a graph and calculate sentence's importance. The algorithm is based on the PageRank algorithm and determines the sentence score based on the sum of its weighted inbound edges.

Luhn [16, 45] is a heuristic-based algorithm that uses sentence length and word frequency to determine the importance of sentences. It assigns higher scores to those sentences that contain more keywords and longer sentences containing more information. LSA (Latent Semantic Analysis) [24] is a statistical algorithm that identifies underlying topics in a text and creates a summary based on the most important topics. LSA creates a vector space model for the text where each word is represented as a vector, and then performs a singular value decomposition to identify the most important topics. Longformer [46] is a deep learning-based algorithm that uses a transformer architecture to perform summarization. Longformer can process long documents and create summaries that maintain the key information in the original text. It uses a hierarchical attention mechanism to identify important words and phrases in the text and create a summary.

In this study, we aim to evaluate the performance of these seven models in terms of their ability to generate concise and accurate summaries of 30 legal documents. The results of this comparative analysis provide valuable insights into the strengths and limitations of each model and help to identify areas for improvement. The legal documents used in this research were sourced from the publicly available website, i.e., Indian Kanoon,[1] which is a free-access repository of judgments, orders, and other legal documents from the Indian judiciary. The documents have been uploaded to a GitHub repository[2] for ease of access and reproducibility.

Motivation

The motivation for summarizing legal documents stems from the increasing volume of legal information being produced and the need for lawyers, legal researchers, and other professionals to quickly and accurately understand the essential information contained in these documents. With the increasing number of court cases and legal proceedings, the amount of legal information available is overwhelming, making it difficult for individuals to effectively review and comprehend all the relevant information. Additionally, the Indian case records are poorly organized and noisy, which makes them challenging for both humans and machine learning algorithms to understand and process.

Text summarization offers a solution to this problem by condensing large legal documents into a shorter, more concise format that retains the essential information and meaning of the original document. This enables individuals to quickly understand the key elements of a case, including the parties involved, the legal issue being addressed, the main arguments and evidence presented, and the outcome of the case. This saves time and effort and helps ensure that the most important information is not missed.

Moreover, the accuracy of legal information is critical, as incorrect or misleading information can have serious consequences for legal proceedings. Text summarization techniques can be used to accurately and objectively summarize legal documents, reducing the risk of errors and improving the overall quality of legal information.

Despite the potential benefits of legal text summarization, existing methods have limitations, including low accuracy, lack of relevance, and limited scalability. There is a need for new and improved methods to overcome these limitations

[1] www.indiankanoon.org.

[2] https://github.com/Saloni-sharma29/Summarizing-Indian-Legal-Documents-A-Comparative-Study-of-Models-and-Techniques.

and provide more accurate and relevant legal text summaries. This presents an opportunity for further research and development in this field and highlights the importance of continuing to advance our understanding of legal text summarization.

Organization

The rest of this paper is organized as follows. We first introduce some state-of-the-art work in the field of Legal text summarization. Then, basic methodology of summarization is introduced. Thereafter, different evaluation metrics have been presented, followed by the experimental results and discussions. Finally, the paper is concluded with future scope.

Background

Legal document summarization is a growing field that has received increased attention in recent years due to the large volume of legal documents being produced [12]. Legal case judgments are usually lengthy and complicated due to the use of various domain-specific abbreviations. A lot of research work has been done till now to summarize the legal documents related to various countries. The structure of the legal documents may vary for different countries. Most of the research works tried to train deep learning models using a supervised or semi-supervised approach for legal text summarization. In this literature review, we have explored various approaches which have been used for summarizing legal documents. A brief discussion to these approaches are as follows.

One of the common approaches to summarization is rule-based methods [1, 33], where a set of predefined rules are used to extract the most important information from legal documents. This approach is widely used but has limitations in handling complex and ambiguous language. Another approach that has been widely used in this domain is machine learning algorithms [8, 34–37, 41], such as decision trees and support vector machines, and Deep learning techniques (recurrent neural networks and transformers). These algorithms are effective in summarizing legal texts, as they can learn from large datasets of legal documents and generate summaries that can accurately reflect the most important information.

Farzindar and Lapalme [9] proposed an approach for legal document summarization based on the exploration of the documents architecture and thematic structures. In this work, the authors selected three sentences from each topic that are having best representation of the topic. The proposed summarization method blends term description with sentence description for each topic using LSA (Latent Semantic Analysis). [28] introduced a model for multi-document summarization using graph-based and information-theoretic concepts.

One of the first works in text summarization for Indian judgments was proposed by [29] where the concept of Conditional Random Field (CRF) is used. The authors have segmented the document into rhetorical roles. Also, various features had been used for the identification of labels. They presented their results on 200 Kerala High Court Judgements of which 50 were hand-annotated.

Bhattacharya et al. [3] presented a systematic comparative study where the summaries of 17000+ Indian supreme court judgments were analysed using algorithms such as LexRank, LSA, DSDR, LetSum, Casesummarizer, and Graphical Model Neural Extractive Summarizer. Basic preprocessing procedures for legal documents have not been automatically operated such as phrase and word tokenization or entity recognition. They suggested that Sentence tokenization is more challenging in the legal terminology which is extremely technical, different from conventional text, and full of abbreviations.

Bhattacharya et al. [4] also proposed an extractive summarization approach DELSumm using unsupervised learning algorithm which is designed to systematically incorporate guidelines from legal experts into an optimization setup. The proposed algorithm outperformed several supervised summarization models that are trained over thousands of document-summary pairs.

Cao et al. [5] proposed an approach in which Convolution Neural Networks and Long Short Term Memory (LSTM) were used for creating sentence representations. They proposed the concept of summary before defining how much a sentence is appropriate to be selected into a summary without consideration of its context. The work of [25] aimed to bridge that gap by providing a collection of pre-processed and annotated documents which can be used for several downstream tasks. This was the largest publicly available annotated corpora of Indian legal text. They created a new dataset consisting of over 10,000 judgments delivered by the supreme court of India and their corresponding handwritten summaries (Also known as headnotes). This dataset was pre-processed by normalizing common legal abbreviations, handling spelling variations in named entities, handling bad punctuation, and accurate sentence tokenization. They used the two-layer Bi-LSTM neural network model.

Polsley et al. [26] presented a tool for automated summarization for legal documents named as 'CaseSummarizer' which uses quality summary methods based on word frequency augmented with subsidiary domain-specific knowledge. Summaries are then produced by an informative

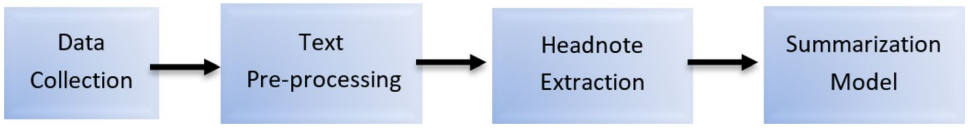

Fig. 1 Data flow diagram of methodology

Table 1 A sample of raw text and normalized text

Raw Text	Normalized Text
"CIVIL APPEAL NO(s). 3189 OF 2022 (arising out of SLP (Civil) No(s). 4125 of 2019) Appellants - eight in number, have assailed the correctness of Judgment and Order dated 12.09.2018 passed by the Division Bench of the High Court of Kerala at Ernakulam in W.A. No. 2108 of 2016 between Sulthan Bathery Municipality vs. Kalyani and 12 others"	"civil appeal numbers 3189 of 2022 (arising out of special leave petitions (civil) numbers 4125 of 2019) appellants - eight in number, have assailed the correctness of judgment andorder dated 12.09.2018 passed by the division bench of the high court of kerala at ernakulam in writ appeal number2108 of 2016 between sulthan bathery municipality verses kalyani and 12 others"

interface with abbreviations, significance heat maps, and other flexible controls.

[12] proposed an approach for normalizing legal texts in the Indian context to improve the performance of the domain-independent model. They used two domain-independent models for legal text summarization, namely BART for extractive text summarization and PEGASUS for abstractive text summarization. They also evaluated the result using ROUGE.

The work of [30] carried out extensive experiments with several extractive and abstractive summarization methods (both supervised and unsupervised) over three legal summarization datasets that they had created. Their analysis includes evaluation by law practitioners leading to several interesting insights on legal summarization in specific and long document summarization. In this paper, they mentioned a Legal Pegasus model[3] which is fine-tuned for the legal domain, and trained to perform abstractive summarization tasks.

Methodology of Document Summarization

A legal judgment report consists of various abbreviations and legal terms, thus it needs pre-processing. During the pre-processing phase, all these terms have been brought to their original normalized form. In this work, we adopted a common methodology used for the summarization as shown in Fig. 1 [11]. All the legal documents are passed through these stages in order to reach their final result. These modules include data collection, text pre-processing, headnote extraction, and summarization model.

Data Collection

We have collected the dataset of Indian judgment reports from publically available websites like Supreme Court Of India (SCI)[4] and IndianKanoon.[5] In this work, we have taken 30 legal documents for experimental and analysis purpose. Since there is no publically available dataset for the legal document in the structured format, so we collected it manually which has been uploaded here.[6]

Text Pre-Processing

The majority of the pre-processing procedures involve normalizing abbreviations, Sentence recognition, stemming, lemmatization, POS tagging, recognizing incorrect punctuation added by typing errors, and eliminating excessive spaces in phrases, which are then utilized to enhance sentence tokenization. The data is extracted from the PDF documents using a python library 'pdfplumber' which is used to extract text and layout information from PDF documents. So, the collected data is converted into text format where pre-processing has been done. We constructed a dictionary where we stored the Legal abbreviation and its full form. After the data extraction, the abbreviations have been removed using the dictionary. We have stored approximately 75 legal and 117 common English abbreviations in the dictionary that are commonly used in legal judgment documents along with their full form. Table 1 shows an example of raw text whose abbreviations are converted into their normalized form with the help of the dictionary. All the other processes have been done using NLTK toolkit.

[3] https://huggingface.co/nsi319/legal-pegasus.

[4] https://main.sci.gov.in/.

[5] https://indiankanoon.org/.

[6] https://github.com/Saloni-sharma29/Summarizing-Indian-Legal-Documents-A-Comparative-Study-of-Models-and-Techniques.

Headnote Extraction

The headnote is extracted from the document to evaluate the summaries generated by different models. For extracting the headnote, [19] uses regular expressions to extract a section of text from the input text that is enclosed between the strings 'headnote:' and 'judgment:'. The format of the legal document is in such a way that the headnote section is presented just before the Judgment section which is very helpful for a regular expression approach to find the pattern.

Summarization Models

After text pre-processing, the whole text of the document has been sent as input to various summarization models in order to generate the summary of the document. Here, we have considered seven summarization models for the experimental and analysis purpose which are discussed below in brief.

BART

BART [21, 31] is a sequence-to-sequence model that can be used for various NLP tasks including summarization. It is a pre-trained transformer-based model that uses a combination of self-attention and cross-attention mechanisms to learn representations of the input text. The BART summarization algorithm generates the system summary by conditioning the model on the input text and predicting a summary sequence. During training, the model learns to predict the target summary from the source text by minimizing the cross-entropy loss between the predicted and target summary sequences. The training data for BART summarization typically consists of pairs of input–output text sequences. The input sequences are usually documents or articles, and the output sequences are their corresponding summaries.

The process of generating a summary involves predicting the most likely token at each time step, given the previous tokens in the summary. The prediction is made by sampling from the conditional probability distribution of the next token, given the previous tokens and the context vector. The BART algorithm also employs beam search to generate multiple candidate summaries and select the best one based on a scoring function. The scoring function typically takes into account the fluency and relevance of the summary. The working of the BART algorithm can be expressed mathematically as follows:

Let x be the input document, and y be the summary. The goal of the BART summarization algorithm is to find the summary y that maximizes the conditional probability $P(y|x)$ which can be computed as follows in Eq. 1.

$$P(y|x) = \prod_{i=1}^{N} P(y_i|y_{1:i-1}, x), \tag{1}$$

where N is the length of the summary sequence, and $P(y_i|y_{1:i-1}, x)$ is the conditional probability of the i^{th} summary token given the previous tokens and the input document.

The BART algorithm uses the auto-regressive objective to estimate the conditional probability distribution of the summary tokens. The objective is to minimize the negative log-likelihood of the target summary. The likelihood (L) of the summary can be computed as follows in Eq. 2.

$$L = -\sum_{i=1}^{N} \log P(y_i|y_{1:i-1}, x) \tag{2}$$

LexRank

LexRank is an unsupervised algorithm for extractive summarization that identifies important sentences in a document based on their similarity to other sentences [7]. The algorithm uses the concept of eigenvector centrality from graph theory to score the sentences and identify the most important ones. The algorithm of the LexRank [7] is as follows.

1. First, the text is pre-processed to remove stop words, punctuation, and other irrelevant text. The remaining words are stemmed or lemmatized to reduce the number of unique words and to group together words with the same stem or lemma.
2. Next, a similarity matrix is constructed, where each entry (i, j) represents the similarity between sentence i and sentence j. There are many ways to compute similarity, but one common method is to use cosine similarity, which is a measure of the angle between two vectors. The vectors are created by counting the number of times each word appears in a sentence and then normalizing the vector by its length.
3. The similarity matrix is then converted into a graph, where each sentence is a node and the edges represent the similarity between sentences.
4. The eigenvector centrality of each node in the graph is then computed using Eq. 3, which is a measure of the importance of a node based on the importance of its neighbors. This is done by computing the principal eigenvector of the adjacency matrix of the graph, which gives a score to each node based on the scores of its neighbors.

$$s(i) = (1-d) + d \sum_{j=1}^{n} \frac{w(i,j)}{\sum_{k=1}^{n} w(j,k) \cdot s(j)}, \quad (3)$$

where, $s(i)$ is the eigenvector centrality score of sentence i, n is the total number of sentences, $w(i, j)$ is the weight of the edge between sentence i and sentence j in the similarity graph, d is a damping factor between 0 and 1.

5. The sentences are then ranked by their eigenvector centrality scores, and the top-ranked sentences are selected as the summary. The number of sentences to include in the summary is a hyper-parameter here that must be set by the user.

Text Rank

TextRank [22] is another unsupervised graph-based algorithm for text summarization, which is similar to LexRank in its approach. TextRank algorithm also represents the text as a graph, where the sentences are nodes and edges are the similarity between the sentences. The similarity between sentences is calculated using cosine similarity between their vector representations, which are obtained through pre-trained word embeddings. TextRank algorithm also uses the PageRank algorithm, originally used in the Google search engine, to calculate the importance score of each sentence based on the graph structure. The importance of each sentence is calculated as follows in Eq. 4.

$$Score(s(i)) = (1-d) + d \sum_{j=1}^{n} \frac{w(j,i)}{\sum_{k=1}^{n} w(j,k)} Score(s(j)), \quad (4)$$

where, $Score(s(i))$ represents the score of sentence i, d is a damping factor, which is usually set to 0.85, $w(j, i)$ represents the weight of the edge between sentence i and sentence j, $w(j, k)$ is the weights of edges originating from sentence j.

TextRank algorithm then selects the top-ranked sentences to form the summary. The number of sentences to be selected can be set by the user as a hyper-parameter.

Note The key difference between TextRank and LexRank is that TextRank uses cosine similarity to calculate the edge weights between sentences, while LexRank uses BM25 similarity. Additionally, TextRank uses the sum of all scores of the connected sentences, while LexRank uses the maximum score of the connected sentences to calculate the score of each sentence.

Luhn

Luhn [16, 45] is one of the oldest algorithms proposed for automated text summarization. The algorithm is based on the observation that important words in a text appear more frequently than others. It works by assigning a score to each sentence in the text based on the frequency of occurrence of significant words in that sentence [10]. The algorithm consists of the following steps:

1. Tokenize the text into words and remove stop words and punctuation.
2. Count the frequency of each word in the text.
3. Assign a score to each sentence by summing the frequency of occurrence of significant words in that sentence. Significant words are those that occur more frequently than a predefined threshold (usually the median frequency).
4. Rank the sentences in descending order of their scores.
5. Select the top-ranked sentences that together make up a summary of the text.

Note that the Luhn algorithm is a simple unsupervised method that does not require training data. However, it has several limitations, such as the inability to capture the semantic meaning of words, and the tendency to select sentences that contain many significant words, even if they are not coherent.

Latent Semantic Analsis

LSA is a mathematical technique used for information retrieval and summarization [24, 43]. It uses Singular Value Decomposition (SVD) to identify the underlying patterns and relationships between the terms and documents in a corpus. The basic steps involved in LSA-based summarization are as follows:

1. The input text is pre-processed by removing stop words, and punctuation marks, and stemming or lemmatizing the words to normalize them.
2. A term-document matrix is then created where the rows represent the terms and the columns represent the documents. Each cell in the matrix contains the frequency of the term in the corresponding document.
3. Thereafter, SVD is performed on the term-document matrix, using Eq. 5, to reduce its dimensions and to extract the most important features [14].

$$X = USV^{\top}, \quad (5)$$

where X is the term-document matrix, U is an orthogonal matrix that contains the left singular vectors, S is a diagonal matrix that contains the singular values, and V^T is an orthogonal matrix that contains the right singular

vectors. The SVD results are used to identify the underlying concepts or topics in the corpus.

4. Each sentence in the document is scored based on cosine similarity function using Eq. 6 with the underlying concepts.

$$cosine\_similarity(A, B) = \frac{A \cdot B}{|A| \cdot |B|}, \tag{6}$$

where $(A.B)$ is the dot product of the vectors A and B, and $|A|$ and $|B|$ are the Euclidean lengths of the vectors A and B, respectively.

5. The top-scoring sentences are selected to form the summary, where the score of i^{th} sentences in the j^{th} document is calculated as follows.

$$score(i, j) = cosine\_similarity(A_i, B_j), \tag{7}$$

where A_i is a vector representing the ith sentence and B_j is a vector representing the jth document. The vectors A_i and B_j are obtained by multiplying the SVD results with the original term-document matrix.

Legal Pegasus

Legal Pegasus [15] is the algorithm developed by OpenAI for summarizing legal documents. It builds upon the Pegasus algorithm which is a pre-training-based approach for text summarization. Legal Pegasus is specifically designed to address the challenges in summarizing legal documents such as the technical language, domain-specific terminology, and complex legal reasoning. The algorithm is based on the transformer architecture and uses an encoder-decoder framework. The encoder takes in the input document and generates a series of hidden representations. The decoder then generates the summary by attending to these hidden representations and generating the output sequence.

The model is optimized using the cross-entropy loss function given in Eq. 7 which measures the difference between the predicted and actual summary.

$$L = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{m_i} y_{ij}\log(p_{ij}), \tag{8}$$

where N is the number of training examples, m_i is the length of the ith summary, y_{ij} is the one-hot encoding of the jth token in the ith summary, and p_{ij} is the predicted probability of the jth token in the ith summary. The loss function is used to update the parameters of the model during training. The loss function is minimized using the Adam optimizer. The quality of the summary is evaluated using the ROUGE metric, which measures the overlap between the predicted and actual summary.

Longformer

Longformer [2] is a transformer-based model that uses a sliding window approach to handle long documents. The model can handle up to 4,096 tokens per document by dividing the document into overlapping chunks and using an attention mechanism to connect the different chunks. The Longformer algorithm can be expressed mathematically as follows ([2]):

1. At first, the input document is represented as the token sequence $X = [x_1, x_2, ..., x_N]$. The document is divided into overlapping chunks of size k with a stride of s, such that the mth chunk is represented by $X_m = [x_{(m-1)s+1}, ..., x_{(m-1)s+k}]$.
2. Each chunk X_m is then encoded using a transformer encoder to produce a sequence of hidden states $H_m = [h_{m,1}, h_{m,2}, ..., h_{m,k}]$, where $h_{m,i}$ is the hidden state of the ith token in the mth chunk.
3. The summary of the document is computed by taking a weighted average of the embeddings of all tokens in the document, where the weights are determined by the attention function using the given Eq. 9.

$$\text{Attention}(Q, K, V) = \text{Softmax}(\frac{QK^T}{\sqrt{d}})V \tag{9}$$

where Q, K, and V are the query, key, and value embeddings, respectively, and d is the dimensionality of the embeddings.

4. Specifically, let the set of all hidden states be represented by, $H = h_{m,i}|m = 1, ..., M, i = 1, ..., k$. Then the summary vector S is computed as:

$$S = \sum_{m=1}^{M}\sum_{i=1}^{k}\text{Attention}(Q, K, V)h_{m,i} \tag{10}$$

The summary vector S is then passed through a linear layer to produce the final summary.

Overall, Longformer provides an efficient and effective approach for summarizing long documents by leveraging the power of transformer-based models and attention mechanisms to handle the complex interdependence between tokens in the document.

Evaluation

For evaluations of the system-generated summaries, we have used ROUGE metric. ROUGE stands for Recall-Oriented Understudy for Gisting Evaluation. Here the term 'Gisting' means to extract the main point from the text [23].

ROUGE is an evaluation matrix that compares the generated summary to a reference summary to calculate the overlap between the two in terms of n-gram, word sequences, and word pairs. The higher ROUGE score indicates a better match between the summary and the reference summary. It consists of five different variants like ROUGE-N, ROUGE-S, ROUGE-L, ROUGE-W, and ROUGE-SU [17]. In this work, we have used ROUGE-N and ROUGE-L metrics for the analysis of various summarization models where *N* refers to the length of the n-gram, which can be ROUGE-1 (unigram), ROUGE-2 (bigram), ROUGE-3 (trigram), and so on. We can defined the ROUGE-N or ROUGE-L Metrics in terms of precision, recall, and F1 Scores parameters as follows.

Precision refers to the proportion of the summary that is relevant to the original document. It is calculated as the number of relevant sentences in the summary divided by the total number of sentences in the summary as shown in the Eq. 11.

$$Precision_{ROUGE-L} = \frac{\text{Common n-grams in generated summary and reference summary}}{\text{Number of n-grams in generated summary}} \tag{11}$$

Recall is a metric used to measure the effectiveness of a summarization algorithm and represents the proportion of important information in the original text that is present in the summary produced by the algorithm. The equation of recall is shown in Eq. 12.

$$Recall_{ROUGE-L} = \frac{\text{Common n-grams in generated summary and reference summary}}{\text{Number of n-grams in reference summary}} \tag{12}$$

F1 score is the harmonic mean of precision and recall, and it combines both precision and recalls into a single score that balances both measures as shown in Eq. 13.

$$F1score_{ROUGE-L} = 2 * \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}, \tag{13}$$

Where n is the length of the n-gram considered (e.g., $n = 1$, 2, 3). These formulas can be used to evaluate the quality of a summary generated by an algorithm by comparing it to a reference summary.

Result and Discussion

In this paper, we have experimented with seven different models that are **BART** [31], **LexRank** [7], **TextRank** [22], **Luhn** [45], **LSA** [43], **Legal Pegasus** [15], and **Longformer** [2] to see their performance on 30 legal judgment data. In our evaluation, we have shown the evaluation results using Precision, Recall, ROUGE-1, ROUGE-2, and ROUGE-L metrics. Here, we have considered both precision and recall metrics as equally important to see the overall performance of the model. Thus, we choose the model with the highest F1 score. A higher F1 score indicates that the model has better overall performance.

Table 2 shows the precision scores of different summarization methods (BART, Legal, Longformer, LSA, Lexrank, Text rank, and Luhn) applied to 30 different documents. From these scores we have observed that the Legal pegasus method outperformes with the precision score of 0.348933333, followed by BART summarization with 0.2833, and then Longformer with 0.164166667.

Table 3 shows the Recall score of all the summarizer models for each documents (labeled 1–30). Based on the table, it appears that the Longformer algorithm has the highest recall scores on average, followed closely by Lexrank and Legal, while LSA and Text rank have the lowest recall scores on average.

Table 4 shows the F1 scores of seven summarization models on each document. The average F1 Score for each model across all documents is as follows: BART (0.181391), Legal Pegasus (0.31889), Long Former (0.1324), LSA (0.149307), Lex Rank (0.166067), Text Rank (0.222533), and Luhn (0.1733).

There is some correlation between the two types of metrics, they are not directly comparable. Precision and recall scores are based on exact matches of individual words, while ROUGE scores take into account the similarity of n-grams. Therefore, it is possible for a model to perform well on one type of metric and poorly on the other.

In table Tables 5 and 6, we can see that the legal Pegasus model has the highest ROUGE scores in Table 6, but has lower precision and recall scores compared to other models in Table 5. This suggests that the legal Pegasus model is better at capturing the overall meaning of the reference summary, but may generate more errors in individual words compared to other models. The following Fig. 2 shows the precision and recall score percentage of all seven models.

In Fig. 3, it can be concluded that the Legal Pegasus model outperformed all other models on all three ROUGE scores, indicating that it is a highly effective tool for summarizing legal documents. This may be due to the fact

Table 2 Precision score over all models

S. no.	Precision						
	BART	Legal	Longformer	LSA	Lexrank	Text rank	Luhn
1	0.268	**0.529**	0.215	0.261	0.318	0.197	0.189
2	0.125	**0.265**	0.042	0.212	0.133	0.214	0.181
3	0.124	**0.335**	0.111	0.264	0.115	0.335	0.222
4	0.111	0.116	0.065	0.114	**0.2**	**0.2**	0.162
5	0.098	0.074	0.012	0.08	0.012	**0.129**	0.076
6	0.045	**0.094**	0.01	0.058	0.056	0.092	0.098
7	0.128	0.13	0.039	**0.151**	0.015	**0.151**	0.175
8	**0.663**	0.76	0.557	0.339	0.148	0.406	0.406
9	0.317	0.462	0.249	0.624	0.652	**0.755**	0.366
10	**0.693**	0.666	0.35	0.511	0.333	0.476	0.407
11	0.417	**0.432**	0.181	0.371	0.226	0.371	0.371
12	0.411	0.484	0.214	**0.615**	**0.615**	0.545	0.484
13	0.12	0.224	0.103	0.185	**0.245**	**0.245**	0.2
14	0.483	0.516	0.223	0.337	0.159	**0.587**	0.587
15	0.43	0.43	0.15	0.343	0.388	**0.72**	0.43
16	0.437	**0.475**	0.127	0.293	0.355	0.381	0.387
17	0.123	0.115	0.07	0.118	0.013	**0.391**	0.141
18	0.113	0.179	0.046	0.159	0.075	0.106	**0.244**
19	0.085	0.085	0.051	0.058	0.006	**0.151**	**0.151**
20	0.07	0.182	0.031	0.095	0.015	**0.199**	0.122
21	0.096	0.119	0.052	0.096	0.097	**0.12**	0.119
22	0.244	0.253	0.073	0.189	**0.264**	0.229	0.171
23	**0.56**	0.486	0.208	0.334	0.162	0.495	0.294
24	0.091	0.348	0.218	0.201	0.3588	**0.397**	0.239
25	0.372	**0.532**	0.36	0.366	0.451	0.451	0.44
26	0.133	**0.245**	0.086	0.119	0.083	0.145	0.133
27	0.464	0.509	0.404	0.404	0.163	**0.648**	0.369
28	0.032	0.147	0.032	0.118	0.124	**0.239**	0.23
29	0.529	0.474	0.204	0.371	0.388	**0.697**	0.438
30	0.717	0.802	0.442	0.821	0.795	**0.882**	0.702
Average	0.2833	0.348933	0.164167	0.273567	0.23216	**0.365133**	0.284467

that Legal Pegasus was specifically trained on legal texts, allowing it to identify and extract key legal concepts more effectively.

The F1 score using ROUGE-2 is consistently lower than the F1 scores using ROUGE-1 and ROUGE-L across all models. This suggests that capturing two-word sequences is more challenging than capturing single words or longer sequences of words. It is important to note that while ROUGE scores are a useful metric for evaluating summarization models, they are not the only factor to consider. Other factors, such as the ability to handle multiple languages, processing speed, and the ability to handle different types of inputs, may also be important depending on the use case.

Manual Testing

We have also done manual testing of the generated summaries for different summarization techniques in terms of their readability where two expert assessors give a rating for each system-generated summary on three-level "Likert" scale format that are readable, partial readable and non-readable. They were given following queries to answer for readability [44].

1. Whether summaries are Non-redundant, and focused to main topic.
2. Whether sentences of summaries are complete.
3. Whether Summaries do not contain complex sentences.

Table 3 Recall score over all models

S. No.	Recall						
	BART	Legal	Longformer	LSA	Lexrank	Text rank	Luhn
1	1.731	1.47	1.784	1.738	1.681	1.802	**1.81**
2	1.874	1.734	**1.957**	1.787	1.866	1.785	1.818
3	1.875	1.664	**1.888**	1.735	1.884	1.664	1.777
4	1.888	1.883	**1.934**	1.885	1.799	1.799	1.837
5	1.901	1.925	1.987	1.919	**1.987**	1.87	1.923
6	1.954	1.905	**1.989**	1.941	1.943	1.907	1.901
7	1.871	1.869	1.96	1.848	**1.984**	1.848	1.824
8	1.336	1.239	1.442	1.66	**1.851**	1.593	1.593
9	1.682	1.537	**1.749**	1.375	1.347	1.244	1.633
10	1.306	1.333	1.649	1.488	**1.666**	1.523	1.592
11	1.582	1.567	**1.818**	1.628	1.773	1.628	1.628
12	1.588	1.515	**1.785**	1.384	1.384	1.454	1.515
13	1.879	1.775	**1.896**	1.814	1.754	1.754	1.799
14	1.516	1.483	1.776	1.662	**1.839**	1.412	1.412
15	1.569	1.569	**1.849**	1.656	1.611	1.279	1.569
16	1.562	1.524	**1.872**	1.706	1.644	1.618	1.612
17	1.876	1.884	1.929	1.88	**1.986**	1.608	1.858
18	1.886	1.82	**1.953**	1.84	1.924	1.893	1.755
19	1.914	1.914	1.948	1.941	**1.993**	1.848	1.848
20	1.929	1.817	1.968	1.904	**1.984**	1.8	1.877
21	1.903	1.88	**1.947**	1.903	1.902	1.879	1.88
22	1.755	1.746	**1.926**	1.81	1.735	1.77	1.828
23	1.439	1.513	1.791	1.665	**1.837**	1.504	1.705
24	**1.908**	1.651	1.781	1.798	1.641	1.602	1.76
25	1.627	1.467	**1.639**	1.633	1.548	1.548	1.559
26	1.866	1.754	0.133	1.88	**1.916**	1.854	1.866
27	1.535	1.49	1.595	1.595	**1.836**	1.351	1.63
28	**1.967**	1.852	**1.967**	1.881	1.875	1.76	1.769
29	1.47	1.525	**1.795**	1.628	1.611	1.302	1.561
30	1.282	1.197	**1.557**	1.178	1.204	1.117	1.297
Average	1.7157	1.650067	**1.775467**	1.7254	1.766833	1.633867	1.714533

They were instructed that if all the answers for the given questions are 'yes' then it is rated as readable. If atleast one answer comes as'yes', then it is rated as partial readable else it is rated as non-readable. Human judgments cannot be consistent every time. Therefore, it is interesting to measure how well do two different judges agree on readability. The best way to measure for inter-judge agreement is the kappa statistics [20]. It is defined as follows:

$$Kappa(\kappa) = \frac{P(A) - P(E)}{1 - P(E)} \tag{14}$$

where $P(A)$ is the proportion of the times the judges agreed, and $P(E)$ is the proportion of the times the judges agree by chance. The value of κ-measure in the interval [2/3, 1] are seen as acceptable. The results of manual readability with kappa measures are shown in Table 7.

Conclusion and Future Scope

In conclusion, the comparative analysis of all seven models has demonstrated the potential of legal single document summarization in improving the efficiency and accuracy of legal document analysis. The results of the comparative analysis have shown that the Legal Pegasus model outperforms the other models in terms of generating concise and accurate summaries of legal documents. The compared summarization methods are quit scalable for large documnets. This highlights the potential of domain-specific language models in NLP tasks and the importance of fine-tuning models for specific domains. Moving forward, there is a lot of scope for further improvement and advancement in the field of legal document summarization. Some potential areas of research include:

Table 4 Performance of all models over the legal documents

S. No.	F-1 Score						
	BART	Legal Pegasus	Long Former	LSA	Lex Rank	Text Rank	Luhn
1	0.121	**0.566**	0.211	0.15	0.31	0.094	0.177
2	0.411	**0.58**	0.031	0.108	0.088	0.127	0.173
3	0.112	**0.542**	0.041	0.191	0.041	0.327	0.091
4	**0.476**	0.439	0.027	0.081	0.197	0.197	0.106
5	0.124	**0.576**	0.011	0.06	0.011	0.121	0.032
6	**0.402**	0.367	0.007	0.027	0.049	0.074	0.091
7	0.231	**0.626**	0.026	0.057	0.012	0.09	0.122
8	**0.627**	0.494	0.515	0.133	0.094	0.127	0.127
9	0.041	0.094	0.223	0.21	**0.632**	0.184	0.091
10	0.09	0.371	0.309	0.1302	0.187	0.161	**0.388**
11	0.042	0.11	0.159	0.26	0.163	**0.267**	**0.267**
12	0.086	0.129	0.107	**0.602**	**0.602**	0.363	0.182
13	0.279	**0.506**	0.064	0.072	0.238	0.238	0.19
14	0.057	0.082	0.18	0.2	0.097	**0.3**	**0.3**
15	0.01	0.062	0.068	0.171	0.131	**0.251**	0.156
16	0.09	0.104	0.112	0.229	0.209	**0.354**	**0.354**
17	**0.484**	0.358	0.034	0.032	0.007	0.388	0.085
18	0.231	**0.429**	0.002	0.072	0.065	0.081	0.138
19	0.068	**0.506**	0.023	0.035	0.004	0.114	0.114
20	0.03	**0.429**	0.031	0.083	0.01	0.124	0.12
21	0.02	0.101	0.052	0.066	0.081	**0.117**	0.113
22	0.088	**0.217**	0.018	0.179	0.145	0.118	0.073
23	0.038	0.063	0.179	0.107	0.118	**0.483**	0.089
24	0.163	0.125	0.209	0.124	**0.249**	0.217	0.117
25	0.096	0.144	0.338	0.351	0.312	0.312	**0.426**
26	0.076	**0.317**	0.07	0.113	0.06	0.086	0.127
27	0.197	0.342	0.39	0.106	0.081	**0.636**	0.144
28	**0.408**	0.33	0.026	0.083	0.095	0.172	0.177
29	0.063	0.33	0.11	0.199	**0.374**	0.205	0.356
30	0.242	0.212	**0.399**	0.248	0.32	0.348	0.273
Average	0.181391	0.31889	0.1324	**1.49307**	0.166067	0.222533	0.1733

Table 5 Average Precision and Recall Scores

Models	Precision	Recall
BART	0.2833	1.7157
Legal Pegasus	0.348933	1.650067
Longformer	0.164167	**1.775467**
LSA	0.273567	1.7254
Lex Rank	0.23216	1.766833
Text Rank	**0.365133**	1.633867
Luhn	0.284467	1.714533

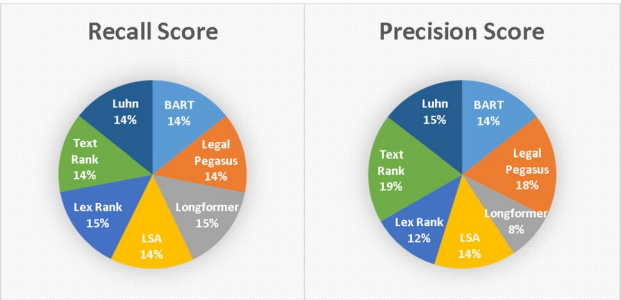

Fig. 2 Precision and recall score of models

Table 6 Average ROUGE scores of all models

Average Score	Summarizers						
	BART	Legal Pegasus	Long Former	LSA	Lex Rank	Text Rank	Lunh
ROUGE-1 F1 Score	0.18402	**0.333787**	0.133367	0.163067	0.175867	0.246	0.1854
ROUGE-2 F1 Score	0.114287	**0.238572**	0.0786	0.071033	0.103333	0.138933	0.096833
ROUGE-L F1 Score	0.181391	**0.31889**	0.1324	0.149307	0.166067	0.222533	0.1733

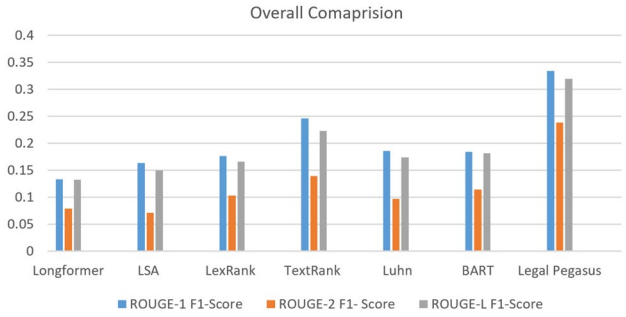

Fig. 3 Performance of all models based on ROUGE F-1 Score

Table 7 Manual readability judgements for generated summaries by summarizers

Summarizers	Judgements	κ-measure
BART	Readable	0.729
Legal Pegasus	Readable	0.921
Longformer	Partial-Readable	0.954
LSA	Partial-Readable	0.775
Lex Rank	Readable	0.786
Text Rank	Readable	0.822
Luhn	Partial-Readable	0.818

1. Incorporating legal domain knowledge into the language models to enhance their performance.
2. Evaluating the models on a diverse set of legal documents, including contracts, judgments, and legislation, to assess their generalizability.
3. Develop techniques to handle complex legal concepts, such as legal jargon and technical terms, in the summarization process.
4. Integrating the summarization models with legal document management systems to streamline the legal document analysis process.

This study provides a foundation for future research in this field and serves as a reference for legal professionals looking to adopt automated summarization tools.

Funding This research work is supported by "Council of Science & Technology, U.P. (Project ID: 1982)" and "Seed Grant to Faculty Members under IoE Scheme (under Dev. Scheme No. 6031)".

Data availability The dataset, which has been used in this paper, is available at https://github.com/Saloni-sharma29/Summarizing-Indian-Legal-Documents-A-Comparative-Study-of-Models-and-Techniques.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

Ethical approval This article does not contain any studies with human participants or animals performed by any of the authors.

References

1. Andhale, N. and Bewoor, L. A. (2016). An overview of text summarization techniques. In 2016 international conference on computing communication control and automation (ICCUBEA), pages 1–7. IEEE.
2. Beltagy, I., Peters, M. E., and Cohan, A. (2020). Longformer: The long-document transformer. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings, pages 4916–4925. Association for Computational Linguistics.
3. Bhattacharya, P., Hiware, K., Rajgaria, S., Pochhi, N., Ghosh, K., and Ghosh, S. (2019). A comparative study of summarization algorithms applied to legal case judgments. In Advances in Information Retrieval: 41st European Conference on IR Research, ECIR 2019, Cologne, Germany, April 14–18, 2019, Proceedings, Part I 41, pages 413–428. Springer.
4. Bhattacharya, P., Poddar, S., Rudra, K., Ghosh, K., and Ghosh, S. (2021). Incorporating domain knowledge for extractive summarization of legal case documents. In Proceedings of the eighteenth international conference on artificial intelligence and law, pages 22–31.
5. Cao, Z., Wei, F., Li, S., Li, W., Zhou, M., and Wang, H. (2015). Learning summary prior representation for extractive summarization. In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers), pages 829–833.
6. Champlin, E. (1978). Pegasus. Zeitschrift für Papyrologie und Epigraphik, pages 269–278.
7. Erkan G, Radev DR. Lexrank: Graph-based lexical centrality as salience in text summarization. Journal of artificial intelligence research. 2004;22:457–79.

8. Farzindar, A. (2004). Atefeh farzindar and guy lapalme,'letsum, an automatic legal text summarizing system in t. gordon (ed.), legal knowledge and information systems. jurix 2004: The seventeenth annual conference. amsterdam: Ios press, 2004, pp. 11-18. In Legal knowledge and information systems: JURIX 2004, the seventeenth annual conference, volume 120, page 11. IOS Press.
9. Farzindar, A. and Lapalme, G. (2004). Legal text summarization by exploration of the thematic structure and argumentative roles. In Text Summarization Branches Out, pages 27–34.
10. Gelbukh, A. (2011). Computational Linguistics and Intelligent Text Processing: 12th International Conference, CICLing 2011, Tokyo, Japan, February 20-26, 2011. Proceedings. Springer Science & Business Media.
11. Ghosh, S., Dutta, M., and Das, T. (2022a). Indian legal text summarization: A text normalisation-based approach. arXiv preprint arXiv:2206.06238.
12. Ghosh, S., Dutta, M., and Das, T. (2022b). Indian Legal Text Summarization: A Text Normalization-based Approach.
13. Gulden C, Kirchner M, Schüttler C, Hinderer M, Kampf M, Prokosch H-U, Toddenroth D. Extractive summarization of clinical trial descriptions. International Journal of Medical Informatics. 2019;129:114–21.
14. Hoecker A, Kartvelishvili V. Svd approach to data unfolding. Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment. 1996;372(3):469–81.
15. Huang, X., Liu, Y., Wang, J. J., Gao, T., Zhao, M., Huang, F., Liu, X., Chen, S., and Wu, Y. (2021). Legal pegasus: The transformer-based legal language modeling toolkit. arXiv preprint arXiv:2102.12349.
16. Hussein KW, Sani NFM, Mahmod R, Abdullah MT. Enhance luhn algorithm for validation of credit cards numbers. Int J Comput Sci Mob Comput. 2013;2(7):262–72.
17. Kanapala A, Pal S, Pamula R. Text summarization from legal documents: a survey. Artificial Intelligence Review. 2019;51:371–402.
18. Khanam, M. H. and Sravani, S. (2016). Text summarization for telugu document. IOSR Journal of Computer Engineering (IOSR-JCE), 18(6):25–28.
19. Kumar, S., Reddy, P. K., Reddy, V. B., and Singh, A. (2011). Similarity analysis of legal judgments. In Proceedings of the fourth annual ACM Bangalore conference, pages 1–4.
20. Larson, R. R. (2010). Introduction to information retrieval.
21. Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., and Zettlemoyer, L. (2019). Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. arXiv preprint arXiv:1910.13461.
22. Mihalcea, R. and Tarau, P. (2004). Textrank: Bringing order into texts. In Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing, pages 404–411. Association for Computational Linguistics.
23. Ng, J.-P. and Abrecht, V. (2015). Better summarization evaluation with word embeddings for rouge. arXiv preprint arXiv:1508.06034.
24. Ozsoy MG, Alpaslan FN, Cicekli I. Text summarization using latent semantic analysis. Journal of Information Science. 2011;37(4):405–17.
25. Parikh, V., Mathur, V., Mehta, P., Mittal, N., and Majumder, P. (2021). LawSum: A weakly supervised approach for Indian Legal Document Summarization.
26. Polsley, S., Jhunjhunwala, P., and Huang, R. (2016). Casesummarizer: A system for automated summarization of legal texts. In Proceedings of COLING 2016, the 26th international conference on Computational Linguistics: System Demonstrations, pages 258–262.
27. Rogers, I. (2002). The google pagerank algorithm and how it works.
28. Samei, B., Estiagh, M., Keshtkar, F., and Hashemi, S. (2014). Multi-document summarization using graph-based iterative ranking algorithms and information theoretical distortion measures. In FLAIRS Conference.
29. Saravanan, M., Ravindran, B., and Raman, S. (2008). Automatic identification of rhetorical roles using conditional random fields for legal document summarization. In Proceedings of the Third International Joint Conference on Natural Language Processing: Volume-I.
30. Shukla, A., Bhattacharya, P., Poddar, S., Mukherjee, R., Ghosh, K., Goyal, P., and Ghosh, S. (2022). Legal case document summarization: Extractive and abstractive methods and their evaluation. arXiv preprint arXiv:2210.07544.
31. Venkataramana, A., Srividya, K., and Cristin, R. (2022). Abstractive text summarization using bart. In 2022 IEEE 2nd Mysore Sub Section International Conference (MysuruCon), pages 1–6. IEEE.
32. Verma, P. and Om, H. (2016). Extraction based text summarization methods on user's review data: A comparative study. In Smart Trends in Information Technology and Computer Communications: First International Conference, SmartCom 2016, Jaipur, India, August 6–7, 2016, Revised Selected Papers 1, pages 346–354. Springer.
33. Verma, P. and Om, H. (2018). Fuzzy evolutionary self-rule generation and text summarization. In 15th International Conference on Natural Language Processing, page 115.
34. Verma, P. and Om, H. (2019a). Collaborative ranking-based text summarization using a metaheuristic approach. In Emerging Technologies in Data Mining and Information Security: Proceedings of IEMIS 2018, Volume 3, pages 417–426. Springer.
35. Verma P, Om H. Mcrmr: Maximum coverage and relevancy with minimal redundancy based multi-document summarization. Expert Systems with Applications. 2019;120:43–56.
36. Verma P, Om H. A novel approach for text summarization using optimal combination of sentence scoring methods. Sādhanā. 2019;44:1–15.
37. Verma, P. and Om, H. (2019d). A variable dimension optimization approach for text summarization. In Harmony Search and Nature Inspired Optimization Algorithms: Theory and Applications, ICHSA 2018, pages 687–696. Springer.
38. Verma P, Pal S, Om H. A comparative analysis on hindi and english extractive text summarization. ACM Transactions on Asian and Low-Resource Language Information Processing (TALLIP). 2019;18(3):1–39.
39. Verma P, Verma A. Accountability of nlp tools in text summarization for indian languages. Journal of scientific research. 2020;64(1):258–63.
40. Verma P, Verma A. A review on text summarization techniques. Journal of scientific research. 2020;64(1):251–7.
41. Verma P, Verma A, Pal S. An approach for extractive text summarization using fuzzy evolutionary and clustering algorithms. Applied Soft Computing. 2022;120: 108670.
42. Verma P, Verma A, Pal S. A fusion of variants of sentence scoring methods and collaborative word rankings for document summarization. Expert Systems. 2022;39(6): e12960.
43. Wang, D., Zhu, S., Li, T., and Gong, Y. (2009). Multi-document summarization using sentence-based topic models. In Proceedings of the ACL-IJCNLP 2009 conference short papers, pages 297–300.
44. William, H. (2004). The principles of readability. eric. Online Submission.
45. Williams, R. V. (2010). Hans peter luhn and herbert m. ohlman: Their roles in the origins of keyword-in-context/permutation automatic indexing. Journal of the American Society for Information Science and Technology, 61(4):835–849.

46. Yang S, Zhang S, Fang M, Yang F, Liu S. A hierarchical representation model based on longformer and transformer for extractive summarization. Electronics. 2022;11(11):1706.